Contents

Figures ........................................... v

Tables .......................................... vii

About this document ........................ i x
Who should use this document ............ i x
How to use this document .................... i x
Where to find more information ........... i x
How to send your comments to IBM ...... xi

Summary of changes ......................... xiii
z/OS Version 2 Release 1 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 SYSDT Data Sets Call — SSI Function 14
User Destination Validation/Conversion — SSI Function Code 11 .......... 39
Verify Subsystem Function Call — SSI Function Code 15 ........ 44
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 .......... 137
SYSOUT Application Program Interface (SAPI)
— SSI Function Code 79 .................. 143
Extended Status Function Call — SSI Function Code 80 ........ 185
JES Properties — SSI Function Code 82 ........ 250
JES device information services — SSI function code 83 .......... 341
Modify Job Function Call — SSI Function Code 85 ........... 431
System Information ..................... 457

Chapter 4. Setting Up Your Subsystem 461
Function Routines/Function Codes .... 461
Environment ............................. 462
Recovery and Integrity ............... 462
Placement of Function Routines .... 463
Do You Need a Subsystem Address Space? .... 463
Defining Your Subsystem ............... 464
Naming your subsystem ................ 464
Passing parameters .................. 465
The primary subsystem .............. 465
Providing a Routine to Initialize Your Subsystem .... 466
What Your Subsystem Initialization Routine Can Do ........ 466
How to Initialize Your Subsystem .... 467
Passing Accounting Parameters to Your Subsystem 469
Processing the SUBPARM Option .......... 469
Example ................................ 470

Chapter 5. Services for Building and
Using Your Subsystem .................. 471
Adding Your Subsystem .................. 471
Using the IEFSSNxx Parmlib Member .... 471
Using the IEFSSI macro ............... 472
Using the SETSSI command .......... 472
Initializing Your Subsystem .......... 472
Coding the Initialization Routine .... 473
Defining What Your Subsystem Can Do .... 474
Building the SSVT ....................... 475
Changing What Your Subsystem Can Do .... 476
Enabling Your Subsystem for New Functions .... 476
Disabling Previously Supported Functions .......... 477
Associating a New Function Routine with a Supported Function Code .......... 477
Activating Your Subsystem ............ 478
Using the IEFSSVT macro ........... 478
Using the IEFSSI macro ............ 478
Deactivating Your Subsystem .......... 479
Swapping Subsystem Functions .... 480
Storing and Retrieving Subsystem-specific Information .... 480
Storing Subsystem-specific Information .......... 480
Retrieving Subsystem-specific Information .......... 481
Defining Subsystem Options ........... 481
Responding to the SETSSI Command ...... 481

© Copyright IBM Corp. 1988, 2013
Chapter 6. SSI Function Codes Your Subsystem Can Support ............... 487
SSI Function Code Descriptions .................. 487
End-of-Task Call — SSI Function Code 4 .......... 487
End-of-Address Space (End-of-Memory) Call — SSI Function Code 8 .......... 491
WTO/WTOR Call — SSI Function Code 9 .......... 495
Command Processing Call — SSI Function Code 10 ....................... 511
Delete Operator Message — SSI Function Code 14 ............... 519
Help Call — SSI Function Code 48 ............... 522
Early Notification of End-of-Task Call — SSI Function Code 50 .............. 524
Request Subsystem Version Information Call — SSI Function Code 54 .......... 528
SMF SUBPARM Option Change Call — SSI Function Code 58 ............... 535
Tape Device Selection Call — SSI Function Code 78 ....................... 538

Chapter 7. Troubleshooting Errors in Your Subsystem ............... 555
Handling Initialization Errors .................... 555
Handling function request errors .................. 556
Capturing the System Dump ....................... 556
Identifying the Type of Error ..................... 557
Determining the Cause of the Error ............... 558

Chapter 8. Examples — Subsystem Interface Routines ............... 561
Example 1 — Subsystem Initialization Routine (TSYSINIT) ............... 561
Example 2 — Subsystem Function Routine (WRITEIT) ............... 566
Example 3 — Subsystem Function Routine (DELETEIT) ............... 568
Example 4 — Subsystem Function Routine (LISTEN) ............... 569
Example 5 — Subsystem Requesting Routine (TSYSCALL) ............... 571

Chapter 9. Using IEFJSVEC with Your Subsystem ............... 575
Defining What Your Subsystem Can Do ............... 575
Building the SSVT ............... 575
Changing What Your Subsystem Can Do ............... 580
Enabling Your Subsystem for New Functions ............... 580
Disabling Previously Supported Functions ............... 585

Appendix. Accessibility ............... 591
Accessibility features ............... 591
Using assistive technologies ............... 591
Keyboard navigation of the user interface ............... 591
Dotted decimal syntax diagrams ............... 591

Notices ............... 595
Policy for unsupported hardware ............... 596
Minimum supported hardware ............... 597
Trademarks ............... 597

Index ............... 599
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Processing for a directed request</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Processing for a broadcast request</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Making a Subsystem Request</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Environment at Time of Call for SSI Function Code 1</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Environment at Time of Call for SSI Function Code 11</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>Environment at Time of Call for SSI Function Code 15</td>
<td>46</td>
</tr>
<tr>
<td>7</td>
<td>Environment at Time of Call for SSI Function Code 20</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>Environment at Time of Call for SSI Function Code 21</td>
<td>57</td>
</tr>
<tr>
<td>9</td>
<td>Environment at Time of Call for SSI Function Code 54</td>
<td>62</td>
</tr>
<tr>
<td>10</td>
<td>Environment at Time of Call for SSI Function Code 70</td>
<td>78</td>
</tr>
<tr>
<td>11</td>
<td>Environment at Time of Call for SSI Function Code 71, Spool Read Subfunction</td>
<td>90</td>
</tr>
<tr>
<td>12</td>
<td>Environment at Time of Call for SSI Function Code 71, JES2 Health Monitor Information Subfunction</td>
<td>98</td>
</tr>
<tr>
<td>13</td>
<td>Environment at Time of Call for SSI Function Code 71, Job Class Information Subfunction</td>
<td>111</td>
</tr>
<tr>
<td>14</td>
<td>Environment at Time of Call for SSI Function Code 71, Convert Device ID Service Subfunction</td>
<td>124</td>
</tr>
<tr>
<td>15</td>
<td>Environment at Time of Call for SSI Function Code 71, Checkpoint Versions Subfunction</td>
<td>130</td>
</tr>
<tr>
<td>16</td>
<td>Environment at Time of Call for SSI Function Code 75</td>
<td>138</td>
</tr>
<tr>
<td>17</td>
<td>Protocol for the SAPI PUT/GET Call</td>
<td>147</td>
</tr>
<tr>
<td>18</td>
<td>Control Blocks of DYNALLOC Call for SAPI-Provided Data Set</td>
<td>150</td>
</tr>
<tr>
<td>19</td>
<td>Protocol for the SAPI COUNT Call</td>
<td>151</td>
</tr>
<tr>
<td>20</td>
<td>Protocol for the SAPI BULK MODIFY Call</td>
<td>153</td>
</tr>
<tr>
<td>21</td>
<td>Environment at Time of Call for SSI Function Code 79</td>
<td>156</td>
</tr>
<tr>
<td>22</td>
<td>Environment at Time of Call for SSI Function Code 80</td>
<td>190</td>
</tr>
<tr>
<td>23</td>
<td>Environment at Time of Call for SSI Function Code 82, NJE Node Information Subfunction</td>
<td>252</td>
</tr>
<tr>
<td>24</td>
<td>Environment at Time of Call for SSI Function Code 82, SPOOL Volume Information Subfunction</td>
<td>271</td>
</tr>
<tr>
<td>25</td>
<td>Environment at Time of Call for SSI Function Code 82, Initiator Information Subfunction</td>
<td>291</td>
</tr>
<tr>
<td>26</td>
<td>Environment at Time of Call for SSI Function Code 82, JESPLEX Information Subfunction</td>
<td>309</td>
</tr>
<tr>
<td>27</td>
<td>Environment at Time of Call for SSI Function Code 82, Job Class Information Subfunction</td>
<td>325</td>
</tr>
<tr>
<td>28</td>
<td>Environment at Time of Call for SSI Function Code 83, the JES device information services</td>
<td>344</td>
</tr>
<tr>
<td>29</td>
<td>Environment at Time of Call for SSI Function Code 85</td>
<td>435</td>
</tr>
<tr>
<td>30</td>
<td>Initializing Your Subsystem</td>
<td>468</td>
</tr>
<tr>
<td>31</td>
<td>Input to the Initialization Routine</td>
<td>473</td>
</tr>
<tr>
<td>32</td>
<td>Subsystem Affinity Service</td>
<td>484</td>
</tr>
<tr>
<td>33</td>
<td>Environment on Entry to the Function Routine for SSI Function Code 4</td>
<td>489</td>
</tr>
<tr>
<td>34</td>
<td>Environment on Entry to the Function Routine for SSI Function Code 8</td>
<td>493</td>
</tr>
<tr>
<td>35</td>
<td>Environment for a Single-line WTO in the SSWT Control Block</td>
<td>500</td>
</tr>
<tr>
<td>36</td>
<td>Environment for the First Line of a Multi-line WTO in the SSWT Control Block</td>
<td>503</td>
</tr>
<tr>
<td>37</td>
<td>Environment for Minor Lines of a Multi-line WTO in the SSWT Control Block</td>
<td>507</td>
</tr>
<tr>
<td>38</td>
<td>Environment for a WTOR in the SSWT Control Block</td>
<td>509</td>
</tr>
<tr>
<td>39</td>
<td>Environment on Entry to the Function Routine for SSI Function Code 10</td>
<td>513</td>
</tr>
<tr>
<td>40</td>
<td>Environment on Entry to the Function Routine for SSI Function Code 48</td>
<td>523</td>
</tr>
<tr>
<td>41</td>
<td>Environment on Entry to the Function Routine for SSI Function Code 50</td>
<td>526</td>
</tr>
<tr>
<td>42</td>
<td>Environment on Entry to the Function Routine for SSI Function Code 54</td>
<td>530</td>
</tr>
<tr>
<td>43</td>
<td>Environment at Time of Call for SSI Function Code 58</td>
<td>536</td>
</tr>
<tr>
<td>44</td>
<td>Environment at Time of Call for SSI Function Code 78</td>
<td>543</td>
</tr>
<tr>
<td>45</td>
<td>Continuation of Environment at Time of Call for SSI Function Code 78</td>
<td>544</td>
</tr>
</tbody>
</table>
## Tables

1. IBM-Defined Keywords .......................... 69  
2. JES Job Information Services Request Types 87  
3. Examples of jobs returned for SSS2JBIL when SSS2JBIH is blank .......................... 165  
4. SAPI Disposition in SSS2 ........................ 169  
5. Summary of output data elements based on input type requested .......................... 187  
6. Examples of jobs returned for STATJBIL when STATJBIH is blank .......................... 198  
7. SSI Function Code 80 Filters .................. 207  
8. JES3 Unsupported Flags and Fields ............... 240  
9. Job phase text level ............................ 245  
10. Job phase text level ............................. 246  
11. JES Properties Request Types .................. 250  
12. SSI 85 actions, JESJOBS class entities and Required access .......................... 432  
13. Examples of jobs returned for SSJMJBIL when SSJMJBIH is blank .................. 446  
14. SSI updating of multi-line messages ............ 507  
15. Relationship between System and User Criteria .......................... 540
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
• Chapter 3, “SSI Function Codes Your Program Can Request,” on page 13

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

• Chapter 4, “Setting Up Your Subsystem,” on page 461
• Chapter 5, “Services for Building and Using Your Subsystem,” on page 471
• Chapter 6, “SSI Function Codes Your Subsystem Can Support,” on page 487
• Chapter 7, “Troubleshooting Errors in Your Subsystem,” on page 555

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.
How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:
1. Send an email to mhvrdfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:
• Your name and address.
• Your email address.
• Your telephone or fax number.
• The publication title and order number:
  z/OS V2R1.0 MVS Using the Subsystem Interface
  SA38-0679-01
• The topic and page number that is related to your comment.
• The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:
• Contact your IBM service representative.
• Call IBM technical support.
• Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/)
Summary of changes

Changes made to z/OS V2R1 as updated December 2013

**New**
- Added JESJOBS security details to “Modify Job Function Call — SSI Function Code 85” on page 431.

**Changed**
None.

**Deleted**
None.

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):
- z/OS Migration
- z/OS Planning for Installation
- z/OS Summary of Message and Interface Changes
- z/OS Introduction and Release Guide
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 IEFSSNx 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. The following IBM subsystems use the SSI:

• JES2
• JES3
• IMS
• Tivoli® NetView® for z/OS®

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)
Introduction

- 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 487 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...
Introduction

separate function routines; the subsystem-provided programs that process the function identified by the function codes.

**Note:** Prior to z/OS V1R12, the subsystem initialization routines specified in parmlib member IEFSSNxx were invoked in the sequence they appeared and under a task that never terminated. From z/OS V1R12, the initialization routines are invoked in parallel after the BEGINPARALLEL keyword in parmlib member IEFSSNxx is processed, and no longer run under a permanent task when they are run in parallel.

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](https://www.ibm.com) 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](https://www.ibm.com) for more information on the RESMGR macro.

If you decide that you need a subsystem, see Chapter 4, “Setting Up Your Subsystem,” on page 461 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 575, however IBM recommends that you use the services described in Chapter 5, “Services for Building and Using Your Subsystem,” on page 471 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.

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

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 SSORTRTRY. 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.
Making a Request

SSOB Function Dependent Area

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

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

Subsystem Identification Block (SSIB)

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

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

- An alternate JES2
- The master subsystem

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

Before you make an SSI request you need to evaluate whether the subsystem name provided by the system in the life-of-job SSIB is the correct subsystem for the function you are requesting. The system provides the subsystem name in the life-of-job SSIB, based on whether the unit of work is a batch job (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 IEFACRT, 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 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
Making a Request

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>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>
Making a Request

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.

<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>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem does not support this function.</td>
</tr>
</tbody>
</table>

Check the Return Code

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

Return Code
(Decimal)
Meaning

SSRTOK (0)
Successful completion — the request went to the subsystem.

SSRTNSUP (4)
The subsystem does not support this function.
Making a Request

SSRTNTUP (8)
The subsystem exists, but is not active.

SSRTOSS (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 (SSOBRRET), 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 SSOBRRET, and take appropriate action.
   - Free the storage.

Example 5 in Chapter 8, “Examples — Subsystem Interface Routines,” on page 561 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 the directed function codes that your program can request. IBM subsystems provide these function codes.* The following table lists each supported SSI function code, its function, the subsystems that support the function and the type of subsystem request.

Attention: Your program can request ONLY the following SSI function codes:

<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/JES3</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>
<tr>
<td>83</td>
<td>JES Device information services</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>85</td>
<td>Modify job</td>
<td>JES2</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,” on page 561 for a coding example of a routine making a request of a subsystem.
SSI Function Code 1

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>External writer variable</th>
<th>External writer value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Supervisor state</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 SSOB and SSSO control blocks can reside in storage above 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

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

**Register**  
**Contents**

1. Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.

13. Address of a standard 18-word save area.

**Input Parameters**

Input parameters for the function routine are:

- SSOB
- SSIB
- SSSO

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

- **Field Name**
- **Description**

  **SSOBID**  
  Identifier 'SSOB'

  **SSOBLLEN**  
  Length of the SSOB (SSOBHSIZ) control block
SSI Function Code 1

**SSOBFUNC**
SSI function code 1 (SSOBSOUT)

**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 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>SSSOLEN</td>
<td>Length of the SSSO control block—set with SSOSIZE 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 SSSOCTRL bit (in flag byte SSSOFLG2) is zero, JES provides the name of the next data set to be allocated.

Flag Value is nonzero: Setting this flag to nonzero 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 following updates described through the bit settings should be performed through the appropriate dynamic text unit keys only. See "Processing Flow for Single Data Set Requests" on page 29 for more information on performing disposition processing on single data sets.

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

- **SSSOSETC** — 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** — Delete the selected data sets.
  This is only valid for JES3 update requests that have data sets on the WRITER or HOLD queue.

- **SSSOROUT** — 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** — Release all selected data sets that are eligible for printing or further processing by JES3.
  This is only valid for JES3 update requests, and for the selected data sets on the JES3 HOLD queue.
  Bits SSSOSETC and SSSOROUT might also be issued concurrently.

**SSSOVER**

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

**SSSOFGLG1**

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:
SSI Function Code 1

- JES3 WRITER queue only (if SSLOAD is off)
- JES3 WRITER and HOLD queues (HOLD=EXTWTR only) if SSLOAD 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 SSSOJOB1 field.

- **SSSOSPQM** — Use the external writer name (the second positional parameter in the SYSOUT= keyword on the DD JCL statement), or user ID 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 (SSOSFR8).

  2. 8-character form name
     
     Use the 8-character form name field (SSSOFOR8), and set both the SSSOSFRM and SSSOSFR8 bits. If using an 8-character form, place the name of the form in the SSSOFOR8 field, and not in the SSSOFORM field.

- **SSSOSFR8** — Use the 8-character form name field (SSSOFOR8) as a selection criterion. Make sure that you do not use the 4-character form name field (SSSOFORM). JES ignores the SSSOFORM field.
  
  If your external writer sets the this bit, the SSSOSFRM bit must also be set to indicate selection by either 4-character or 8-character forms.

**SSSOFLG2**

Flag byte

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

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

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

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

- **SSSOPSEE** — Process SYSOUT extension
  Your external writer sets this bit on if SOEXT=YES was specified on the IEFSSSO macro invocation to indicate that additional fields exist in the IEFSSSO.

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

**SSSOJOBN**
Job name for a retrieval request.

Your external writer:
- Sets the value of the specific job name in the SSSOJOBN field left-justified, and padded to the right with blank (X’40’) characters.
- Sets the SSSOSJBN bit for this selection to occur.

**SSSOJOBI**
Job ID for a retrieval request.

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

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

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

**SSSOFLGA**
Flag byte containing the SSSOUSER and SSSOWTRN bits.
Your external writer can set either the SSSOUSER bit (user ID), or the SSSOWTRN bit (writer name), but not both.

If your external writer sets the SSSOUSER bit, the value contained in the SSSOPGMN field is a user ID. Setting the SSSOUSER bit for user ID 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.
SSI Function Code 1

- The JESSPOOL security class is active. For information on the JESSPOOL security class, see z/OS Security Server RACF Security Administrator’s Guide.

If your external writer sets the SSSOWTRN bit, the value contained in the SSSOPGMN field is a writer name. Setting the SSSOWTRN bit for writer name selection allows your external writer to access the data set if both:
- A data set resource profile in the security product (RACF) exists.
- The JESSPOOL security class is active.

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

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

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

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

For an update request (JES3 only), your external writer:
- Sets the value of the specific destination in the SSSODEST field left-justified, and padded to the right with blank (X'40') characters.
- Sets the SSSOROUT bit for this selection to occur.

SSSOPGMN
Name selected for a retrieval request.

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

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

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

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

SSSODSN
Data set name

For the initial retrieval request, your external writer sets this field to binary zeros. JES returns the name of a SYSOUT data set in this SSSODSN field.
In a subsequent dynamic allocation, your external writer uses the name of this data set for processing purposes. See "Processing Flow for Single Data Set Requests" on page 29 for more information on this field.

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

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

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

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

Your external writer:
- Sets the value of the form name in the SSSOFORM field left-justified and padded to the right with blank ('X'40') characters.
- Sets the SSOSFRM bit for this selection to occur.
  - If the SSOSFR8 bit is also set, specify the 8-character form name in the SSSOFOR8 field, and this SSSOFORM field is not used.
  - If the SSOSFR8 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 SSOSCLS 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 invocation. IBM recommends that you specify SOEXT=YES on the IEFSSSO invocation, as additional information is returned to the external writer.

**Field Name**  
**Description**

**SSSOFLG5**  
Flag byte

Your external writer can set one or more of the following selection bits:
- **SSSOTKNR** — Security token length and security token version information set.
SSI Function Code 1

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

**SSSOSECT**

Address of a caller-supplied area in which the security token is returned.

If the SSSOTKNR bit has also been set, your external writer must also provide the length and version of the token that is returned at the address specified in the SSSOSECT field. JES returns the security token in the format specified. See the SSSOTKNR bit and the SSSOTKNG bit on output for additional information.

If the SSSOTKNR bit has also been set off:

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

**SSSOFOR8**

8-character form name selected

Your external writer must have set both the SSOSFRM and SSOSFR8 bits for this selection to occur.

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

If the SSOSFR8 bit is also set, the 4-character form name in SSOSFORM is ignored. JES uses the name in the SSSOFOR8 field as the forms selection criteria.

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

**Output Register Information**

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

<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>
SSI Function Code 1

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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOORTOK (0)</td>
<td>Successful completion.</td>
</tr>
</tbody>
</table>

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
SSI Function Code 1

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.

SSSOINV (12)
Invalid search argument.

The job ID specified in the SSSOJOB1 field failed syntactical parsing, or both the SSSOWTRN bit and the SSSOUER bit had also been set in the SSSOFLGA flag byte.

SSSOUP (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.

SSSOINV (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.

SSSODST (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](https://www.ibm.com/support/docview.wss?uid=swg27049284) for the correct syntax) or a valid NJE destination was supplied (an external writer cannot select work destined for NJE nodes).

SSSOAUTH (32)
Authorization failed

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

SSSOTKNM (36)
Token map failed

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

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSSOFLG2</strong></td>
<td>Flag byte</td>
</tr>
<tr>
<td>v</td>
<td><strong>SSSODDST</strong> — DD name set in the extension.</td>
</tr>
<tr>
<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>
<td></td>
</tr>
<tr>
<td><strong>SSSOCOPY</strong></td>
<td>Number of copies.</td>
</tr>
<tr>
<td>JES provides the data set to your external writer as many times as the copy count from the creating JCL specifies it.</td>
<td></td>
</tr>
<tr>
<td>The value of this field depends on which JES is being used:</td>
<td></td>
</tr>
<tr>
<td><strong>JES2:</strong> This field is always set to '1' on a retrieval request.</td>
<td></td>
</tr>
<tr>
<td><strong>JES3:</strong> This field is always set to '1' on a retrieval request.</td>
<td></td>
</tr>
<tr>
<td><strong>SSSOJOBN</strong></td>
<td>Job name associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td><strong>SSSOJOBI</strong></td>
<td>Job ID associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td><strong>SSSOCLAS</strong></td>
<td>Class associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<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>
<td></td>
</tr>
<tr>
<td><strong>SSSOMLRL</strong></td>
<td>Maximum logical record length associated with the returned data set.</td>
</tr>
<tr>
<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>
<td></td>
</tr>
<tr>
<td><strong>SSSODEST</strong></td>
<td>Destination associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td><strong>SSSOPGMN</strong></td>
<td>Writer name or user ID associated with the returned data set (retrieval request, if available).</td>
</tr>
<tr>
<td>The specific information returned depends on the setting of the SSSOWTRN or SSSOUSER bits (retrieval request).</td>
<td></td>
</tr>
<tr>
<td><strong>JES2:</strong> If neither the SSSOWTRN or SSSOUSER bits are specified, then this field contains the writer name associated with this data set.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> The SSSOPGMN field is filled in regardless of whether the SSSOSPGM bit is set. It contains a user ID only when the SSSOUSER bit is set.</td>
<td></td>
</tr>
<tr>
<td><strong>SSSODSN</strong></td>
<td>Returned data set name (retrieval request).</td>
</tr>
</tbody>
</table>
SSI Function Code 1

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

The returned data set name is in the fully-qualified, form of:
userid.jobname.jobid.dsidentifier.dsname (JES2) or
userid.jobname.jobid.dsnumber.dsname (JES3).

SSSOFORM
First four characters of the form name associated with the returned data set name (retrieval request). The SSSOFORM 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.

Field Name Description

SSSOFGLG5
Flag byte
• SSSOTKNG — Token mapped.
  JES sets the SSSOTKNG bit if the token was returned to the version requested by your external writer through the setting of the SSSOTKNGR on the retrieval request.
  The SSSOFLG5 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:
SSI Function Code 1

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

**SSSOSTPD**
Data set step name of the returned data set provided if:
- JES set the SSSODDST bit upon return to your external writer.
- A retrieval request is being made.

**SSSODDND**
Data set ddname of the returned data set provided if:
- JES set the SSSODDST bit upon return to your external writer.
- A retrieval request is being made.

**SSSOSECT**
Pointer whose contents remain unchanged from the retrieval request, but whose address now points to the returned data set’s SAF token.

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

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

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

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

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

Your external writer must be in AMODE 31 to access this data. The data is in the following format:
- A 1-byte field containing the number of pairs that follow.
- Zero or more accounting pairs, each of the form:
  - A 1-byte field containing the length of the accounting string.
  - The actual accounting string itself with the length that is specified in the first byte.

A length of zero indicates an omitted field.

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

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

The SSSOGNVA flag is set if the field is valid.
## Processing Flow for Single Data Set Requests

Your external writer can process single data set requests by:

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

### Processing One Data Set at a Time (JES2 and JES3)

Your external writer can use the following steps for proper selection, allocation, retrieval, and unallocation of an individual SYSOUT data set:

1. Build the appropriate SSOB and SSSO control blocks for the request according to the individual selection criteria desired.
2. Issue the IEFSSREQ macro asking JES for the name of a data set.
   - This step includes setting the SSSOUFLG flag byte to X'00', and the SSSOCTRL bit to 0.
   - Upon return the name of the data set is in the SSSODSN field.
3. Allocate the data set through dynamic allocation (DYNALLOC macro).
   - Your external writer can use the following text units:
     - **DALDSNAM**
       - Used with the returned name from the SSSODSN field.
     - **DALSSREQ**
       - Indicates a request that JES needs to handle. The parameter value in this text unit is the name of the subsystem that processed the IEFSSREQ macro.
     - **DALRTDDN**
       - Indicates the DDNAME associated with the allocation be returned to the caller of DYNALLOC. Your external writer then places this DDNAME in the DCB macro that needs to open the SYSOUT data set as input for your reads.
       - Prime the parameter in this text unit with blank (X'40') characters before issuing the DYNALLOC macro. This text unit is returned from DYNALLOC with the correct DDNAME.
       - Your external writer will also use this DDNAME in the dynamic unallocation of the data set when performing unallocation processing.
4. Open the program-supplied DCB.
   - Move the returned DDNAME from the DALRTDDN field as the DCB's DDNAME before issuing the OPEN.
   - The following is an example of a DCB that may be used to obtain the records:

```
INDCB DCB DSORG=PS,MACRF=GL,BUFNO=1,
        SYNA=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 [DFSMS Macro Instructions for Data Sets](https://www.ibm.com/docs/en/zos/2.4.0?topic=dfsms-instructions).
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:
SSI Function Code 1

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

- **DUNOVDSP**
  This text unit indicates a disposition override. You must specify one of the following:
  - Keep the data set on the spool. For JES2, when you specify keep as the disposition, JES2 assumes that the external writer has failed and treats the next PSO request as if you had set the SSSOCTRL bit.
  - Delete the data set from the spool.
  If you are performing immediate disposition and wish to delete the data set, use the X'04' value as the disposition flag. Otherwise, you can use the X'08' value to keep the data set on the spool.

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

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

- **DUNOVSNH**
  For JES2, the data set selected is already on the output queue with a disposition of WRITE or KEEP, and this text unit is not specifiable.
  For JES3, this text unit removes the data set from the HOLD queue, and puts it on the WRITER queue.

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

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

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

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

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

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

- It is initiated from the user's address space

  Therefore, it is a completely separate MVS job. This separation allows for processing overlap and address space integrity. In JES3, because the SSI is involved for scheduling communication, the external writers may exist on local processors as well as the global processor.
SSI Function Code 1

- It is functionally independent of JES
  There is neither a print processor running in the JES2 address space, nor a writer DSP running in the JES3 global address space.

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

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

- It allows the installation to control the selection of work
  Standard external writers select work through a SYSOUT class dedicated to external writers or a writer name. JES2 and JES3 handle external writer processing differently.
  **JES2**: The work to be processed is located on the output queue, and has an OUTDISP of WRITE or KEEP. However, conversational data sets, which include data sets located on the output queue with an OUTDISP of HOLD or LEAVE, are not processed in JES2 by the standard external writer. These data sets are destined to be processed by TSO/E users through the OUTPUT command.
  **JES3**: The work to be processed is located on either the WRITER queue, or the HOLD queue. However, IBM recommends that you process only data sets on the HOLD queue (either by specific SYSOUT class specification as defined on the initialization statement, or by writer name).

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

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

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

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

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

- It handles update requests
SSI Function Code 1

An update request is allowable only for JES3. JES3 allows update requests through the IEFSSREQ macro for one or more data sets whose selection criteria matches the criteria supplied by the external writer directly through the IEFSSREQ macro. However, individual data sets obtained through the IEFSSREQ retrieval/allocation process should have their attributes changed during the dynamic unallocation as described in the retrieval information above. Update requests may be performed on more than one data set at a time when the external writer:
- Issues the IEFSSREQ macro
- Does not specify a specific data set name within the SSSO control block.
This is a powerful facility. However, you should be careful when using it, as the scope of such a modification may be large when more than one data set is involved.

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

• Spool access is provided by sequential access methods
SAM is used to obtain the records of the SYSOUT data set from the spool. This implies familiar coding techniques, such as OPENs, GETs, and CLOSEs.

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

• It may wait for JES to post it for new work if idle
When JES sends a no-work-available notice to the external writer, it may sit idle until it receives a POST from JES, telling it that work is available. It may then ask JES again for the newly available work. This process uses WAIT and POST logic with an ECB returned to the external writer.
JES2 does not POST the external writer if invoked from a batch job; it must be a started task for such posting to occur.

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

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

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

*Writer Communication Area Contents:* The fields in the writer communication area contain:
SSI Function Code 1

- 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.

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 'PRT803'. *(SHOWS AN EXAMPLE OF USING THE AVAILABLE DYNAMIC *
**  ALLOCATION TEXT UNIT TO CHANGE THE ATTRIBUTES OF THE *
**  RECEIVE SYSOUT DATA SET DURING UNALLOCATION.) *
** 6. THE PROGRAM THEN CYCLES BACK AND ASKS JES FOR THE NEXT *
**  DATA SET (GOES TO STEP 1). *
**
* NAME OF MODULE: SSIREQ01 *
* REGISTER USE: *
* 0 PARM REGISTER *
* 1 PARM REGISTER *
* 2 SSOB *
* 3 SSSO *
* 4 DCB *
* 5 RB *
* 6 MAX RECORD LENGTH *
* 7 DUMP CODE *
* 8 ABEND VALUE REGISTER *
* 9 IEFSSREQ RETURN CODES *

Chapter 3. SSI Function Codes Your Program Can Request  33
SSI Function Code 1

* 10  BASE REGISTER  *
* 11  TEXT RECORD STRUCTURE PTR  *
* 12  UNUSED  *
* 13  SAVE AREA CHAIN REGISTER  *
* 14  PARM REGISTER / RETURN ADDR  *
* 15  PARM REGISTER / COND CODE  *
*
* ATTRIBUTES: SUPERVISOR STATE, AMODE(31), RMODE(24)  *
*
* NOTE: THIS IS A SAMPLE.  *
*********************************************************************
**
TITLE '- EQUATES'
*********************************************************************
* GENERAL EQUATES *
*********************************************************************
EQUHOBON EQU X'80000000' HIGH ORDER BIT ON
FF EQU X'FF' ALL BITS ON IN A BYTE
*********************************************************************
* AFTER COMPARE INSTRUCTIONS *
*********************************************************************
GT EQU 2 A HIGH
LT EQU 4 A LOW
NE EQU 7 A NOT EQUAL B
EQ EQU 8 A EQUAL B
GE EQU 11 A NOT LOW
LE EQU 13 A NOT HIGH
*
*********************************************************************
* AFTER ARITHMETIC INSTRUCTIONS *
*********************************************************************
OV EQU 1 OVERFLOW
PLUS EQU 2 PLUS
MINUS EQU 4 MINUS
NZERO EQU 7 NOT ZERO
ZERO EQU 8 ZERO
ZEROS EQU 8 ZERO
NMINUS EQU 11 NOT MINUS
NOV EQU 12 NOT OVERFLOW
NPLUS EQU 13 NOT PLUS
*
*********************************************************************
* AFTER TEST UNDER MASK INSTRUCTIONS *
*********************************************************************
ALLON EQU 1 ALL ON
MIXED EQU 4 MIXED
NALLOFF EQU 5 ALLON+MIXED
ALLOFF EQU 8 ALL OFF
NALLON EQU 12 ALLOFF+MIXED
*********************************************************************
* ABEND CODE INDICATIONS *
*********************************************************************
BADR15 EQU 1 IEFSSREQ R15 NON-ZERO
BADRETN EQU 2 SSOBRETN NON-ZERO AND NOT 8
BAD599A EQU 3 DYNALLOC ALLOC FAILED
BADOPEN EQU 4 OPEN DCB FAILED
BADSS99U EQU 5 DYNALLOC UNALLC FAILED
BADRLEN EQU 6 PSO DATASET TOO LARGE (RECLLEN)
*********************************************************************
* GENERAL PURPOSE REGISTERS *
*********************************************************************
R0 EQU 0 PARM REGISTER
R1 EQU 1 PARM REGISTER
R2 EQU 2 SSOB
R3 EQU 3 SSO
R4 EQU 4 DCB
Chapter 3. SSI Function Codes Your Program Can Request
SSI Function Code 1

* WRITER NAME, NOT USERID
MVC SSOPGMN,=CL8'ANDREW' IND. CORRECT WRITER NAME
* THAT IS USED AS SELECTION
OI SSOFGLG2,SSSOPSEE IND. LONG FORM OF IEFSSSO
*********************************************************************
* NOW GO TAP JES ON THE SHOULDER FOR A DATASET! *
*********************************************************************
NEXTDS DS 0H GET NEXT DSNAME FROM JES
MODESET MODE=SUP GET INTO SUPERVISOR STATE
LR R1,R2 R1=ADDRESS OF SSOB
O R1,=A(EQUHOBON) TURN ON THE HIGH-ORDER BIT
ST R1,MYSSOBPT SAVE POINTER FOR SSREQ
LA R1,MYSSOBPT POINT TO SSOB POINTER
IEFSSREQ , GO TO JES FOR A DATASET
MODESET MODE=PROB BACK TO PROBLEM STATE
LA R8,BADR15 ASSUME BAD REG 15 RETURN
LTR R9,R15 DID THE IEFSSREQ WORK OK?
BC NZERO,ABEND NOT GOOD...TAKE AN ABEND
LA R8,BADRETN ASSUME BAD SSORETN
ICM R9,B'1111',SSOBRETN 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,BADRLEN ASSUME SIZE TOO LARGE FOR WTO
SR R6,R6 CLEAR REG 6
ICM R6,B'0011',SSSOMLRL GET MAX RECORD LENGTH
CH R6,=H'150' IS MAX RCD LENGTH>150??
BC GT,ABEND YES - TIME FOR US TO GO HOME
STH R6,RECLEN SAVE MAX RECORD LENGTH
LA R5,MY99RB PT TO RB
USING S99RB,R5 ADDRESSABILITY TO THE RB
XC S99RB(RBLEN),S99RB ZERO THE RB
MVI S99RBLN,RBLEN RB LENGTH
MVI S99VERB,S99VRBAL RB VERB CODE=ALLOC
LA R1,MY99TPTA ADDR SVC 99 ALLOC TU PTRS
ST R1,S99TXTPP STORED IN RB
LA R1,MY99RBPT PT TO RB POINTER
MVC TXTDSNAM,SSSODSN MOVE DATASET NAME TO BE ALLOCATED
DYNALLOC ISSUE DYNAMIC ALLOCATION
LA R8,BADS99A ASSUME IT DIDN'T WORK
LR R9,R1 COPY FOR DUMP
LTR R15,R15 SVC 99 WORK OKAY??
BC NZERO,ABEND NO, TAKE A DUMP
*********************************************************************
* SYSOUT DATASET ALLOCATED OKAY. MOVE RETURNED DDNAME INTO *
* THE DCB PRIOR TO OPENING IT. *
*********************************************************************
LA R4,INDCB PT TO THE INPUT DCB
USING IHADCB,R4 ADDRESSABILITY
MVC DCBDDNAM(8),TXTDDA99 MOVE IN RETURNED DDNAME
MVC TXTDDU99,TXTDDA99 SAVE FOR UNALLOCATION
MVC DCBLRECL,SSSOMLRL MOVE MAX LENGTH RECORD IN
* OPEN INDCB OPEN THE DCB
LA R8,BADOOPEN ASSUME THE OPEN FAILED
LR R9,R4 COPY FOR DUMP
TM DCBOFLG5,DCBOFOPN DID IT WORK?
BC NOPE, TAKE A DUMP
BC ALLOFF,ABEND ABEND NOPE, TAKE A DUMP
TITLE '- GET THE RECORDS - DISPLAY TO PROGRAM'
GETNEXT DS 0H LOOP FOR READING/DISPLAYING
SSI Function Code 1

* SWITCH TO 24 BIT MODE FOR GET MACRO *

LA R15,SSITO24 SWITCH TO 24 BIT MODE ...
BSM 0,R15 ... FOR RESTRICTED MACRO

** SSITO24 DS 0H **
GET INDCB R1==> RECORD AFTER THE GET
L R15,SSITO31A RETURN TO 31 BIT MODE ...
BSM 0,R15 ... AND CONTINUE

** SSITO31A DC A(SSITO31+EQUHOBON) FOR MODE SWITCHING **

* RETURN TO 31 BIT MODE AND CONTINUE *

** SSITO31 DS 0H **
EX R6,MOVEIT MOVE UP TO 150 BYTES OF REC
LA R11,RECLEN POINT TO RECORD FOR OUTPUT
WTO TEXT=(11),ROUTCDE=11 DISPLAY TO JOBLG
MVI RECTEXT,C' ' CLEAR RECORD OUT...
MVC RECTEXT+1(RECTEXT-1),RECTEXT .FOR NEXT ONE
B GETNEXT GO GET NEXT RECORD

** TITLE '- EODAD ROUTINE' **
MYEODAD DS 0H END-OF-DATASET
CLOSE INDCB CLOSE THE INPUT DCB
DROP R4 IHADCB

* UNALLOCATE THE SYSOUT DATASET, CHANGING CLASS + DESTINATION *

** XC S99RB(RBLEN),S99RB ZERO THE RB **
MVI S99RBLEN,RBLEN RB LENGTH
MVI S99VERB,S99VRBUN RB VERB CODE=UNALLOC
LA R1,MY99PTU ADDR SVC 99 ALLOC TU PTRS
ST R1,S99TXTPP STORED IN RB
LA R1,MY99RPNT PT TO RB POINTER
DYNALLOC ISSUE DYNAMIC UNALLOCATION
LA RB,BADS99U ASSUME IT DIDN'T WORK
LR R9,R1 COPY FOR DUMP
LTR R15,R15 SVC 99 WORK OKAY??
BC NZERO,ABEND NO, TAKE A DUMP
B NEXTDS GO GET NEXT DATA SET

** TITLE '- BAD RETURN FROM IEFSSREQ' **
TESTIT DS 0H

* RB HAS THE 'BADRETN' ASSUMPTION VALUE FOR POSSIBLE ABEND. *
* R9 HAS A NON-ZERO VALUE FROM SSBORETN 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 **
WTO '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,SSSOUTRC POINT TO RETURNED DATA AREA
WAIT ECB=(1) R1==>RETURNED WAIT-FOR ECB
MODESET KEY=NZERO BACK TO ORIGINAL

Chapter 3. SSI Function Codes Your Program Can Request 37
**SSI Function Code 1**

B      NEXTDS WE'RE POSTED - GO GET IT!

TITLE 'CLOSE OUT ROUTINES'

EXIT DS 0H FINAL CALL, RETURN TO MVS

MVI SSSOFGL2, 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, HERE'S WHERE IT LIVES

ADDR=(R2)

L R13, 4(R13) OLD SA PTR

RETURN (14, 12), RC=0 BACK TO MVS

TITLE 'ABEND ROUTINES'

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

* THIS IS THE ABEND ROUTINE. R8 CONTAINS THE PROGRAM REASON CODE, *
* R9 CONTAINS SPECIFIC ERROR/REASON CODE AS RETURNED BY THE       *
* SERVICE ROUTINE.                                                *
*********************************************************************

ABEND DS 0H ISSUE THE ABEND MACRO

ABEND (8), DUMP, STEP TAKE A DUMP IF WANTED

SA DS 9D SAVE AREAS

MYECB DS F DOUBLEWORD FOR WTOR

MYREPLY DS CL1 REPLY AREA FOR WTORS

RESRV DS XL3 ROUND TO FULL WORD

TITLE 'DYNALLOC DATA'

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

* THE FOLLOWING CONTROL BLOCKS ARE FOR DYNAMIC ALLOCATION AND *
* UNALLOCATION.                                               *
*********************************************************************

* S99 REQUEST BLOCK POINTER *

MY99RBPT DC A(EQUHOBON+MY99RB) S99 RB PTR

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

MY99RB DS CL(RBLEN) MY SVC 99 RB

RBLEN EQU (S99RBEND-S99RB) LENGTH OF RB FOR MY99RB

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

* TEXT UNIT POINTERS FOR ALLOCATION *

MY99TPTA DC A(TXTDALDS) TU FOR DATASET NAME

DC A(TXTSSREQ) NAME OF SUBSYSTEM TU PTR

DC A(EQUHOBON+TXTRTDDN) RETURN DD NAME TU

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

* TEXT UNIT POINTERS FOR UNALLOCATION *

MY99TPTU DC A(TXTDUNDD) TU FOR UNALLOC BY DNAME

DC A(TXTDUNNH) NOHOLD TU

DC A(TXTDUNCL) CHANGE THE CLASS TU

DC A(EQUHOBON+TXTDUNDS) CHANGE THE DEST TU

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

* TEXT UNITS FOR ALLOCATION *

TXTDALDS DC AL2(DALDSNAM) DATASET NAME KEY

DC X'0001' NUMBER

DC AL2(44) DSNAMES LENGTH

TXTDSNAM DS CLA4' ' DSNAMES FROM IEFSSREQ

TXTCLOSE DC AL2(DALCLOSE) UNALLOCATE AT CLOSE KEY

DC X'0000' # FIELD (0000 REQUIRED)

TXTSSREQ DC AL2(DALSSREQ) REQUEST OF SUBSYSTEM

DC X'0001' # FIELD (0001 REQUIRED)

DC X'0004' LEN OF SS NAME FOLLOWING

DC CLA4'JES3' NAME OF SUBSYSTEM

TXTRTDDN DC AL2(DALRTDDN) RETURN DDNAME FIELD
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
SSI Function Code 11

- IEFSSUS

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key.</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 SSOB, SSIB, and SSUS control blocks can reside in storage above or below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 5 shows the environment at the time of the call for SSI function code 11.

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
### SSI Function Code 11

- **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 (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 11 (SSOBUSER)</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 “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 (SSUS 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 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>
</table>
SSI Function Code 11

**SSUSLEN**
- Length of the SSUS (SSUSIZE) control block

**SSUSFLG1**
- Flag Byte

**SSUS1NOD**
- Return the node name

**SSUSCVXE**
- Destination conversion extension exists

**SSUSVER**
- Version of mapping for the caller – Set this field to SSUSCVER (an IBM-defined integer constant within the SSUS control block).

**SSUSUSER**
- Remote destination to be verified.

**SSUSFLG2**
- Conversion flag byte

**SSUS1TO2**
- Convert one 18 byte field to two 8 byte fields. The input field is SSUSDEST and the output fields are SSUSDST1 and SSUSDST2.

**SSUS2TO1**
- Convert two 8 byte fields to one 18 byte field. The input fields are SSUSDST1 and SSUSDST2 and the output field is SSUSDEST.

**SSUSITOC (JES2 only)**
- 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.

**SSUSCTOI (JES2 only)**
- Convert an 18 byte field to a 4 byte internal destination with a 8 byte user destination. The input field is SSUSDEST and the output fields are SSUSIDST and SSUSUDST.

**SSUSGENC (JES2 only)**
- 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.
SSI Function Code 11

**SSUSDSDLN**
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 (Decimal)</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>
<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
- 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>SSUSRTO (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 or conversion.</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.

**Note:**
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 the methods described in "Defining Your Subsystem" on page 464.

For more information, see "Maintaining Information About the Callers of Your Subsystem" on page 483.
SSI Function Code 15

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 483 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
- IEFSSIB
- IEFSSVS

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</td>
</tr>
<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</td>
</tr>
<tr>
<td></td>
<td>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 on page 46 shows the environment at the time of the 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>
</table>
SSI Function Code 15

SSOBID
Identifier 'SSOB'

SSOBLLEN
Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC
SSI function code 15 (SSOBVERS)

SSOBSSIB
Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See "Subsystem Identification Block (SSIB)" on page 8 for more information on the life-of-job SSIB.

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

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

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

Field Name
Description

SSIBID
Identifier 'SSIB'

SSIBLEN
Length of the SSIB (SSIBSIZE) control block

SSIBSSNM
Subsystem name — name of the subsystem that this verify subsystem function call is directed to (MSTR).

SSIBJBID
Name of the subsystem to be verified

Note: This is an 8-character field. Because subsystem names can only be 1-4 characters, the subsystem name specified should be left-justified and padded to the right with blank (X'40') characters.

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

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

Field Name
Description

SSVSLEN
Length of the SSVS (SSVSSIZE) control block

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

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

Chapter 3. SSI Function Codes Your Program Can Request 47
SSI Function Code 15

Register

<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2—13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</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 Verify Subsystem function call completed. Check the SSOBRETN field for specific function information.

SSRTNSUP (4)
The subsystem specified in the SSIBSSNM field does not support the Verify Subsystem function call.

SSRTNTUP (8)
The subsystem specified in the SSIBSSNM field exists, but is not active.

SSRTOSS (12)
The subsystem specified in the SSIBSSNM field is not defined to MVS.

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

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

SSRTNSSI (24)
The SSI has not been initialized.

Output Parameters
Output parameters for the function routine are:
• SSOBRETN
• SSVS

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

Value (Decimal)

Meaning

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

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

Field Name Description

SSVSSCTP  Pointer to the subsystem's SSCVT.

SSVSNUM  The subsystem affinity index value that you can use in a SSAFF macro request. See “Maintaining Information About the Callers of Your Subsystem” on page 483 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:
SSI Function Code 20

- 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>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Supervisor state</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 SSOB, SSIB, and SSRR control blocks can reside in storage above 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
</tbody>
</table>
SSI Function Code 20

SSOBLEN
Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC
SSI function code 20 (SSOBREQST)

SSOBSSIB
Address of an SSIB control block

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

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

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

<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>SSIBSSNAM</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.
SSI Function Code 20

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 SSRCVER (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 (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
</table>
SSI Function Code 20

SSRTOK (0)
The Request Job ID call completed. Check the SSOBRETN field for specific function information.

SSRTNSUP (4)
The subsystem specified in the SSIBSSNM field does not support this function.

SSRTNTUP (8)
The subsystem specified in the SSIBSSNM field exists, but is not active.

SSRTNOSS (12)
The subsystem specified in the SSIBSSNM field is not defined to MVS.

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

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

SSRTNSSI(24)
The SSI has not been initialized.

Output Parameters
Output parameters for the function routine are:

- SSIB
- SSOBRETN
- SSRRJCRP

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

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

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

Value (Decimal)
Meaning

SSRROK (0)
Successful completion. JES assigned a job ID to the caller. The job ID is available in the SSIBJBID field. See “Restrictions” on page 55 for information on the processing that takes place after successful completion has been obtained.

SSRRFAIL (4)
The Request Job ID call did not successfully complete.
This can happen if JES is in the process of ending, and therefore cannot return job IDs.
This caller cannot make use of subsequent JES services.

**SSRRFREQ (8)**
The Request Job ID call is already known to this JES, and may not have a second job ID established.

**SSRRNOEC (16)**
For JES2 only, an ECB was not supplied through the SSRRSECB pointer field on the Request Job ID call.

**SSRRPRME (20)**
There is an error in the SSRR data area. For example, both the SSRRJOBL bit and the SSRRNJBL bits may be set.

**SSRRPERR (36)**
The JES processing this call has returned a program error. This can happen if the JES does not have enough virtual storage available to create either the job structure or other control blocks for the requesting address space.

**SSRRJCRP Contents:** The SSRJCRP field is a 4-byte pointer to the job correlator for the associated job. The job correlator is stored in the Subsystem Job Block Extension (SJXB) in field SJXJCOR.

**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
SSI Function Code 20

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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Supervisor state</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>
</tbody>
</table>
SSI Function Code 21

Caller variable  Caller value
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.

Register 1

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

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

Input Parameters
Input parameters for the function routine are:
- SSOB
- SSIB
- SSRR

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

Field Name  Description
SSI Function Code 21

SSOBID
Identifier 'SSOB'

SSOBLEN
Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC
SSI function code 21 (SSOBRTN)

SSOBSSIB
Address of an SSIB control block

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

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

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

Field Name
Description

SSIBID
Identifier 'SSIB'

SSIBLEN
Length of the SSIB (SSIBSIZE) control block

SSIBSSNM
Subsystem name — name of the subsystem to which this Return Job ID
call is directed.
This name identifies either the primary subsystem, or in the case of JES2, a
secondary JES subsystem.
You must use the same subsystem name in this SSIBSSNM field as you
used for the original Request Job ID call (SSI function code 20).

SSIBJBID
Returned job ID
You must use the job ID obtained during the previously issued Request Job
ID call (SSI function code 20).

SSIBSUSE
(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon
completion of the Request Job ID call (SSI function code 20).

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

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

Field Name
Description

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

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

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

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

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

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

Output Parameters
Output parameters for the function routine are:
- SSIB
- SSOBRETN
SSI Function Code 21

SSI 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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRROK (0)</td>
<td>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.</td>
</tr>
<tr>
<td>SSRRFRET (12)</td>
<td>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.</td>
</tr>
<tr>
<td>SSRRPERR (36)</td>
<td>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.</td>
</tr>
</tbody>
</table>

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
SSI Function Code 54

- 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 66.
  - 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 68.
  - 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 68.

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
SSI Function Code 54

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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>
</tbody>
</table>

Control Parameters: The SSOB, SSIB, and SSVI control blocks can reside in storage above 16 megabytes.

Recovery: The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Assembler Services Guide](https://www.ibm.com/support/docview/cnshb000000000190287) for more information on an ESTAE-type recovery environment.

Figure 9 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:

- **Register 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.
- **Register 13**: Address of a standard 18-word save area.

**Input Parameters**

Input parameters for the function routine are:

- SSOB
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 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 “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 (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.
Field Name
Description

SSVILEN
Length of entire area — Set this field to a value that is at least equal to the
value of SSVIMSZ (a constant contained within the SSVI control block).
The length includes the fixed header section, plus the system variable
section and the installation variable section. The caller must ensure that the
length specified in the SSVILEN field is large enough to contain the
requested information.

SSVIVER
Version of mapping for the caller — Set this field to SSVICVER (an
IBM-defined integer constant within the SSVI control block).

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

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

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

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

Return Code (Decimal)
Meaning

SSRTOK (0)  Successful completion. The subsystem request completed. Check field
SSOBRET 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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSVIOK (0)</strong></td>
<td>Successful completion. The requested information was returned. See the SSVI control block section description below for the specific format of the returned information.</td>
</tr>
<tr>
<td><strong>SSVINSTR (8)</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
| **SSVIPARM (16)** | The SSVI data area contains one or more of the following parameter errors:  
  - **SSOBINDV** (in the SSOB control block) did not contain the address of a valid SSVI control block  
  - **SSVIID** did not contain 'SSVI'  
  - **SSVIVER** did not specify a valid version of the SSVI control block  
  - **SSVILEN** contained a value that is less than the value of SSVIMSIZ (an IBM-defined integer constant within the SSVI control block). When you receive this return code, fix the problem and resubmit the request. |
| **SSVIABLG (24)** | An abend or logical error was encountered within the called subsystem’s function code routine. When you receive this return code, search the problem report databases for a fix to the problem. If no fix exists, contact the IBM support center. |
SSI Function Code 54

SSI Contents: The output information returned in the SSVI control block is contained in one or more of the following areas mapped within the SSVI control block:
- Fixed header output section
- System variable output section
- Installation variable output section

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

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

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

Field Name  
Description

SSVIRLEN  
A 2-byte binary field that contains either the length of the storage used (if the caller's request was successful), or the length of storage required (if the request failed because the caller did not specify enough storage).

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

Decimal Value  
Meaning

SSVIOK (0)  
Request was successful. The SSVIRLEN field contains the length, in bytes, of the returned data.

SSVINSTR (8)  
Request failed. The caller did not specify enough storage in the SSVIRLEN field. The SSVIRLEN field contains the amount of storage, in bytes, the subsystem requires to return the requested information.

Note that this field is not set when the SSOBRETN field contains return code SSVIPARM (16) or SSVIABLG (24).

SSVIRVER  
A 1-byte binary field that contains the version of the SSVI control block used by the subsystem. When the caller's version of the SSVI control block does not match the version used by the called subsystem, the subsystem returns information based on the older of the two versions of the SSVI control block.

SSVIFLEN  
A 2-byte integer field that contains the length of the fixed header output section of the SSVI control block the subsystem uses.

SSVIASID  
A 2-byte binary field that indicates the ASID of the subsystem. A value of
SSI Function Code 54

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 2.10, z/OS 1.4, or z/OS 1.9. The master subsystem returns the same value as that contained in CVTPRODN.

**SSVIFMID**
An 8-byte character field that specifies the FMID of the subsystem (for example, HBB5510, HJE5510, 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 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 '7' 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. The JES2 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. The JES3 service level can increase by at least one each time significant maintenance or function is added and is maintained across new releases. For additional information concerning this field, see [z/OS JES2 Installation Exits](#) or [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.

**SSVIUOFO**
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.
SSI Function Code 54

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:

Field Name | Description
---|---
SSVIVLEN | A 2-byte signed hexadecimal field that contains the length of the variable output data string. The length of this field is not included in the length of the string.

SSVIDAT | A variable length character string (its length is set through SSVIVLEN) that contains a set of keywords and their respective values. When master (MSTR) or JES is the called subsystem, any, all or none of the keyword values shown in Table 1 on page 69 are returned to the SSI code 54 caller.

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

IBM-Defined Keywords: The following table shows the IBM-defined keywords that can be returned in the variable-length character string:
**Table 1. IBM-Defined Keywords**

<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>GLOBALRELEASE='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>MULT_CHAR_JOBCLASS=YES</td>
<td>NO</td>
</tr>
<tr>
<td>MULTIPLE_STCTSO='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>
<tr>
<td>SAPI_IP_SELECT='NO'</td>
<td>Indicates selection by IP address (Internet protocol) not supported.</td>
</tr>
<tr>
<td>Keyword</td>
<td>Explanation</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</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 uppercase.

**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.

**Example Code**

```plaintext
SSIREQ54 TITLE '- ISSUE SUBSYSTEM INFORMATION SSI CALL'
SSIREQ54 AMODE 31
SSIREQ54 RMODE ANY
  SPLEVEL SET=4
*********************************************************************
* FUNCTION: THIS PROGRAM GENERATES A SUBSYSTEM VERSION INFORMATION *
* CALL. IT DISPLAYS THE RETURNED INFORMATION ON THE               *
* ON THE OPERATOR CONSOLE. THE SUBSYSTEM CALL IS                    *
```

Chapter 3. SSI Function Codes Your Program Can Request  71
SSI Function Code 54

* DIRECTED TO THE MASTER SUBSYSTEM. *

* NAME OF MODULE: SSIREQ54 *

* REGISTER USE: *

* 0 PARM REGISTER *
* 1 PARM REGISTER *
* 2 SSOB *
* 3 SSIB *
* 4 SSVI *
* 5 SSVI SIZE USED *
* 6 SSVI SIZE NEEDED *
* 7 UNUSED *
* 8 ABEND VALUE REGISTER *
* 9 IEFSSREQ/SSVI RETURN CODES *
* 10 UNUSED *
* 11 UNUSED *
* 12 SSIREQ54 BASE REGISTER *
* 13 SAVE AREA CHAIN REGISTER *
* 14 PARM REGISTER / RETURN ADDR *
* 15 PARM REGISTER / COND CODE *

* ATTRIBUTES: PROBLEM STATE, AMODE(31), RMODE(ANY) *

* NOTE: THIS IS A SAMPLE PROGRAM. *

*********************************************************************
SPACE ,
SSIREQ54 START 0
TITLE ' - EQUATES'
*********************************************************************
* GENERAL EQUATES *
*********************************************************************
NOP EQU 0 NO OPERATION
FF EQU X'FF' ALL BITS ON
EQUHOBON EQU X'80000000' HIGH ORDER BIT ON
*********************************************************************
* AFTERCOMPARE 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       ALL OFF+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

**SSOBGN** EQU *       REQUIRED IF NOT USING IEFJSSOB DEFN
**TITLE** ' - LDA - LOCAL DATA AREA DSECT'  
**LDAA*   DS 9D       SAVE AREA FOR LOWER CALLERS
**LDA1O   DS CL8'LDAAREA '   IDENTIFICATION OF LDA AREA
**LDA@5S5OB DS F       POINTER TO SSOB FOR IEFSSREQ'S USE
**LDASSOB DC XL(SSOBHSIZ)'00'    AREA FOR SSOB
**LDASSIB DC XL(SSIBSIZE)'00'    AND SSIB
**LDAEND EQU *       START OF LOCAL DATA AREA
**LDASIZE EQU **   LENGTH OF AREA TO GETMAIN
**TITLE** '- HOUSEKEEPING REENTRANT ENTRY ROUTINE'

**SSIREQ54** CSECT
**BAKR** R14,0       SAVE CALLER'S ARS, GPRS, AND
**LR** R12,R15       SET UP PROGRAM BASE REGISTER
**USING SSIREQ54,R12  INFORM ASSEMBLER

---

Chapter 3. SSI Function Codes Your Program Can Request 73
**SSI Function Code 54**

```assembly
STORAGE OBTAIN, GET A SAVE AREA THAT'S REENTRANT
LENGTH=LDASIZE, STANDARD SAVE AREA SIZE
COND=NO UNCONDITIONAL REQ - NO RC INFO
SPACE
LR R13,R1 SAVE STORAGE ADDRESS
USING LDASTART.R13 ADDRESS LOCAL DATA AREA (LDA)
MVC LDAID,='LDAAREA' INDICATION OF LOCAL DATA AREA
WTO 'SSIREQ54 EXECUTING V1', LET OP KNOW
ROUTDCE=(2,11)
TITLE '- SS/SO/SSVI PROCESSING ROUTINE'
*********************************************************************
* SET UP SS/SO, SSIB, AND SSVI CONTROL BLOCKS. *
*********************************************************************
SPACE
*********************************************************************
* OBTAIN STORAGE FOR AN SSVI. *
*********************************************************************
LA R5,SSVIMSIZ MINIMUM SIZE REQUIRED
TRYIT DS 0H
STORAGE OBTAIN, GET A SAVE AREA THAT'S REENTRANT
LENGTH=(5), STANDARD SAVE AREA SIZE
COND=NO UNCONDITIONAL REQ - NO RC INFO
LR R4,R1 POINT TO THE SSVI
USING SSVI,R4 ADDRESSABILITY
SPACE
*********************************************************************
* WHEN ISSUING THE IEFSSREQ MACRO, REGISTER 1 MUST POINT TO *
* A CONTROL BLOCK THAT HAS IT'S HIGH-ORDER BIT SET, AND IT'S *
* LOW-ORDER 31 BITS POINTING TO THE SS/SO FOR THE SPECIFIC *
* FUNCTION CALL. THEREFORE, SET THIS CONTROL BLOCK *
* (LDASSOB) WITH THE HIGH ORDER BIT SET, AND THE LOW-ORDER *
* 31 Bits POINTING TO LDASSOB FIELD. *
*********************************************************************
SPACE
LA R2,LDASSOB POINT TO THE SS/SO
USING SS/SO,R2 ADDRESSABILITY
O R2,=A(EQUHOBON) SET HIGH ORDER BIT ON
ST R2,LDASSOB STORE FOR IEFSSREQ'S USE
* LATER WHEN ISSUING MACRO
*********************************************************************
* NOW PROCESS THE SS/SO (THE SUBSYSTEM OPTION BLOCK). *
*********************************************************************
SPACE,
XC SSOBEGIN(SSOBSHSIZ),SSOBEGIN CLEAR THE SS/SO
MVC SSOBID,='SSOBD' MOVE IDENTIFIER IN
MVC SSOBLEN,=Y(SSOBSHSIZ) MOVE SIZE OF THE HEADER IN
LA R1,LDASSIB POINT TO THE SSIB
ST R1,SSOBSSIB SAVE IN SSOB
MVC SSOFUNC,=Y(SSOBSVI) MOVE THE FUNCTION ID IN
ST R4,SSOBINDV SAVE SSVI ADDRESS IN SSOB
*********************************************************************
* DONE WITH THE SS/SO - 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 SSIBBEGIN(SSIBSIZE),SSIBBEGIN CLEAR SSIB
MVC SSIBID,='SSIBID' MOVE IDENTIFIER IN
MVC SSIBLEN,=Y(SSIBSIZE) MOVE SIZE OF THE SSIB IN
MVC SSIBSSNM,='MSTR' SHOW MASTER SUBSYSTEM TO BE
* USED TO GET THE INFO
```
Chapter 3. SSI Function Codes Your Program Can Request
SSI Function Code 54

```
LA      R7,SSV(R6)    R7==>USER VARIABLE DATA AREA
USING   SSVIVDAT,R7 ADDRESSABILITY
LH      R8,SSVIVLEN  GET THE LENGTH
CH      R8,=H'125'  GREATER THAN 125 CHARS?
BC      LE,SHOWIT1  NO, USE THE REAL LENGTH
MVC     SSVIVLEN,+H'125' ELSE, USE ONLY FIRST 125
SHOWIT1 DS 0H  R8=NUMBER OF CHARs TO DISPLAY
WTO     TEXT=SSVIVLEN, SHOW TO THE CONSOLE
        ROUTCDE=(11)
B       SHOWSYS2 BRANCH AROUND WTO
SHOWSYS DS 0H
WTO     'SSIREQ54 NO USER DATA PRESENT', LET OP KNOW
        ROUTCDE=(2,11)
SHOWSYS2 DS 0H
ICM     R6,B'1111',SSVISDOF ANY SYSTEM DATA?
BC      NZERO,SHOWSYS3 YES, DISPLAY IT
WTO     'SSIREQ54 NO SYSTEM DATA', LET OP KNOW
        ROUTCDE=(2,11)
B       RETURN
SHOWSYS3 DS 0H
LA      R7,SSV(R6)    R7==>USER VARIABLE DATA AREA
USING   SSVIVDAT,R7 ADDRESSABILITY
LH      R8,SSVIVLEN  GET THE LENGTH
CH      R8,=H'125'  GREATER THAN 125 CHARS?
BC      LE,SHOWIT2  NO, USE THE REAL LENGTH
MVC     SSVIVLEN,+H'125' ELSE, USE ONLY FIRST 125
SHOWIT2 DS 0H  R8=NUMBER OF CHARs TO DISPLAY
WTO     TEXT=SSVIVLEN, SHOW TO THE CONSOLE
        ROUTCDE=(11)
SPACE   ,
WTO     'SSIREQ54 RETURNING', LET OP KNOW
        ROUTCDE=(2,11)
SPACE   ,
*********************************************************************
* GIVE BACK THE STORAGE WE BOUGHT EARLIER.  *
*********************************************************************
SPACE   ,
RETURN  DS 0H
STORAGE RELEASE, FREE MY INFO AREA
LENGTH=(5), VARIABLY OBTAINED SIZE
ADDR=(4) HERE'S WHERE IT LIVES
STORAGE RELEASE, FREE MY REENTRANT SAVE AREA
LENGTH=LDASIZE, STANDARD SAVE AREA SIZE
ADDR=(R13) HERE'S WHERE IT LIVES
SPACE   ,
*********************************************************************
* SET PROGRAM RETURN CODE.  *
*********************************************************************
SPACE   ,
SLR     R15,R15 SET RETURN CODE OF ZERO
*********************************************************************
* RETURN TO CALLER WITH ORIGINAL STATUS AND REGISTERS.  *
*********************************************************************
SPACE   ,
PR      RETURN TO CALLER USING STACK, X
RESET REGS 2-14, ADDRESSING MODE, X
ASC MODE, AND RETURN TO CALLER
*********************************************************************
* ABEND ROUTINES FOLLOW  *
*********************************************************************
SPACE   ,
ABEND   DS 0H  R15 NON-ZERO AFTER IEFSSREQ
WTO     'PROGRAM HAD FATAL ERROR - SEE REGS 8 AND 9'
        ROUTCDE=(2,11)
SPACE   ,
ABEND (R8),DUMP,STEP LET THE USER IN ON THE BAD NEWS
```
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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</td>
</tr>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
</tbody>
</table>
### SSI Function Code 70

**Caller variable**
- Caller value

**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](https://www.ibm.com/support/knowledgecenter/SSEU75_2.2.1/COM.2.2.1.91.pdf) 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.

### Input Register Information

**Register 1**
- **1**

**SSOB**
- 'SSOB' (SSOBDID)
- Length (SSOBLEN)
- Function ID (SSOBFUNC)
- SSIB (SSOBSSIB)
- Function Dependent Area (SSOBDLA)

**SSIB**
- 'SSIB' (SSIBID)
- Length (SSIBLEN)
- Subsystem Name (SSIBSSNM)
- Return Job ID (SSIBJBID)
- Subsystem Use (SSIBSUSE)

**SSSF**
- Eyecatcher (SSSFID)
- SSSF
  - Length (SSSFLEN)
  - Version (SSSFVER)
  - Reason Code (SSSFRAS)
  - Function (SSSFREQ)
  - Flag Byte (SSSFGLG1)

**Figure 10. Environment at Time of Call for SSI Function Code 70**

### Input Register Information

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

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

### Input Parameters

Input parameters for the function routine are:
- SSOB
- SSIB
- 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>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 70 (SSOBSFS)</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 (SSSF 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>SSISSN</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 SSISSN 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, SSSFHSIZE, and the size of the request dependent area. For the</td>
</tr>
</tbody>
</table>
SSI Function Code 70

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.

**SSFVER**
Version of mapping for the caller – Set this field to SSSFCVER (an IBM-defined integer constant within the SSSF control block).

**SSSFREQF**
Function request value – designates the function type requested. Possible values are:

- **SSSFWSWBM**
  SWB modify of output descriptors

- **SSSFWSWBF**
  SWB merge of output descriptors

- **SSSFWSWBC**
  Return storage used for SWB merge function. Use this 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.

**SSSSFGLG1**
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 ensure 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.
SSI Function Code 70

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 SSSFLG1 is ON.

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: SSSFERAD 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 SSSFERAD and SSSFERLN must be non-zero). The user can choose to fill in pair 1, pair 2, or both pairs.
SSI Function Code 70

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

**SSSFERAD**
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: SSSFERAD 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 SSSFERAD 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 ensures that the user has SAF authority to the ISFAUTH resource (SDSF class).

**SSSFSECL**
Perform a SECLABEL dominance check to ensure 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 ensure 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.

**SSSFCNID**
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).
SSI Function Code 70

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.

**SSSFFGTK**
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:

<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 scheduler facilities call 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>
<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>
</tbody>
</table>
SSI Function Code 70

SSRTNSSI(24)
   The SSI has not been initialized.

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

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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSFOK (0)</td>
<td>Request processed successfully.</td>
</tr>
<tr>
<td>SSSFUERR (8)</td>
<td>Request rejected, see reason code in field SSSFREAS.</td>
</tr>
<tr>
<td>SSSFEXTE (12)</td>
<td>No extension found.</td>
</tr>
<tr>
<td>SSSFNOST (16)</td>
<td>No storage to process request.</td>
</tr>
</tbody>
</table>

SSSFREAS Contents: When control returns to the caller 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>SSSFINVSE (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>Error processing request. See reason code in field SSSFIMREA for SWB modify function or SSSFIMREA for SWB merge/SWB merge cleanup functions.</td>
</tr>
<tr>
<td>SSSFEXEC (40)</td>
<td>Exit cancelled request.</td>
</tr>
<tr>
<td>SSSFDISA (44)</td>
<td>Scheduler Services disabled.</td>
</tr>
<tr>
<td>SSSFGLBL (48)</td>
<td>Insufficient Global Level</td>
</tr>
</tbody>
</table>
### SSI Function Code 70

#### SSSFMREA 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 (SSSFFLG1).</td>
</tr>
<tr>
<td>SSSFMIVX (24)</td>
<td>Invalid modify extension.</td>
</tr>
<tr>
<td>SSSFMTKE (28)</td>
<td>Modify data set token error.</td>
</tr>
<tr>
<td>SSSFMNTK (32)</td>
<td>No data set token provided.</td>
</tr>
<tr>
<td>SSSFMJNF (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>SSSFMSPC (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>SSSFMSPPL (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>
**SSI Function Code 70**

**SSSFFREA Contents:** When control returns to the caller of SWB merge/SWB merge cleanup function and the field SSSFREAS = SSSFINRI (36), the SSSFFREA field contains one of the following decimal values:

- **SSSFFOK (0)**
  - Check the SSSFWRTN and SSSFWRSN fields for information.

- **SSSFFDST (4)**
  - Data set token not given

- **SSSFFDSG (8)**
  - Data set gone

- **SSSFFDSV (12)**
  - Failure obtaining checkpoint version (JES2 only)

- **SSSFFJBG (16)**
  - Job gone

- **SSSFFSWI (20)**
  - Invalid SWBTU buffer

- **SSSFFDSE (24)**
  - Invalid data set token

- **SSSFFGTE (28)**
  - Invalid group token

- **SSSFFNOS (32)**
  - No storage to process request

- **SSSFFSPL (36)**
  - Spool I/O error

- **SSSFFTNU (40)**
  - Token not usable for requested function

- **SSSFFDSQ (48)**
  - Data set on BDT/TCP queue

- **SSSFFDSF (52)**
  - Data set referenced by JECL FORMAT statement

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 = SWBTREQ RETRIEVE
  28 = SWBTREQ SPLICE
  32 = SWBTREQ SPLIT

SSSRFLG
Returned flag byte.

SSSSFIPA
IP address available in SWBTU buffer.

JES Job Information Services— SSI Function Code 71
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” on page 88</td>
<td>SSJISIOM/SSJISIRS</td>
<td>IAZSLPIO</td>
</tr>
<tr>
<td>“JES2 Health monitor information” on page 96</td>
<td>SSJIMNOD/SSJIMNRS</td>
<td>IAZMOND</td>
</tr>
<tr>
<td>“Job Class Information” on page 110</td>
<td>SSJIFJCO/SSJIFJCR</td>
<td>IAZJCLD</td>
</tr>
<tr>
<td>“Convert Device ID Service” on page 122</td>
<td>SSJICVDV</td>
<td>IAZCVDEV</td>
</tr>
<tr>
<td>“Checkpoint Version Information Service” on page 128</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.
**SSI Function Code 71**

**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). SSJISIOM 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 insures 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** on page 89
- **Issued to** on page 89
- **Related SSI Codes** on page 89
- **Related Concepts** on page 89
- **Environment** on page 89
- **Input Register Information** on page 90
- **Input parameters** on page 90
- **Output Register Information** on page 93
- **Return Code Information** on page 93
- **Output Parameters** on page 94

**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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJI, and IAZSPLIO, control blocks can reside in 24- or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>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</td>
</tr>
</tbody>
</table>

Figure 11 on page 90 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:

<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
- IAZSPLIO (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 (SSOBSIZE) control block</td>
</tr>
</tbody>
</table>

Figure 11. Environment at Time of Call for SSI Function Code 71, Spool Read Subfunction
### SSI Function Code 71

**SSOBFUNC**
SSI function code 71(SSOBSSJI)

**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>
</tbody>
</table>

**SSJIFREQ**
Function to be performed on this request. Valid functions and their related SSJIUSER area are:

<table>
<thead>
<tr>
<th>Field Value</th>
<th>SSJIUSER Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJISIOM</td>
<td>IAZSPLIO SPOOL read service, read record from SPOOL</td>
</tr>
<tr>
<td>SSJISIRS</td>
<td>IAZSPLIO SPOOL read service, release storage</td>
</tr>
</tbody>
</table>
SSI Function Code 71

SSJIUSER
Point 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>
</tbody>
</table>
| SPIOSPAD     | SPOOL address token of the record to be read. For JES2, the token value can be any of the following:  
  • A SPOOL address token returned by extended status: for example, STJ2SPOL, STS2SPOL or STO2SPST;  
  • An MTTR placed in the first four bytes, and the next four bytes set to zero;  
  • The first byte set to X'FF', the second byte set to zero (0), and the remaining 6 bytes an MQTR. |
| SPIOCTYP     | 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:  
<table>
<thead>
<tr>
<th>Field Value</th>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHK</td>
<td>$CHK</td>
<td>Printer check record</td>
</tr>
<tr>
<td>HDB</td>
<td>$BUFFER</td>
<td>SYSIN/SYSOUT data buffer</td>
</tr>
<tr>
<td>IOT</td>
<td>$IOT</td>
<td>Data set information blocks. Contains the PDDBs</td>
</tr>
<tr>
<td>JCT</td>
<td>$JCT</td>
<td>JES2 job control table. Main job control block</td>
</tr>
<tr>
<td>NHSB</td>
<td>$NHSB</td>
<td>NJE headers and trailers</td>
</tr>
<tr>
<td>OCT</td>
<td>$OCT</td>
<td>/* OUTPUT JECL card descriptors</td>
</tr>
<tr>
<td>SIG</td>
<td>none</td>
<td>Signature record (record 0)</td>
</tr>
<tr>
<td>SWBI</td>
<td>$SWBIT</td>
<td>SYSOUT SWB information</td>
</tr>
<tr>
<td>SPIOJNAM</td>
<td>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’.</td>
<td></td>
</tr>
</tbody>
</table>
SSI Function Code 71

**SPIOJID**
Optional input that requests that the JOBID in the data area read matches the value specified. Only 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.

**Return Code (Decimal)**

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

Chapter 3. SSI Function Codes Your Program Can Request 93
SSI Function Code 71

**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)

**SSOBTRETN 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>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>Error in the parameter list, i.e., the SSJ extension is an invalid format - it is not an SSJ, the service version number is not supported, or the SSJ 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 = SSJIERRU (8):
SSI Function Code 71

Value (Decimal)
Meaning

SSJIUNSF (4)
Unsupported subfunction requested.

SSJINTDS (24)
SSJIUSER does not point to the correct control block.

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:

Value (Decimal)
Meaning

SPIOOK (0)
Success

SPIONTVF (4)
Control block verification failed

SPIOCBIO (8)
SPOOL control block I/O error

SPIOCBTK (12)
SPOOL control block track address

SPIOCBNG (16)
General control block problem

SPIOSTRG (20)
Error obtaining 31-bit storage

SPIOSJER (24)
Error obtaining 24-bit storage

SPIOILLOG (28)
A logic error has occurred

SPIONSPL (32)
SPIOSTRP not initialized correctly

SPIONBUF (36)
Could not locate instorage buffer. Most likely, the buffer has been written to SPOOL and is no longer in memory.

SPIONSAF (40)
Authorization (SAF) failure accessing data.

SPOOL read service, IAZSPLIO contents: For the SPOOL read service (function code SSJISIOM) the following is returned in IAZSPLIO:

Field Name
Description

SPIOVERO
Subsystem version number (currently 1)
SSI Function Code 71

SPIOOOUTA
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.

SPIOOLEN
Length of the data area returned (not including the prefix area).

SPIOND1
Indicator field.

Value (Decimal)
Meaning

SPIONSTG
The control block was retrieved from an instorage buffer.

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”
- “Use Information”
- “Issued to”
- “Related SSI Codes”
- “Related Concepts”
- “Environment”
- “Input Register Information” on page 98
- “Input Parameters” on page 98
- “Output Register Information” on page 101
- “Return Code Information” on page 101
- “Output Parameters” on page 102

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 SSJUSER.

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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJI, and IAZMOND control blocks can reside in 24- or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>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</td>
</tr>
</tbody>
</table>

Figure 12 on page 98 shows the environment at the time of the call for SSI function code 71, JES2 Health Monitor Information Subfunction.
Input Register Information: Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register

Contents

1

Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.

13

Address of a standard 18-word save area.

Input Parameters: Input parameters for the function routine are:

- SSOB
- SSIB
- IAZSSJI
- IAZMOND (JES2 Health Monitor Information)

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

 Field Name
 Description

SSOBDID
 Identifier 'SSOB'
SSI Function Code 71

SSOBLEN
   Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC
   SSI function code 71(SSOBSSJI)

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:

Field Name   Description
SSIBID       Identifier 'SSIB'

SSIBLEN     Length of the SSIB (SSIBSIZE) control block

SSIBSSNM    Subsystem name — name of the subsystem to which this Job Information
             Services request is directed

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:

Field Name   Description
SSJIID       Eyecatcher for the control block (set to 'SSJI')

SSJILEN     Length of the IAZSSJI (SSJISIZE) control block

SSJISVRN     Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of
             the control block

SSJIFREQ     Function to be performed on this request. Valid functions and their related
             SSJIUSER area are:

Field Value   SSJIUSER Description
SSJIMNOD     IAZMOND JES2 Health Monitor information, obtain data
SSI Function Code 71

SSI Function Code 71

SSJIMNRS
IAZMOND JES2 Health Monitor information, release storage

SSJIUSER
Pointer to service specific data area (IAZMOND)

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>
<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>MONDSMATS</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>MONDSJSN</td>
<td>JES2 Notices</td>
</tr>
<tr>
<td>MONDSJST</td>
<td>JES2 Tracks</td>
</tr>
</tbody>
</table>
SSI Function Code 71

MONDSSTO
JES2 storage usage statistics

MONDSEL2
Information selection flag byte 2

MONDSMNS
Monitor status information

MONDHSTC
History count limit.

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, them 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.

Return Code (Decimal)
Meaning
SSI Function Code 71

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
- 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:

Value (Decimal)
Meaning
SSJIOK (0) Request successful.
SSJIERVR (4) Request completed with possible errors, see SSJIRETN for reason code.
SSJIERRU (8) Request cannot be completed because of user error, see SSJIRETN for reason code.
SSJIERRJ (12) Request cannot be completed, SSJIRETN contains internal reason code.
SSJIPARM (16) The parameter list, ie, the SSJ extension is an invalid format - it is not an SSJ, the service version number is not supported, or the SSJ is not large enough.

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:
### 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>
<tr>
<td>MONDJDWN</td>
<td>JES2 is down</td>
</tr>
</tbody>
</table>
SSI Function Code 71

MONDMOWN
    Health Monitor is down

MONDRESQ
    Pointer to resource usage statistics (MDRSDATA)

MONDCPUS
    Pointer to main task CPU statistics (MDCPDATA)

MONDERRC
    Pointer to JES2 ERROR statistics (MDERDATA)

MONDWAIT
    Pointer to main task WAIT statistics (MDERDATA)

MONDMGS
    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:
Field Name    Description
MDRSEYE        Eyecatcher 'MDRS'
MDRSNEXT       Next MDRS entry
MDRSNAME       Resource name
MDRSENTO       Offset to first time entry
MDRSCNT        Number of time entries
MDRSENTL       Length of a time entry
MDRSFLG1       General flag byte
MDRS1OVR       Resource currently over warning level

The fields in the MDRSNTRY time entry are:
### SSI Function Code 71

<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>
<tr>
<td>MDRSWARN</td>
<td>Warn level (%)</td>
</tr>
<tr>
<td>MDRSOVER</td>
<td>Usage over warn level (% * 100)</td>
</tr>
</tbody>
</table>

**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>
</tbody>
</table>
SSI Function Code 71

MDCPLLOK
Local lock sample count

MDCPNDSP
Non Dispatchable count

MDCPPAGE
Page wait sample count

MDCPDMVS
Awaiting MVS dispatch count

**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 MDERENTRY 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 MDERENTRY 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>
<tr>
<td>MDERTYPE</td>
<td>Error category</td>
</tr>
<tr>
<td>MIDERMAIN</td>
<td>Main task</td>
</tr>
<tr>
<td>MDERDIST</td>
<td>DISTERR</td>
</tr>
<tr>
<td>MDERCBIO</td>
<td>CBIO error</td>
</tr>
<tr>
<td>MDEROTHER</td>
<td>Other</td>
</tr>
</tbody>
</table>
MDERSTSK
   JES2 Subtask

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>MDWTCTOR</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>
</tbody>
</table>

   NONE  Wait while JES2 was in control
         Exit #  Wait while this exit was in control
         MULTEXIT Multiple exits were in control
         MULTIPLE JES2 and exit code in control
SSI Function Code 71

MDWTFLAG
   Wait flag byte

MDWTFINI
   JES2 was initializing

MDWTFTRM
   JES2 was terminating

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:

Field Name      Description

MDMSEYE          Eyecatcher 'MDMS'

MDMSNEXT         Next MDMS entry

MDMSLEN          Length of entry

MDMSTIME         Time condition started (STCKE). Only for Alerts and Tracks.

MDMSTYPE         Message type

   MDMSTALR       Alert message

   MDMSTTRK       Track message

   MDMSTNOT       Notice message

MDMDL1LN         Length of first line of message

MDMDL1TX         Text of first line of message

MDMDL2LN         Length of second line of message

MDMDL2TX         Text of second line of message

MDMDL3LN         Length of third line of message

MDMDL3TX         Text of third line of message

MDMDL4LN         Length of fourth line of message

MDMDL4TX         Text of fourth line of message
**SSI Function Code 71**

**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 'MDMI'</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>MDMIINFO</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:

The fields in the MDSTNTRY time entry are:
### SSI Function Code 71

<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
- Input Register Information on page 112
- Output Register Information on page 114
- Return Code Information on page 114
- Output Parameters on page 114

**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 SSJUSER.

**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:
SSI Function Code 71

- IEFSSOBH
- IEFJSSIB
- IAZSSJI
- IAZJBCLD (Job Class Information)

The caller must meet the following requirements:

**Caller variable** | **Caller value**
---|---
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 for more information on an ESTAE-type recovery environment

Figure 13 shows the environment at the time of the call for SSI function code 71, Job Class Information Subfunction.

![Figure 13. Environment at Time of Call for SSI Function Code 71, Job Class Information Subfunction](image-url)
SSI Function Code 71

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

Register

Contents

1 Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.

13 Address of a standard 18-word save area.

Input Parameters: Input parameters for the function routine are:

- SSOB
- SSIB
- IAZSSJII
- IAZJBCLD (Job Class Information)

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

Field Name  
Description

SSOBID  
Identifier 'SSOB'

SSOBLLEN  
Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC  
SSI function code 71(SSOBSSJI)

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 (IAZSSJII control block)

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

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

Field Name  
Description

SSIBID  
Identifier 'SSIB'

SSIBLEN  
Length of the SSIB (SSIBSIZE) control block

SSIBSSNM  
Subsystem name — name of the subsystem to which this Job Information Services request is directed

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:

Field Name
- Description

SSJIID
- Eyecatcher for the control block (set to 'SSJI')

SSJILEN
- Length of the IAZSSJI (SSJISIZE) control block

SSJISVRN
- Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block

SSJIFREQ
- Function to be performed on this request. Valid functions and their related SSJIUSER area are:
  Field Value
  - Description
    SSJIFJCO
    - IAZJBCLD Job Class information service, obtain data
    SSJIFJCR
    - IAZJBCLD Job Class information service, return storage

SSJIUSER
- Pointer to service specific data area (IAZJBCLD)

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:

Field Name
- Description

JBCLSSID
- Eyecatcher of the control block (set to 'JBCL')

JBCLEN
- Length of the IAZJBCLD (JBCLSZE) control block

JBCLSVRN
- Input version of the IAZJBCLD control block. Set to JBCLSVR# for the current (latest) version.

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

JBCLIJOB
- Return a particular job class indicated by JBCLNAM
SSI Function Code 71

**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)**

<table>
<thead>
<tr>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
</tr>
<tr>
<td>The job information services request completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
</tr>
<tr>
<td>The subsystem specified in the SSIBSSNM field does not support the job information services function call.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
</tr>
<tr>
<td>The subsystem specified in the SSIBSSNM field exists but is not active.</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
</tr>
<tr>
<td>The subsystem specified in the SSIBSSNM field is not defined to MVS.</td>
</tr>
<tr>
<td>SSRTDIST (16)</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>Either the SSIB control block or the SSOB control block has incorrect lengths or formats.</td>
</tr>
<tr>
<td>SSRTNSSI(24)</td>
</tr>
<tr>
<td>The SSI has not been initialized.</td>
</tr>
</tbody>
</table>

**Output Parameters:** Output parameters for the function routine are:

- SSOBRETN
### SSI Function Code 71

- **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>SSJIERRV (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>SSJUSER does not point to the correct data area</td>
</tr>
<tr>
<td>SSJIUNSD (28)</td>
<td>SSJUSER control block version number is not correct</td>
</tr>
<tr>
<td>SSJISMDS (32)</td>
<td>SSJUSER 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 SSJIJCO), 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:
SSI Function Code 71

Field Name
Description

JBCLVERO
Subsystem version number (currently 4)

JBCLSMCL
STC message class

JBCLTMCL
TSU message class

JBCLDPTR
Pointer to first job class data buffer

JBCLNJJC
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):
Field Name
Description

JBCTEYE
Eyecatcher 'DCAT'

JBCTOHDR
Offset to first section

JBCTNEXT
Address of next CAT

JBCLDHSZ
Size of header

Class Attribute Table Prefix section (mapped by JBCTPREF DSECT):
Field Name
Description

JBCTHDLN
Length of entire job class entry (maximum value is 65535)

JBCTHDTDP
Type of this section

JBCTHDMMD
Modifier

JBCTHDSZ
Size of prefix section

Class Attribute Table Data section (mapped by JBCLDCAT DSECT):
Field Name
Description

JBCTLEN
Length of job class data section
SSI Function Code 71

JBCCTYPE
   Type of this section (JBCLTCAT)

JBCCTMOD
   Modifier

JBCJOBFL
   Job flags
   JBCBATCH
      Batch job
   JBCSUJB
      Time sharing user
   JBTCJOB
      System task

JBCVALJB
   Valid job types

JBCNOJNL
   No journal option

JBCNOUPT
   No output option

JBCSCAN
   TYPRUN=SCAN was specified

JBCCOPY
   TYPRUN=COPY was specified

JBCRSTRRT
   Allow warm start to re-que for execution

JBCJBOPT
   Job options flag
   JBCTHOLD
      TYPRUN=HOLD

JBCNOLOG
   No job log option

JBCXBMII
   XBM II job

JBCQHELD
   Class queue is held

JBCPROCN
   Procedure library number

JBCSMFLG
   SMF Flag
   JBCNOUSO
      Do not take IEFUSO exit

JBCNOTY6
   Do not produce Type 6 SMF record

JBCNOUJP
   Do not take IEFUJP exit
SSI Function Code 71

JBCNOT26
Do not produce Type 26 SMF record

JBCPERFM
Default performance group

JBCCPBGN
Beginning of converter parameters

JBCCACCT
Accounting information flag

JBCCNONE
No information is required

JBCCNAME
Programmer required

JBCCNUMB
Account number required

JBCCALL
Programmer and account number required

JBCCSWAL
SWA above 16M line

JBCCTIME
Default job step interval time

JBCCMNT
Maximum minutes

JBCCSECS
Maximum seconds

JBCCREGN
Default job step region

JBCCRGN
Numeric specification

JBCCRGA
Kilobytes or megabytes specification

JBCCMND
Command disposition

JBCCEXEC
Pass the command through

JBCCDSPL
Display and then pass command

JBCCOVER
Ask operator disposition

JBCCIGN
Ignore the command

JBCCBLP
Bypass label processing option

JBCCBLPY
Process bypass label parm
SSI Function Code 71

JBCCOCG
Operator command group

JBCCGSYS
Group 1 commands (SYS)

JBCCGIO
Group 2 commands (I/O)

JBCCGCON
Group 3 commands (CONS)

JBCCGALL
All command groups

JBCCLJCL
Default MSGLEVEL, JCL listed if not MSGLEVEL

JBCCTMSG
Allocation termination messages

JBCCONVP
End of converter parameters

JBCOPSWT
Converter option switches

JBCFLAG1
Normal OUTDISP for JESDS

JBC1CDP
Conditionally purge output for jobs in this class

JBC1NODP
NORMAL OUTDISP=PURGE

JBC1NODW
NORMAL OUTDISP=WRITE

JBC1NODH
NORMAL OUTDISP=HOLD

JBC1NODK
NORMAL OUTDISP=KEEP

JBC1NODL
NORMAL OUTDISP=LEAVE

JBCFLAG2
Abnormal OUTDISP for JESDS

JBC1AODP
ABNORMAL OUTDISP=PURGE

JBC1AODW
ABNORMAL OUTDISP=WRITE

JBC1AODH
ABNORMAL OUTDISP=HOLD

JBC1AODK
ABNORMAL OUTDISP=KEEP

JBC1AODL
ABNORMAL OUTDISP=LEAVE
SSI Function Code 71

JBCFLAG3
Processing flags

JBC3WLM
WLM managed class

JBC3SPEC
Special class (STC/TSU)

JBC3PSEU
Pseudo class (Only class name and counts valid)

JBC3SLINV
Default SCHENV (CATSCHED) no longer defined

JBC3DUOK
Duplicate job names OK for this job class

JBCXBM
PROCNAME for XBM II job

JBCCLASS
Job class

JCMAXJ
Maximum executing jobs in this class in the JESPLEX

JBCCURJ
Current executing job in this class in the JESPLEX

JBCQSIZE
Jobs eligible for execution (including executing jobs)

JBCHLDCT
Jobs held in class (not including executing jobs)

JBCSIZ1
Version 1 job class length

JBCDSCHE
Default SCHENV, Job classes only

JBCDMCLS
Default MSGCLASS, TSU and STC classes only

JBCSIZ2
Version 2 job class length

JBCJLOG
JESLOG control information

JBCFLG
Flags

JBJLELIG
Spin eligible

JBJLTIMI
Spin on time interval

JBJLTDIM
Spin on time of day

JBJLLINE
Spin upon line delta
SSI Function Code 71

**JBJLSUP**
Suppress

**JBJLNOSP**
No spin

**JBJSOURC**
Source of JESLOG info

**JBJSEXIT**
JESLOG from Exit

**JBJSJCL**
JESLOG from JCL

**JBJSCAT**
JESLOG from CAT

**JBJSSRR**
JESLOG from IEFSSRR

**JBCJLVAL**
Spin value

**JBCTSIZ3**
Version 3 job class length

**JBCTSIZ4**
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>JBCM_TYPE</td>
<td>Type of this section (JBCLTMEM)</td>
</tr>
<tr>
<td>JBCM_MOD</td>
<td>Modifier</td>
</tr>
<tr>
<td>JCMFRST</td>
<td>First member section offset</td>
</tr>
<tr>
<td>JCMCNT</td>
<td>Count of member entries</td>
</tr>
<tr>
<td>JCMMLALEN</td>
<td>Length of a member entry</td>
</tr>
<tr>
<td>JCM1ST</td>
<td>Beginning of first member entry</td>
</tr>
<tr>
<td>JCMSIZE</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>
</table>
SSI Function Code 71

JBCMNAME
  JES2 member name

JBCMSYS
  MVS system name

JBCMFLG1
  Member flags

JBCMJBA
  Jobclass active on member

JBC1ACT
  Member is active

JBC1PXQ
  $P XEQ issued on member

JBC1PJS
  Member is draining ($P)

JBCMJMAX
  Maximum jobs active

JBCMJACT
  Current jobs active

JBCMESIZ
  Member entry data section length

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” on page 123
- “Environment” on page 123
- “Input Register Information” on page 124
- “Input Parameters” on page 124
- “Output Register Information” on page 126
- “Return Code Information” on page 126
- “Output Parameters” on page 127

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 SSJUSER.

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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJI, and IAZCVDEV control blocks can reside in 24- or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>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</td>
</tr>
</tbody>
</table>

Figure 14 on page 124 shows the environment at the time of the call for SSI function code 71, Convert Device ID Service Subfunction.
SSI Function Code 71

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
- 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>
</tbody>
</table>
SSI Function Code 71

SSOBLLEN
Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC
SSI function code 71(SSOBSSJI)

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:

Field Name
Description

SSIBID
Identifier 'SSIB'

SSIBLEN
Length of the SSIB (SSIBSIZE) control block

SSIBSSNM
Subsystem name — name of the subsystem to which this Job Information Services request is directed

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:

Field Name
Description

SSJIID
Eyecatcher for the control block (set to 'SSJI')

SSJILEN
Length of the IAZSSJI (SSJISIZE) control block

SSJISVRN
Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block

SSJIFREQ
Function to be performed on this request. Valid functions and their related SSJIUSER area are:

Field Value
Description

SSJICVDV
IAZCVDEV, Convert device ID service
SSI Function Code 71

SSJIUSER
   Pointer to service specific data area (IAZCVDEV)

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:

Field Name | Description
---|---
CVDVSSID | Eyecatcher of the control block (set to 'CVDV')
CVDVLEN | Length of the IAZCVDEV (CVDSZE) control block
CVDEVVRN | 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
CVDVID | Device ID in binary

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:

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 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.
SSI Function Code 71

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
• 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>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, 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>
</tbody>
</table>
SSI Function Code 71

SSJISMDS (32)
SSJIUSER CB length is too small

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVDVVERO</td>
<td>Subsystem version number (currently 1)</td>
</tr>
<tr>
<td>CVDVNAME</td>
<td>Convert device name in EBCDIC. If the device type is not known, then this will be set to ‘UNKNOWN’</td>
</tr>
<tr>
<td>CVDSIZE</td>
<td>Size of IAZCVDEV</td>
</tr>
</tbody>
</table>

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” on page 129
- “Environment” on page 129
- “Input Register Information” on page 130
- “Input parameters” on page 130
- “Output Register Information” on page 132
- “Return Code Information” on page 132
- “Output Parameters” on page 133

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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Supervisor state, any PSW key</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=SASN=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, IAZSSJI, and IAZDSERV control blocks can reside in 24- or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>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</td>
</tr>
</tbody>
</table>

Figure 15 on page 130 shows the environment at the time of the call for SSI function code 71, Checkpoint Versions Subfunction.
SSI Function Code 71

**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>
</tbody>
</table>
SSI Function Code 71

SSOBLEN
Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC
SSI function code 71(SSOBSSJI)

SSOBSIB
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:

Field Name
Description

SSIBID
Identifier 'SSIB'

SSIBLEN
Length of the SSIB (SSIBSIZE) control block

SSIBSSNM
Subsystem name — name of the subsystem to which this Job Information
Services request is directed

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:

Field Name
Description

SSJIID
Eyecatcher for the control block (set to 'SSJI')

SSJILEN
Length of the IAZSSJI (SSJISIZE) control block

SSJISVRN
Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of
the control block

SSJIFREQ
Function to be performed on this request. Valid functions and their related
SSJIUSER area are:

Field Value
SSJIUSER Description

SSJIFOBT
IAZDSERV, Obtain Checkpoint versions information
SSI Function Code 71

**SSJIFREL**
IAZDSERV, Release Checkpoint versions information

**SSJIUSER**
Pointer to service specific data area (IAZDSERV)

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>
<tr>
<td>DSRVF1LI</td>
<td>Use live version</td>
</tr>
</tbody>
</table>

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 (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>z/OS V2R1.0 MVS Using the Subsystem Interface</td>
</tr>
</tbody>
</table>
SSR Tok 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
- IAZDSERV (Checkpoint Versions Information service)

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>SSJI OK 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>SSJI ERRU 8</td>
<td>Request cannot be completed because of user error, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJI ERRJ 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:
SSI Function Code 71

When SSOBRETN is SSJI E R VR (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>
<tr>
<td>(136)</td>
<td>ALESERV error</td>
</tr>
</tbody>
</table>

When SSOBRETN is SSJI E R RU (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>
<tr>
<td>SSJIINVR (36)</td>
<td>Invalid input data to release. Could be successive releases without an intervening obtain or release without having done an obtain.</td>
</tr>
</tbody>
</table>

When SSOBRETN is SSJI E R RJ (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>SSJIIOK (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:
### Field Name
### Description

**DSRVCVPT**  
Pointer to Checkpoint version

**DSRVCNUM**  
Version number

**DSRVJOTK**  
JOT token

**DSRVJOPT**  
Pointer to JOT

**DSRVJOAL**  
ALET of JOT

**DSRVJQTK**  
JQE token

**DSRVJQPT**  
Pointer to JQE

**DSRVJQAL**  
ALET of JQE

**DSRVQSTK**  
QSE token

**DSRVQSPT**  
Pointer to QSE

**DSRVQSAL**  
ALET of QSE

**DSRVHCTK**  
HCT token

**DSRVHCPT**  
Pointer to HCT

**DSRVHCAL**  
ALET of HCT

**DSRVTIME**  
Time stamp

**DSRVSZE1**  
Version 1 length

**DSRVJNTK**  
JNT token

**DSRVJNPT**  
Pointer to JNT

**DSRVJNAL**  
ALET of JNT

**DSRVSZE2**  
Version 2 length

**DSRVJQXK**  
JQX token

---

**SSI Function Code 71**

Chapter 3. SSI Function Codes Your Program Can Request 135
SSI Function Code 71

DSRVJZPT
   Pointer to JQX

DSRVJXAL
   ALET of JQX

DSRVJTTC
   JQE trackgroup extension token

DSRVJPT
   Pointer to JQE trackgroup extension

DSRVJTAL
   ALET of JQE trackgroup extension

DSRVDAK
   DAS token

DSRVDAPT
   Pointer to DAS

DSRVDAAL
   ALET of DAS

DSRVSZE3
   Version 3 length

DSRVFLG1
   DSERV flags

DSRVF1L1
   Use live version

DSRVJ2LV
   Checkpoint level ($ACTIVATE level)

DSRVSZE4
   Version 4 length

DSRVWQS K
   WLM Q position token

DSRVWQST
   Pointer to WQPOS

DSRVWQSL
   ALET of WQPOS

DSRVSZE5
   DSERV Version 5 fixed parameter length

DSRVJOXK
   JOX token

DSRVOXPT
   Pointer to JOX

DSRVOXAL
   ALET of JOX

DSRVCNPT
   Reserved for subsystem use

DSRVSZE6
   DSERV Version 6 fixed parameter length
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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key.</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>
</tbody>
</table>
**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>
</tbody>
</table>
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>'SSIB'</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. 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.</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.

**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>
<tr>
<td>SSNUFLG1</td>
<td>Flag Byte</td>
</tr>
<tr>
<td></td>
<td>• SSNU1MLO — logon message flag</td>
</tr>
<tr>
<td></td>
<td>If SSNU1MLO is set, a message is issued only if the user is logged on.</td>
</tr>
<tr>
<td>SSNUTKNA</td>
<td>Associated security token of issuing user. This field is optional, and is only valid for authorized callers. For unauthorized callers, the value will be ignored and a reason code (SSNUERCD) of 40 (SSNUUNTK) will be set.</td>
</tr>
<tr>
<td></td>
<td>If not specified, or ignored, a security token is extracted using the caller’s security environment.</td>
</tr>
<tr>
<td></td>
<td>If the message is not destined for the home (local) node, the security token is passed on a WRITER class RACROUTE AUTH call to validate that the user has the authority to issue messages to another NJE node.</td>
</tr>
<tr>
<td></td>
<td>If the message is destined for the home (local) node, the security token is passed on the SEND operator command to verify that the message can be delivered to the destination user.</td>
</tr>
<tr>
<td>SSNUNODE</td>
<td>Node on which messages are sent. For the home node, use the home node name or binary zeros in the SSNUNODE field. Do not use blanks (X‘40’) or an alias name (JES3) because they will not be treated as the home node name. If the destination is the home node, no WRITER class security checks are performed.</td>
</tr>
<tr>
<td>SSNUUSER</td>
<td>User ID to which messages are sent.</td>
</tr>
<tr>
<td>SSNUMLEN</td>
<td>Length of the message pointed to by the SSNUMSG field. The message must be no greater than 100 characters.</td>
</tr>
<tr>
<td>SSNUMSG</td>
<td>Address of the EBCDIC data message that is issued.</td>
</tr>
<tr>
<td>SSNUMEMB</td>
<td>Preferred JES2 member name to issue the SEND on if the user is logged on. If the user is not logged on, the SEND is issued locally. This field is not used by JES3 because the JES3 send is always issued from the global processor.</td>
</tr>
</tbody>
</table>
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 (Decimal)</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>
<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 invalid lengths or formats.</td>
</tr>
<tr>
<td>SSTRNSSSI(24)</td>
<td>The SSI has not been initialized.</td>
</tr>
</tbody>
</table>

**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 (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
</table>
SSI Function Code 75

SSNUOK (0)
The message was issued successfully. The SSNUERCD field contains a zero (SSNURQOK).

SSNUOKB (4)
The message was issued successfully but had a minor error. See the SSNUERCD field in the SSNU control block for the specific reason code.

SSNUERR (8)
The message was not issued. See the SSNUERCD field in the SSNU control block for the specific reason code.

SSNUNEX (12)
The 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'.

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 (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSNURQOK (0)</td>
<td>The request was successful.</td>
</tr>
<tr>
<td>SSNUMSGT (4)</td>
<td>The request was successful, but the message text was truncated because it was too long.</td>
</tr>
<tr>
<td>SSNUEXC (8)</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 (12)</td>
<td>An invalid user ID was specified (blanks or zeros).</td>
</tr>
<tr>
<td>SSNUINVD (16)</td>
<td>An invalid nodename was supplied. The message was not issued.</td>
</tr>
<tr>
<td>SSNUIVID (20)</td>
<td>An invalid identifier (SSNUID) was supplied. The message was not issued.</td>
</tr>
<tr>
<td>SSNUIVER (24)</td>
<td>An invalid version of the SSNU control block was supplied. The message was not issued. The value supplied in the SSNUIVER 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).</td>
</tr>
<tr>
<td>SSNUNOST (28)</td>
<td>Storage in the processing subsystem was not available for the function. The message was not issued.</td>
</tr>
<tr>
<td>SSNUNOAU (32)</td>
<td>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.</td>
</tr>
<tr>
<td>SSNUMSGE (36)</td>
<td>The supplied message address or length was not valid (address specified was zero). The message was not issued.</td>
</tr>
</tbody>
</table>
SSI Function Code 75

SSNUUNTK (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 multitask 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 146.

- **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 150.

- **BULK MODIFY**
  - Modifies selected attributes of one or more data sets. The BULK MODIFY call is described on “BULK MODIFY Requests” on page 151.

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:
When the selection destination field (SSS2DEST) is in the form of A.B, the A portion cannot 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 user ID)
- SSS2DEST (Destination)
- SSS2FORM (Form Numbers)
- SSS2JBIL (Job ID)
- SSS2JOBN (Job Name)
- SSS2ODST (Origination Node Name)
- SSS2PGMN (User Writer Name)
- SSS2PRMO (Process Modes)

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.
SSI Function Code 79

- 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 147.
Chapter 3. SSI Function Codes Your Program Can Request
Programming Considerations for PUT/GET

- The application thread must provide a pointer to an ECB in field SSS2ECBP if the application thread wants JES to post it when newly created work has characteristics matching the thread’s selection criteria. This occurs after JES returns SSS2EODS for a PUT/GET request. If an ECB is not supplied, it is the responsibility of the thread to initiate an IEFSSREQ request.

- For JES3 only, once the application thread begins PUT/GET processing, a COUNT or BULK MODIFY request can not be initiated prior to receiving an SSS2EODS response to a PUT/GET request.

- SSS2CDS contains a 1 for the single returned data set in a SAPI GET/PUT call. If the data set disposition is DELETE, all copies of the data set are deleted.

- Information contained within the SYSOUT data set’s scheduler work blocks (SWBs) can also be returned to the application thread. Much of the information contained within the SWB is normally not processed by JES, and therefore much more information about the data set can be retrieved from the SWB than is returned in fields of the SSS2. Examples of such information contained within the SWB are NAME, BUILDING, ADDRESS, and so on.

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

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

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

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

In the SSS2, reason code field SSS2WRTN contains either a value of SSS2WOK (0) or SSS2WERR (4). SSS2WOK indicates that JES processing needed for SWB retrieval was completely successful, and output fields SSS2SWBT and SSS2SWTU can be used as described above. If SSS2WRTN is set with SSS2WERR, then an error 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.
SSI Function Code 79

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 SET-WTO for additional information concerning the use of the SJFREQ and SWBTUREQ services to retrieve the information in the SWB by either, or both, of the methods described.

- It is the responsibility of the application thread to understand the implications of disposing a data set as KEEP. Because of the potential to process the data set again, the application thread must ensure a loop condition does not arise.

- An EOD (SSOBRETN=SSS2EODS) response is a possible return only for PUT/GET processing. When SAPl 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 IERSSREQ 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 IERSSREQ request after changing its selection criteria.
  - Issue another IERSSREQ 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 150.
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 151. If the application does provide for multitasking, each thread in the application address space must follow the protocol shown in Figure 19 on page 151.

Figure 18. Control Blocks of DYNALLOC Call for SAPI-Provided Data Set
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.
SSI Function Code 79

- **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 153. 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 153.
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.
SSI Function Code 79

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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key.</td>
</tr>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>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 and SSS2 control blocks can reside in storage above 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The application thread should provide an ESTAE-type recovery environment for each task. See <a href="https://pubs.novell.com/np/public/300/300h/index.html">z/OS MVS Programming: Authorized Assembler Services Guide</a> for more information on an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

Figure 21 on page 156 shows the environment at the time of the call for SSI function code 79.
Input Register Information
Before issuing the IEFSSREQ macro, your application thread must ensure that the following general purpose registers contain:

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

13 Address of a standard 18-word save area.

Input Parameters
Input parameters for the function routine are:
• SSOB
• SSIB
• SSS2

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

Field Name
Description
SSOBID
Identifier SSOB
SSOBLLEN
Length of the SSOB (SSOBSIZ) control block
SSOBFUNC
SSI function code 79 (SSOBSOU2)
SSI Function Code 79

SSOBSSIB
Address of an SSIB control block or zero. (If this field is zero, the life-of-job SSIB is used.) See “Subsystem Identification Block (SSIB)” on page 8 for more about the life-of-job SSIB.

SSOBRETN
Return code from JES

SSOBINDV
Address of the function dependent area (SSS2 control block)

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

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

<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>SSBSSSNM</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 SSBSSSNM 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>
</table>
| SSS2LEN    | Input field
The length of the SSS2, set with the value SSS2SIZE |
| SSS2VER    | Input field
Set with the current value of SSS2CVER |
SSS2EYE
Input field
Eye catcher
Set with the character string SSS2

SSS2TYPE
Input field
Type of call. Set with either SSS2PUGE, SSS2COUN, or SSS2BULK below.

SSS2PUGE
Request type of PUT/GET.
Find a data set matching the selection criteria.

SSS2COUN
Request type of Count.
Find 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.
If an application uses a data set token in order to select a single
data set for the bulk modify request, then the request can receive a
return code of SSS2CLON. This return code (in SSOBRETN) will be
given if the single data set selected by token is part of an output
group containing more than one data set and requiring
homogeneous processing (flag SSS2DSH is on). The function is
workable only if the provided token has been returned by SAPI
(SI 79) in a field pointed to by SSS2DSTR or returned by extended
status (SSI 80) in field STSTCTKN.

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.
SSI Function Code 79

**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.
An SSS2RBA specification, with the attendant SSS2CHKP bit, interrupts the normal processing of a group of data sets, sets the RBA for the selected data set, and suspends processing of the data sets in the group. Processing continues with the next eligible group of data sets.
A SSS2RBA specification, with the attendant SSS2CHKP bit, is intended to be used by applications to interrupt the normal processing of a group of data sets, and to defer processing to a later time or terminate the thread.
JES2 only: if SSS2RBA or SSS2CLFT is the only change specified, SSS2RNPT, SSS2RNPR, and SSS2RHLD, if specified, will be applied to the current and remaining data sets in the output group. If there are changes to other characteristics of the data set that would result in the removal of the data set from the group, the remaining data sets in the group will not be affected by SSS2RNPT, SSS2RNPR, and SSS2RHLD specifications, and thus will be eligible for re-selection.

**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.
SSI Function Code 79

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

**SSS2JBN**
Use SSS2JOBN as a filter

**SSS2DUP**
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.

**SSS2DU2**
Give a reason code of SSS2DUPJ if duplicate job. This setting is only meaningful if SSS2JOBN is also set.

**SSS2JBI**
Use SSS2JBIL and SSS2JBIH as filters

**SSS2SEL2**
Input field
Used for selection of new data sets.

**SSS2SPGM**
Use SSS2PGMN as a filter

**SSS2FRM**
Use SSS2FORM as a filter

**SSS2CRE**
Use SSS2CREA as a filter

**SSS2PRM**
Use SSS2PRMO as a filter

**SSS2IPA**
Only select output that has an Internet Protocol (IP) address
SSI Function Code 79

**SSI2SIPN**
Only select output that has no IP address. This setting is mutually exclusive with SSI2SIPA

**SSI2SFCB**
Use SSI2FCB as a filter

**SSI2SUUCS**
Use SSI2UCS as a filter

**SSI2SEL3**
Input field
Used for selection of new data sets.

**SSI2STTC**
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 SSI2SMOD as a filter

**SSI2SFLS**
Use SSI2FLSH as a filter

**SSI2SAGE**
Data sets selected must be at least as old as the value in SSI2SAGE.

**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
SSI Function Code 79

Used for selection of new data sets.

**SSS2CPN**
Select data set having no CPDS.

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

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

**SSS2SODS**
Use SSS2ODST as a filter.

**SSS2SRON**
SSS2SRON 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 SSS2RRO 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.
SSI Function Code 79

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:
- SSS2JOB 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:
- SSS2JBI is not set.
- JES2 is used but JES2 is not running with checkpoint mode z11.

SSS2SIG0
Ignore line and page limits if corresponding actual line and page counts for the data set are zero (if SSS2SENL and SSS2SENP are off).

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 SSS2JOBN on)
SSI Function Code 79

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 SSS2JBIL 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. SSS2JBILH 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, SSS2JBILH 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.

**SSS2JBILH**

Input field

Used for selection of new data sets.

High jobid used for selection (if SSS2JBH on). This value must be null or at least as high as SSS2JBIL.

When SSS2JBH 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 SSS2JBILH 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 SSS2JBILH, are returned. Generics characters '*' or '?' are not allowed.

The following table describes examples of jobs returned for SSS2JBIL when SSS2JBILH is blank. Numeric matches are in normal font, EBCDIC matches are in italicized font.
### Table 3. Examples of jobs returned for SSS2JBIL when SSS2JBILH is blank

<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>
<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, Z100, 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, Z100, 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, etc. EBCDIC match(es): 55555555</td>
</tr>
<tr>
<td>*55555</td>
<td>Numeric match(es): JO555555 or ST555555 etc. EBCDIC match(es), T9555555, J8555555, and so on.</td>
<td>Numeric match(es): JO555555, ZZ555555, and so on. EBCDIC match(es): Z5555555, 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, K7001F0 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>
SSI Function Code 79

**SSS2CREA**
Input field
Used for selection of new data sets. Supports wildcards.
Creator user ID used for selection (if SSS2CRE 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 SSS2PRM 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 user ID portion of the destination can contain the generic characters “*” and “?”.
This can match SYSOUT with a route code that contains a corresponding user ID 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). 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'.
SSI Function Code 79

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

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

**SSS2CLAS**
Input field
Specifies the new class for data sets modified through bulk modify.
New class if SSS2SETC is on.

**SSS2LMIN**
Input field
Used for selection of new data sets.
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).
SSI Function Code 79

**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 SSS2UCS 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.
SSI Function Code 79

**SSI2CTKN**
Input field
Address of client token used for selection (if SSI2SCTK is on).
Mutually exclusive with SSI2SJBI.

**SSI2ODST**
Input field
Specifies the eight-character origination node name from which the job was submitted.
Valid only if SSI2SODS in SSI2SEL5 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.

Table 4 describes the SAPI disposition in SSI2 and the associated queue status options.

<table>
<thead>
<tr>
<th>SAPI Disposition in SSI2</th>
<th>PROCESSED</th>
<th>HOLD</th>
<th>RELEASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field/Flag in SSI2</td>
<td>SSI2DKPE</td>
<td>SSI2DKPE+</td>
<td>SSI2DKPE+</td>
</tr>
<tr>
<td></td>
<td>off</td>
<td>SSI2DHLD</td>
<td>SSI2DRLS</td>
</tr>
<tr>
<td>SYSOUT OUTDISP:</td>
<td></td>
<td></td>
<td>Queue based on resulting OUTDISP</td>
</tr>
<tr>
<td>Keep</td>
<td>Leave</td>
<td>Leave</td>
<td>Keep</td>
</tr>
<tr>
<td>Leave</td>
<td>Purge</td>
<td>Leave</td>
<td>Keep</td>
</tr>
<tr>
<td>Write</td>
<td>Purge</td>
<td>Hold</td>
<td>Write</td>
</tr>
<tr>
<td>Hold</td>
<td>Purge</td>
<td>Hold</td>
<td>Write</td>
</tr>
</tbody>
</table>

**SSI2DSP1**
Input field
Flags describing the disposition for the data set whose name is currently in SSI2DSN.
Settings in SSI2DSP1 and other dispositions are honored if and only if the keep bit (SSI2DKPE) is on. In JES3, the absence of the keep bit implies that the data set will be deleted.
In JES2, if SSI2DKPE is off and the data set has OUTDISP=KEEP, the data set will have OUTDISP=LEAVE after processing. If SSI2DKPE is off and the data set does not have OUTDISP=KEEP, the data set is deleted regardless of other disposition settings in this section.

**SSI2DKPE**
Keep the data set

**SSI2RHLD**
Keep the data set and make it non-selectable (system hold)

**SSI2RNPR**
Keep the data set and leave it selectable, but never return to this SAPI address space
SSI Function Code 79

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

- The JES is restarted
- The application is restarted
- Some characteristic is changed by the operator or another
  application
- Selection by token is requested.

SSS2DHLDS
Hold the data set
This bit is mutually exclusive with SSS2DRLS.

SSS2DRLS
Release the data set
This bit is mutually exclusive with SSS2DHLD.

SSS2CHKP
Use SSS2RBA to checkpoint the data set position. The next data set
returned will have SSS2DSF on. Refer to SSS2RBA for more
information.

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

SSS2RNPT
Leave the data set selectable, but never return to this SYSOUT API
thread again. The data set could be seen by the thread if:

- The JES is restarted
- Some characteristic is changed by the operator to another
  application
- Selection by token is requested.

SSS2DSP2
Input field
Flags describing the disposition for the data set whose name is currently in
SSS2DSN.

SSS2RPRI
Use the priority in SSS2DPRI.

SSS2DNFO
Set forms code to the installation default

SSS2REMV
Removes the data set from the current JOE.
Use this flag with SSS2CTK (select by client token) for a SAPI
thread that will affect a single data set. Use this flag with a pair of
PUT/GET calls, using SSS2CTK on the first call and CTRL on the
second call. This is not restricted in the interface. Using this flag
with other selection criteria can result in incorrect groupings. This
specification is supported by JES2 only.

SSS2RENF
Register Data Set Notification
SSI Function Code 79

All ENF (event notification facility) signals are issued for the data set.

SSS2RENF is only valid for authorized callers and is ignored for unauthorized callers.

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.

An SSS2CLFT specification interrupts the normal processing of a group of data sets, sets the copies left value for the selected data set, and suspends processing of the data sets in the group.

A SSS2CLFT specification is intended to be used by applications to interrupt the normal processing of a group of data sets, and to defer processing to a later time or terminate the thread.

JES2 only: if SSS2CLFT or SSS2RBA is the only change specified, SSS2RNPT, SSS2RNPR, and SSS2RHL, if specified, will be applied to the current and remaining data sets in the output group. If there are changes to other characteristics of the data set that would result in the removal of the data set from the group, the remaining data sets in the group will not be affected by SSS2RNPT, SSS2RNPR, and SSS2RHL specifications, and thus will be eligible for re-selection.

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:
SSI Function Code 79

Register

<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2 — 13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</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>SSRTOK (0)</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
</tr>
<tr>
<td>SSRTDIST (16)</td>
</tr>
<tr>
<td>SSRTLERR (20)</td>
</tr>
<tr>
<td>SSRTNSSI (24)</td>
</tr>
</tbody>
</table>

Output Parameters

Output parameters for the function routine are SSORET and SS2.

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

Value (Decimal)

<table>
<thead>
<tr>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2RTOK (0)</td>
</tr>
<tr>
<td>SSS2EODS (4)</td>
</tr>
</tbody>
</table>
SSI Function Code 79

See the reason codes defined for SSS2REAS at "Reason Codes for SSOBRETN being SSS2EODS" on page 175.

SSS2INVA (8)
Invalid search arguments

SSS2UNAV (12)
Unable to process now

SSS2DUPJ (16)
Duplicate jobnames. (This reason code can occur only if SSS2SDUP is on.) The duplicate job may or may not have characteristics matching the SSS2 filter set.

SSS2IDST (20)
Invalid destination specified

SSS2TKNM (28)
Token map failed.
Application will not be allowed to allocate to data set and DISP=(,KEEP) will be forced

SSS2LERR (32)
Logic error
See the reason codes defined for SSS2REAS at "Reason Codes for SSOBRETN being SSS2LERR."?

SSS2ICLS (36)
SSS2CLAS not A-Z and not 0-9

SSS2BDIS (40)
Disposition settings incorrect
See the reason codes defined for SSS2REAS at "Reason Codes for SSOBRETN being SSS2BDIS" on page 175.

SSS2CLON (44)
Disposition for data set group not uniform (See SSS2DSH).
DISP=(,KEEP) is forced with no override disposition information honored

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

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

Value (Decimal)
Meaning

SSS2RENI (4)
SSS2JEST zero, but SSS2DSN not null

SSS2REIP (8)
SSS2SIPA and SSS2SIPN are mutually exclusive

SSS2RALO (12)
Prior data set still allocated

SSS2RDUP (16)
SSS2SDUP on in SSS2SEL1 and wildcards used in SSS2JOBN

SSS2RJBI (20)
SSS2JBIH < SSS2JBIL and SSS2JBI on
SSI Function Code 79

**SSI2RCRE (24)**
SSI2CREA has error and SSI2CRE on

**SSI2RLEN (28)**
SSI2LEN is less than SSI2SIZE

**SSI2RTYP (32)**
SSI2TYPE is not valid

**SSI2RDES (36)**
SSI2DEST has error and SSI2SDST on

**SSI2RJNM (40)**
SSI2JOBN has error and SSI2SJBN on

**SSI2RFRM (44)**
SSI2FORM has error and SSI2SFRM on

**SSI2RPGM (48)**
SSI2PGMN has error and SSI2SPGM on

**SSI2RPRM (52)**
SSI2PRMO has error and SSI2SPRM on

**SSI2RCLSL (56)**
SSI2CLSL has error and SSI2SCLSL on

**SSI2RFCB (60)**
SSI2FCB has error and SSI2SFCB on

**SSI2RUCLS (64)**
SSI2UCS has error and SSI2SUCS on

**SSI2RCHR (68)**
SSI2CHAR has error and SSI2SCHR on

**SSI2RMO (72)**
SSI2MOD has error and SSI2SMOD on

**SSI2RFL (76)**
SSI2FLSH has error and SSI2SFLS on

**SSI2RLPM (80)**
SSI2LMIN or SSI2LMAX is negative and SSI2SLIN is on -- or -- SSI2PMIN or SSI2PMAX is negative and SSI2SPAG is on

**SSI2RLPG (84)**
SSI2LMIN > SSI2LMAX and SSI2SLIN is on -- or -- SSI2PMIN > SSI2PMAX and SSI2SPAG on

**SSI2RDE2 (88)**
SSI2DES2 has error and SSI2TYPE is SSI2BULK and SSI2ROUT on

**SSI2RVOL (92)**
SSI2VOL has error and SSI2SVOL on

**SSI2REYE (96)**
SSI2EYE does not have SSI2

**SSI2RCTK (100)**
SSI2CTK is on but SSI2CTKN is not specified or not valid.

**SSI2RBRO (104)**
SSI2BRO is on but Bulk Modify or Count was requested.
SSI Function Code 79

SSS2RECJ (108)
SSS2CTK and SSS2JBI are mutually exclusive.

SSS2RODS (112)
SSS2ODST has error and SSS2SRODS on

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

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

<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>SSS2RDPM (192)</td>
<td>SSS2DPM has error</td>
</tr>
<tr>
<td>SSS2RDDS (196)</td>
<td>SSS2DDS has error</td>
</tr>
<tr>
<td>SSS2RDHR (200)</td>
<td>Both SSS2DHLD and SSS2DRLS specified</td>
</tr>
<tr>
<td>SSS2RRON (204)</td>
<td>SSS2RON 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 173 and “Reason Codes for SSOBRETN being SSS2BDIS.”
SSI Function Code 79

**SSS2JEST**
Output field

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

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

**SSS2BTOK**
Output field

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

See “PUT/GET Requests” on page 146 for information concerning programming considerations related to the use of SSS2BTOK.

**SSS2COPY**
Output field

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

**SSS2CPYG**
Output field

Copy groups

**SSS2JOBRI**
Output field

Jobname of selected job

**SSS2BIR**
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 user ID 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
SSI Function Code 79

**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
- 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
SSI Function Code 79

- 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

**SSS2NVOL**
Selecting output based on a volume serial list not supported

**SSS2NNHD**
Returning addresses of NJE headers not supported
SSI Function Code 79

**SSI2NMOD**
Selecting output based on a modification is not supported

**SSI2NPRI**
Selecting output in priority order is not supported

**SSI2NIPA**
IP address selection not supported. Turned on if JES does not support and SSI2SI PA or SSI2SI PN is on

**SSI2RET3**
Output field

**SSI2RSTC**
Data set created by started task

**SSI2RTSU**
Data set created by time sharing user

**SSI2RJOB**
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 Queue where the data set resides

**SSI2RHLV**
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.
SSI Function Code 79

For a SAPI PUT/GET request, these field values are for the single returned data set. For a SAPI COUNT request, these values represent the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

**SSS2LNCT**
- Output field
- Line count
- For a PUT/GET request, this value is for the single returned data set. For a COUNT request, this value represents the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

**SSS2PGCT**
- Output field
- Page count
- For a PUT/GET request, this value is for the single returned data set. For a COUNT request, this value represents the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

**SSS2BYCT**
- Output field
- Byte count after blank truncation
- For a PUT/GET request, this value is for the single returned data set. For a COUNT request, 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.
SSI Function Code 79

See “PUT/GET Requests” on page 146 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 “PUT/GET Requests” on page 146 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
SSI Function Code 79

**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/SEs) 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 IAZSSS2 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 re-fetching the data set, SSS2CTKN must point to the entire 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.

**SSS2WSI**
The Work Selection Identifier assigned to each SYSOUT data set (JES3 only):
- The identifier is a value assigned by JES3 based on the work selection output characteristics of a SYSOUT data set.
- For a job, SYSOUT data sets with identical work selection output characteristics are assigned the same value.
• The assigned values are unique to the job and cannot be used across jobs.
• The value of the identifier does not affect the order of returned data sets.

**SSS2DSNM**
Data set number (key) of the returned data set.

**Job-Level Output-Only Fields**
Similar to the prior ‘Output-Only’ section, but these fields are applicable to all data sets from a single job. The information contained within is on a job-level basis, not on an individual data set-level basis.

**SSS2PNAM**
Output field
Programmer name from the JOB statement

**SSS2ROOM**
Output field
Job level room number

**SSS2NOTN**
Output field
Job notify node

**SSS2NOTU**
Output field
Job notify user ID

**SSS2ACCT**
Output field
Address of encoded accounting information
Accounting information is provided in ‘SMF’ format, just as if it is in type 5 and type 30 SMF records.

**AL1(number-of-pairs-that-follow)**
followed by 0 or more pairs of the form:

**AL1(length),CLlength'string'**
A length of 0 indicates an omitted field

Example: Accounting information of (X3600,42,,ANDY):

```
DC AL1(4) Nr of fields
DC AL1(5),CL5'X3600' field 1
DC AL1(2),CL2'42' field 2
DC AL1(0) field 3 (null)
DC AL1(4),CL4'ANDY' field 4
```

**SSS2XEQ**
Output field
Node where job executed

**SSS2ORG**
Output field
Node where job entered system

**SSS2TIME**
Output field
SSI Function Code 79

Time on input processor for the selected job. This is in hundredths of seconds since midnight.
The time field is local, not UCT/GMT.

**SSS2DATE**
Output field
Date on input processor for the selected job. This is in the form 0cyydddF.
The date field is local, not UCT/GMT.

**SSS2SYS**
Output field
System name of the MVS image where the job output was created.
This field is not available if the SYSOUT came from a network node or the job was reloaded.

**SSS2MBR**
Output field
Member name of the JES2 image where the job output was created.
This field is not available if the SYSOUT came from a network node or the job was reloaded.

**SSS2NJEJ**
Output field
Address of NJE job header. This field is non-zero if the job header is available and SSS2NJEH flag is on.

**SSS2NACT**
Output field
Network account number.
In JES2, this information is retrieved from the /*NETACCT JECL statement.
In JES3, this information is retrieved from the /*NETACCT JECL statement.

**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.
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 187 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 user ID 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 or STATJOBF_64 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 QueueVerbose Element (chained from SJQE)
- SOUT - SYSOUT Element (chained from SJQE)
- SSVE - SYSOUT Verbose Element (chained from SOUT and SJQE)
SSI Function Code 80

Based on the input type you requested, the following table describes the output data elements that are returned.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATTYPE</td>
<td>SJQE</td>
</tr>
<tr>
<td>STATTERS</td>
<td>X</td>
</tr>
<tr>
<td>STATVRBO</td>
<td>X</td>
</tr>
<tr>
<td>STATOUTT</td>
<td>X</td>
</tr>
<tr>
<td>STATOUTV</td>
<td>X</td>
</tr>
<tr>
<td>STATDLST</td>
<td>X</td>
</tr>
</tbody>
</table>

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 or STATTRSA_64, 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 and STATTRSA_64 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 and STATTRSA_64 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 or STATTRSA_64 to a STATJQ or STATSE (obtained previously with no intervening memory management call). The related STATJQ will chain to a verbose element (STATVE). If the terse area is returned in 64-bit storage (STATO164 is set), use STATTRSA_64 instead of STATTRSA. For broadcast SSIs, if STATO164 is on, it is possible to have both 31-bit chained elements anchored from STATJOBF and 64-bit chained elements anchored from STATJOBF_64. In this case, the caller should set STATTRSA if the job is chained to STATJOBF, and STATTRSA_64 if the job is chained to STATJOBF_64. For directed SSIs, a test of STATO164 is adequate to determine which field to set.

- Obtaining verbose SYSOUT level data
  Set STATTYPE to STATOUTV and also set one of the following input fields:
  - Set STATTRSA and STATTRSA_64 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.
  - STATVE 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 STATRSA and STATRSA_64 to zero and STATCTK has STATCTK 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 STATRSA or STATRSA_64 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. If the terse area is returned in 64-bit storage (STATO164 is set), use STATRSA_64 instead of STATRSA. For broadcast SSIs, if STATO164 is on, it is possible to have both 31-bit chained elements anchored from STATJOBF and 64-bit chained elements anchored from STATJOBF_64. In this case, the caller should set STATRSA if the job is chained to STATJOBF, and STATRSA_64 if the job is chained to STATJOBF_64. For directed SSIs, a test of STATO164 is adequate to determine which field to set.

- Set STATRSA or STATRSA_64 to a STATSE (obtained previously with no intervening memory management call). Similar to the case in which STATSECTK is set, verbose job data will be obtained for the job, and all the STATVOs related to the STATSE. If the terse area is returned in 64-bit storage (STATO164 is set), use STATRSA_64 instead of STATRSA. For broadcast SSIs, if STATO164 is on, it is possible to have both 31-bit chained elements anchored from STATJOBF and 64-bit chained elements anchored from STATJOBF_64. In this case, the caller should set STATRSA if the job is chained to STATJOBF, and STATRSA_64 if the job is chained to STATJOBF_64. For directed SSIs, a test of STATO164 is adequate to determine which field to set.

Note: Additional SYSOUT filters can be set (bits in STATSSLx) when STATRSA or STATRSA_64 is set to STATJQ or STATSJBI. The STATSEs are built that match the SYSOUT filters and then STATVOs are built that correspond to each of the STATSEs. If the terse area is returned in 64-bit storage (STATO164 is set), use STATRSA_64 instead of STATRSA. For broadcast SSIs, if STATO164 is on, it is possible to have both 31-bit chained elements anchored from STATJOBF and 64-bit chained elements anchored from STATJOBF_64. In this case, the caller should set STATRSA if the job is chained to STATJOBF, and STATRSA_64 if the job is chained to STATJOBF_64. For directed SSIs, a test of STATO164 is adequate to determine which field to set.

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. Because this is considered a verbose type call (I/O is necessary to obtain the information), only information about a single job can be requested (STATRSA and STATRSA_64 are 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
SSI Function Code 80

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, SYSOUT class, 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
For a description of how to turn job phase codes or job delay reasons, or both, into displayable text, refer to “Text lookup service (IAZTLKUP)” on page 241.

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
• IEFSSIB
• IAZSSST

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key.</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 SSOB, SSIB, and IAZSSST control blocks can reside above or below 16 megabytes in virtual storage.</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 22 on page 190 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:

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

Input Parameters
Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSST

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

Field Name  
Description
SSOBID  
Identifier ‘SSOB’
SSOBLen  
Length of the SSOB (SSOBHSIZ) control block
SSI Function Code 80

SSOBFUNC
SSI function code 80 (SSOBESTA)

SSOBSSIB
Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See “Subsystem Identification Block (SSIB)” on page 8 for more information on the life-of-job SSIB.

SSOBINDV
Address of the function-dependent area (IAZSSST control block).

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

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

Field Name  Description
SSIBID      Identifier ‘SSIB’
SSIBLEN     Length of the SSIB (SSIBSIZE) control block
SSIBSSNM    Subsystem name — name of the subsystem to which this extended status function call is directed (or MSTR if it is to be broadcast).

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

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

Field Name  Description
STATLEN     Length of the IAZSSST (STATSIZE) control block. For STATVER, set to STATV010 or STATV020, a length of at least STATSIZ1 or STATSIZ2 is required. For STATVER set to STATV030 or greater, a length of at least STATSIZ3 is required. STATSIZE is always equated to the largest length of the IAZSSST control block and in general should be used to obtain storage for the IAZSSST and to set STATLEN.
STATEYE      Eyecatcher for the control block (set to C’STAT’)
STATVER      Input version of the IAZSSST control block (set to STATV010 for the initial version of the control block, STATV020 for OS/390 Version 2 Release 4, STATV030 for OS/390 Version 2 Release 5, STATV040 for z/OS V1R7, STATV050 for z/OS V1R9, STATV060 for z/OS V1R10, STATV070 or STATV071 for z/OS V1R11, or STATV080 for z/OS V2R1).
STATTYPE     Function to be performed on this request. Valid functions are:

Field Value  Description
SSI Function Code 80

STATTERS
Requests obtaining terse job level data. The data returned on this call does not require large amounts of system overhead.

STATVRBO
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.

STATMEM
Return memory from a previous request. After one or more requests for data, the memory obtained must be returned using this function.

STATOUTT
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.

STATOUTV
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.

STATDLST
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.

The caller can also set the following fields in the IAZSSST control block on input to limit (or select) the jobs for which data will be returned:

STATSEL1
Flag byte which describes the filters to use to select jobs. Each bit corresponds to a filter field which must match any job returned.

Bit Name | Description
--- | ---
STATSCLS | Apply job class filter in STATCLSL or STATCLSP. Only one class needs to match. If only specifying one class it must be specified in STATCLSL.
STATSDST | 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.
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. STATSJBN cannot be specified with STATSJIL.
SSI Function Code 80

STATSJB
Apply job ID filters in STATJBIL and STATJBIIH. STATSJB cannot be specified with STATSCTK, STATSJIIL or STATSCOR.

STATSOJ
Apply original job ID filter in STATOJBI. 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.

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSSTC</td>
<td>Started tasks are selected.</td>
</tr>
<tr>
<td>STATSTSU</td>
<td>Time sharing users are selected.</td>
</tr>
<tr>
<td>STATSJOB</td>
<td>Batch jobs are selected.</td>
</tr>
<tr>
<td>STATSAPC</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSPRI</td>
<td>Apply JES job priority filter in STATPRIO.</td>
</tr>
<tr>
<td>STATSVOL</td>
<td>Apply SPOOL volume filters in STATVOL (this is valid only when requesting data from a JES2 subsystem).</td>
</tr>
<tr>
<td>STATSPHZ</td>
<td>Apply current job phase in STATPHAZ.</td>
</tr>
<tr>
<td>STATSHLD</td>
<td>Select jobs that are currently held. Setting both STATSHLD and STATSNHL on is the same as setting both bits off.</td>
</tr>
<tr>
<td>STATSNHL</td>
<td>Select jobs that are not currently held. Setting both STATSNHL and STATSHLD on is the same as setting both bits off.</td>
</tr>
<tr>
<td>STATSSYS</td>
<td>Only jobs active on the system listed in STATSYS are returned.</td>
</tr>
</tbody>
</table>
ssi function code 80

statmem
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.

note:
1. in a jes2 environment, wlm service class queue information is always returned regardless of the setting for statspos.
2. in a jes3 environment, wlm service class queue information is returned only if statspos is specified.
3. be aware that there is a performance penalty in a jes3 environment when statspos 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.

bit name
description

statsorg
apply origin node filter in statorgn. only supported by jes2.

statsxeq
apply execution node filter in statxeqn. only supported by jes2.

statssrv
apply wlm service class filter in statsrvc. when filtering by service class and not filtering by job number (statsjbi) nor job phase (statsphz), only jobs on the service class queue specified in statsrvc are returned. when filtering on job number or job phase, any job assigned the service class specified in statsrvc is returned (even if the job is not in a wlm-managed job class). service classes are only available if the job has completed conversion processing and has not completed execution processing. this filter is only supported by jes2 subsystems. statver must be set to statv020 or greater to use this filter.

statssen
apply scheduling environment filter in statsenv.

statsclx
apply job class filter in statclsl and statclsp only to jobs in stat-select or stat-onmain phases.

statsojd
do not apply job name filter in statjobn and statjbnp and job id filters in statjbil and statjbih to jobs that created output with stst1apc on. statsojd cannot be specified with statsjil. (only supported by jes2.)
STATSQPS
Always return current job position in the queue (even if a special queue scan is necessary).

STATSJIL
Use STATJBNP as a list of 8 character JES JOBIDs for which information is to be returned. The complete list is specified using STATJBNP. JOBIDs cannot be specified using STATJBI and STATJBIH when STATSJIL is specified. STATSJIL cannot be specified with STATSJB1, STATSB1,STATSCTK, STATST1, STATSPN, STATSOND or STATSCOR.

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.

Bit Name  Description

STATSCTK
Use the SYSOUT token in STATCTKN as a filter. SYSOUT tokens can be obtained from dynamic allocation or field STSTCTKN from a previous extended status request. STATSCTK cannot be specified with STATSJB1 or STATSCOR.

STATSSOW
Apply the SYSOUT owner filter in STATCSC.

STATSSDS
Apply the SYSOUT destination filter in STATDSC. STATDSC also contains additional SYSOUT destination filters. Mutually exclusive with STATSSL or STATSSNT.

STATSSCL
Apply the SYSOUT class filter in STATCL. STATCL also contains additional SYSOUT class filters.

STATSSWR
Apply the SYSOUT external writer filter in STATSW.

STATSSHL
Select SYSOUT that is currently held. This is the type of hold created by specifying HOLD=YES on the DD statement or OUTDISP=HOLD on the output card. 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.

STATSSNH
Select SYSOUT that is not currently held. Setting both STATSSHL and STATSSNH on is the same as setting both bits off.

STATSSL2
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
SSI Function Code 80

(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.

<table>
<thead>
<tr>
<th>Bit name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSSFR</td>
<td>Apply the SYSOUT forms name filter in STATSFOR. STATVER must be set to</td>
</tr>
<tr>
<td></td>
<td>STATV040 or higher and the JES processing the request must be at the z/OS</td>
</tr>
<tr>
<td></td>
<td>V1R7 level or higher to use these filters. On JES3, the global must be at</td>
</tr>
<tr>
<td></td>
<td>the z/OS V1R7 level or higher.</td>
</tr>
<tr>
<td>STATSSPR</td>
<td>Apply the SYSOUT PRMODE filter in STATSPRM. STATVER must be set to STATV040</td>
</tr>
<tr>
<td></td>
<td>or higher and the JES processing the request must be at the z/OS V1R7 level</td>
</tr>
<tr>
<td></td>
<td>or higher to use these filters. On JES3, the global must be at the z/OS V1R7</td>
</tr>
<tr>
<td></td>
<td>level or higher.</td>
</tr>
<tr>
<td>STATSSSP</td>
<td>Apply the Select SPIN output only filter in STATSSSP. STATVER must be set to</td>
</tr>
<tr>
<td></td>
<td>STATV040 or higher and the JES processing the request must be at the z/OS</td>
</tr>
<tr>
<td></td>
<td>V1R7 level or higher to use these filters. On JES3, the global must be at</td>
</tr>
<tr>
<td></td>
<td>the z/OS V1R7 level or higher.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> STATSSSP and STATSSNS are mutually exclusive. If STATSSSP and</td>
</tr>
<tr>
<td></td>
<td>STATSSNS are both ON or both OFF, then the spin state of the output will not</td>
</tr>
<tr>
<td></td>
<td>be considered.</td>
</tr>
<tr>
<td>STATSSNS</td>
<td>Apply the non-SPIN output only filter in STATSSNS. STATVER must be set to</td>
</tr>
<tr>
<td></td>
<td>STATV040 or higher and the JES processing the request must be at the z/OS</td>
</tr>
<tr>
<td></td>
<td>V1R7 level or higher to use these filters. On JES3, the global must be at</td>
</tr>
<tr>
<td></td>
<td>the z/OS V1R7 level or higher.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> STATSSSP and STATSSNS are mutually exclusive. If STATSSSP and</td>
</tr>
<tr>
<td></td>
<td>STATSSNS are both ON or both OFF, then the spin state of the output will not</td>
</tr>
<tr>
<td></td>
<td>be considered.</td>
</tr>
<tr>
<td>STATSSIP</td>
<td>Select SYSOUT elements that are routed to an IP address. STATVER must be set</td>
</tr>
<tr>
<td></td>
<td>to STATV050 or higher and the JES processing the request must be at the z/OS</td>
</tr>
<tr>
<td></td>
<td>V1R9 level or higher to use these filters.</td>
</tr>
<tr>
<td>STATSSNP</td>
<td>Select SYSOUT elements that are not routed to an IP address. STATVER must be</td>
</tr>
<tr>
<td></td>
<td>set to STATV050 or higher and the JES processing the request must be at the</td>
</tr>
<tr>
<td></td>
<td>z/OS V1R9 level or higher to use these filters.</td>
</tr>
<tr>
<td>STATSSOD</td>
<td>When on with STATSSOW, it indicates to match if SYSOUT is</td>
</tr>
<tr>
<td></td>
<td>destined to STATSCRE on the local node. STATVER must be set to</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SSI Function Code 80

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 STATSB is on, it indicates to match if SYSOUT is destined to STATJOBN or STATJBNP on the local node (ignored if STATSB is off).
- If JES2 is running with checkpoint mode z11 in R11 and STATSB and STATTP are on, it indicates to match if SYSOUT is destined to STATJOBN, STATJBNP or transaction job name on the local node (ignored if STATSB is off).

STATJOBN
Job name filter (used if STATSB 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 STATSB 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’, ‘ST’, ‘TS’, ‘TSU’, ‘S’, ‘ST’, ‘STC’, ‘T’, ‘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. STATJBILH must be blank. Generics characters ‘*’ or ‘?’ are not allowed for verbose requests.

When STATBPI 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 STATBPI 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, STATJBILH must be blank.

Note: INT and IN are valid only in JES3.

STATJBILH
High job ID value (used if STATSB is set). If this field is not specified, then information is only returned using the filter specified in STATJBIL.

When STATJBILH is 2-8 characters and starts with one of the prefixes ‘J’, ‘JO’, ‘JOB’, ‘T’, ‘TS’, ‘TSU’, ‘S’, ‘ST’, ‘STC’, ‘T’, ‘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 STATJBIH suffix are returned. Generics characters ‘*’ or ‘?’ are not allowed.

When STATBPI 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 STATJBIH, 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 STATJBIH is blank. Numeric matches are in normal font, EBCDIC matches are in italicized font.

**Table 6. 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): 5555555</td>
</tr>
<tr>
<td>*55555</td>
<td>Numeric match(es): JO5555555 or ST5555555 and so on. EBCDIC match(es): T9555555, J8555555, and so on.</td>
<td>Numeric match(es): JO5555555, ZZ555555, and so on. EBCDIC match(es): Z5555555, 5555555, 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>
</tbody>
</table>
### Table 6. Examples of jobs returned for STATJBIL when STATJBIH is blank. (continued)

<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>*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, KT0001P0, 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 user ID 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 STATSECL 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 user ID portion of the destination can contain the generic characters ‘*’ and ‘?’.

---

Copyright © 2022, IBM. All Rights Reserved.
SSI Function Code 80

that contains a corresponding user ID 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.

STATORG
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 STASTCLS 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.

STATPRI
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 Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT_INPUT</td>
<td>Job is active in input processing</td>
</tr>
<tr>
<td>STAT_WTCONV</td>
<td>Job is queued for conversion</td>
</tr>
</tbody>
</table>
SSI Function Code 80

STAT_CONV
  Job is actively converting

STAT_VOLWT
  Job is queued for SETUP (not currently used by JES2 code)

STAT_SETUP
  Job is active in SETUP (not currently used by JES2 code)

STAT_SELECT
  Job is queued for execution

STAT_ONMAIN
  Job is actively executing

STAT_SPIN
  JES2 is processing SPIN data sets for the JOB

STAT_WTBKDN
  Job is queued for output processing

STAT_BRKDWN
  Job is active in output processing

STAT_OUTPT
  Job is on the hard copy queue

STAT_WTPURG
  Job is queued for purge

STAT_PURG
  Job is currently being purged

STAT_RECV
  Job is active on an NJE SYSOUT receiver

STAT_WTXMIT
  Job is queued for execution on another NJE node

STAT_XMIT
  Job 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>
</tbody>
</table>
### SSI Function Code 80

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>STAT_ONMAIN</td>
<td>Job is scheduled on main</td>
</tr>
<tr>
<td>STAT_BRKDWN</td>
<td>Job is awaiting breakdown</td>
</tr>
<tr>
<td>STAT_RESTART</td>
<td>Job is awaiting MDS restart processing</td>
</tr>
<tr>
<td>STAT_DONE</td>
<td>Main and MDS processing complete for job</td>
</tr>
<tr>
<td>STAT_OUTPT</td>
<td>Job is awaiting output service</td>
</tr>
<tr>
<td>STAT_OUTQUE</td>
<td>Job is awaiting output service writer</td>
</tr>
<tr>
<td>STAT_OSWAIT</td>
<td>Job is awaiting rsvd services</td>
</tr>
<tr>
<td>STAT_CMPLT</td>
<td>Output service complete for job</td>
</tr>
<tr>
<td>STAT_DEMSEL</td>
<td>Job is awaiting selection on main (demand select job)</td>
</tr>
<tr>
<td>STAT_EFWAIT</td>
<td>Ending function request waiting for I/O completion</td>
</tr>
<tr>
<td>STAT_EFBAD</td>
<td>Ending function request not processed</td>
</tr>
<tr>
<td>STAT_MAXNDX</td>
<td>Maximum request index value</td>
</tr>
</tbody>
</table>

**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...
SSI Function Code 80

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 STATSESE 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.

STATOPT1
Option byte

STATIRAC
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 STRTRONOD, STRTXNOD, STRPRPRE/STRPRND, STRRORE/STRPUND, 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.

STAT1WSI
Specifies that for a STATOUTT request, JES3 will consolidate the SYSOUT elements (STATSE) within a job. One STATSE is returned for each Work Selection Identifier within a job. Each returned STATSE consolidates the information that would normally be returned in multiple SYSOUT elements with identical output characteristics. The STAT1WSI bit is ignored for any STATTYPE that is not equal to STATOUTT. This bit applies to JES3 only.

STAT1LMT
Limits the number of STATJQ elements returned using the value set in STATJQLM. Once the limit is reached, processing stops and control is returned to the caller. An SSOBRETN return code of STATRTOK (0) with a STATREAS reason code of STATRLMT (4) indicates processing ended due to this limit. The request cannot be restarted.

STAT1NDP
Suppresses duplicate data sets being returned by a DSLIST request.

STAT1B64
Specifies whether returned areas are permitted to be obtained by 64-bit storage.
SSI Function Code 80

| STAT1WMS  | Waits for latest MAS level information (JES2 only). |
| STAT1WMB  | Wait for latest member information (JES2 only).      |

**STATSSL3**
More SYSOUT selection criteria. This is supported if STATVER is STATV050 or higher and the corresponding JES2 is z/OS V1R9 or higher.

**STATSSL4**
STATSSL4 is used to support filtering of information returned based on transaction name, transaction job id, or transaction owner.

**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. STATSTPN cannot be specified with STATSIL.

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:
- STATSJB is not set.
- Requesting verbose information
- STATSOJ is not set for JES2.
- JES2 is used but JES2 is not running with checkpoint mode z11.
STATSTPI

Transaction job ID filtering. STATSTPI cannot be specified with STATSJIL.
- 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:
- STATOWN is not set.
- Requesting verbose information
- JES2 is used but JES2 is not running with checkpoint mode z11.

STATSSJ1

If SYSOUT is destined to STATJOBN or STATJBNP on the local node, STATSSJ1 indicates to match using the first jobname supplied in STATJOBN in the following situations:
- If STATSSJ1 is on with STATSBPN
- If STATSTPN is on with STATJSBN

STATSSJ1 is ignored if STATSBJN is off.

STAT1CHR

One byte value that indicates a one character wild card.

STATZOMO

One byte value that indicates a zero or more characters wild card.

STATSEL5

Flag byte that describes the filters to use to select jobs. Each bit corresponds to a filter field which must match any job returned.

STATSCOR

Use STATJCRP as a pointer to a job correlator filter. STATSCOR cannot be specified with STATSJBI, STATSCTK or STATSJIL. If specified on a JES3 request, no results will be returned.

STATJQLM

Limit on how many STATJQs can be returned on this call (used if STAT1LMT is set).

STATTRSA

Pointer to a STATJQ or STATSE (or zero) for which verbose 31-bit data is to
SSI Function Code 80

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.

STATRSA_64
Pointer to a STATJQ or STATSE (or zero) for which verbose 64-bit 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.

STATCTKN
Pointer to the SYSOUT token to be used for selection (used if STATSCTK is set). The token can only be obtained from dynamic allocation or from a previous extended status request.

STATSCRE
The user ID that was in control when the SYSOUT data set was allocated (used if STATSSOW is set). The user ID 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 user ID portion of the destination can contain the generic characters '*' and '?'. This can match SYSOUT with a route code that contains a corresponding user ID 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 selection (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.
SSI Function Code 80

**STATSUBR**
The submitting user ID for selection (used if STATSSUB is set). The submitting user ID 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:

```plaintext
STATCLSL = C'A'
STATCLSN = F'3'
STATCLSP = A(CLASSLST)
CLASSLST = CL8'B',CL8'C',CL8'D'
```

**Table 7. SSI Function Code 80 Filters**

<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>STATSCLS</td>
<td>STATSEL1</td>
<td>STAACLSP</td>
<td>STATCLSN</td>
<td>Job class filters</td>
</tr>
<tr>
<td>STATJOBN</td>
<td>STATSBJBN</td>
<td>STATSEL1</td>
<td>STATJBNP</td>
<td>STATBJBNN</td>
<td>Job name filters</td>
</tr>
<tr>
<td>STATDEST</td>
<td>STATSDST</td>
<td>STATSEL1</td>
<td>STADSTP</td>
<td>STATDSTN</td>
<td>Job destination filters</td>
</tr>
<tr>
<td>STATPHAZ</td>
<td>STATSPHZ</td>
<td>STATSEL3</td>
<td>STAPHZP</td>
<td>STATPHZN</td>
<td>Job phase filters</td>
</tr>
<tr>
<td>STATSCLA</td>
<td>STATSCLL</td>
<td>STATSSL1</td>
<td>STATCLP</td>
<td>STATCLN</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 when STATSBJBN is specified, and a count of job IDs when STATSBJIL is specified

**STATJBNP**
Pointer to STATJOBN extension containing additional 8 character job name filters when STATSBJBN is specified. A list of 8 character job IDs to return when STATSBJIL is specified.

**STATDSTN**
Additional job destination count

**STATDSTP**
Pointer to STATDEST extension containing additional job destination filters

**STATPHZN**
Additional job phase count
SSI Function Code 80

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.

STATJCRP  
Pointer to the job correlator to be used for selection (used if STATSCOR is set).

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.

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The extended status function call has 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 extended status 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</td>
</tr>
</tbody>
</table>
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.

**STATRJST (176)**
Incorrect combination of 31-bit and 64-bit requests.

**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 and STATJOBF_64 for output.</td>
</tr>
</tbody>
</table>

**STATINVA (4)**
The search arguments, though syntactically valid, cannot be used (for
example, specifying a volume serial in STATVOL that is not being used as a SPOOL volume).

**STATLERR (8)**
Logic error in one of the search arguments. See output parameter
STATREAS (below) for details as to the exact error.

**STATINVT (12)**
The request type in STATTYPE is not valid.

**STATREAS Contents:** When SSOBRETN contains a 0 (STATRTOK) indicating a
successful request, the field STATREAS indicates the specific condition 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>0 (0)</td>
<td>Processing completed normally.</td>
</tr>
</tbody>
</table>

**STATRLMT (4)**
Processing ended due to reaching the output limit specified in STATJQLM.
**SSI Function Code 80**

*STATREAS Contents:* When SSOBRTN 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><strong>STATRDST (4)</strong></td>
<td>Destination in STATDEST is not valid.</td>
</tr>
<tr>
<td><strong>STATRJBL (8)</strong></td>
<td>Low job ID in STATJBIL is not valid.</td>
</tr>
<tr>
<td><strong>STATRJBH (12)</strong></td>
<td>High job ID in STATJBIH is not valid.</td>
</tr>
<tr>
<td><strong>STATRJLM (16)</strong></td>
<td>The high job ID in STATJBIH is less than the low job ID in STATJBIL.</td>
</tr>
<tr>
<td><strong>STATRCLS (20)</strong></td>
<td>Job class in STATCLSL is not valid.</td>
</tr>
<tr>
<td><strong>STATRVOL (24)</strong></td>
<td>The volume list in STATVOL is null or has characters that are not that are not allowed.</td>
</tr>
<tr>
<td><strong>STATRJBH (28)</strong></td>
<td>The phase specified in STATPHAZ is either not valid or not supported by this subsystem.</td>
</tr>
<tr>
<td><strong>STATRQUE (32)</strong></td>
<td>Unable to access job queue.</td>
</tr>
<tr>
<td><strong>STATREYE (36)</strong></td>
<td>The eyecatcher in STATEYE is not C'STAT'</td>
</tr>
<tr>
<td><strong>STATRLEN (40)</strong></td>
<td>The length of the IAZSSST specified in STATLEN is too short.</td>
</tr>
<tr>
<td><strong>STATRJBN (44)</strong></td>
<td>The job name in STATJOBN is not valid.</td>
</tr>
<tr>
<td><strong>STATROWN (48)</strong></td>
<td>The owning user ID in STATOWNR is not valid.</td>
</tr>
<tr>
<td><strong>STATRSYS (52)</strong></td>
<td>The system name in STATSYS is not a valid system name.</td>
</tr>
<tr>
<td><strong>STATRMEM (56)</strong></td>
<td>The member name in STATMEMB is not valid.</td>
</tr>
<tr>
<td><strong>STATRCST (60)</strong></td>
<td>STATSEL2 specifies to select only non-batch jobs and batch job class selection was specified in STATSCLS.</td>
</tr>
<tr>
<td><strong>STATROJB (64)</strong></td>
<td>Original job ID in STATOJBI is not valid.</td>
</tr>
<tr>
<td><strong>STATRSEC (68)</strong></td>
<td>The SECLABEL in STATSECL is not valid.</td>
</tr>
<tr>
<td><strong>STATROGR (72)</strong></td>
<td>The origin node in STATORGN is not defined.</td>
</tr>
<tr>
<td><strong>STATRXEQ (76)</strong></td>
<td>The execution node in STATXEQN is not defined.</td>
</tr>
</tbody>
</table>
SSI Function Code 80

STATRPRI (80)
The priority in STATPRIO is not valid for this JES.

STATRSVC (84)
The service class in STATSRVC is not valid.

STATSSEN (88)
The scheduling environment in STATSSEN is not valid.

STATRSCRT (92)
The SYSOUT token pointed to by STATCTKN is not valid.

STATRSCR (96)
The SYSOUT owner in STATSCRE is not valid.

STATRSSD (100)
The SYSOUT destination in STATSDES is not valid.

STATRSSC (104)
The SYSOUT class in STATSCLA is not valid.

STATRSXW (108)
The SYSOUT external writer in STATSWTR is not valid.

STATRECJ (112)
STATSJBI and STATSTCK are mutually exclusive.

STATRVBM (116)
STATVRBO or STATOUTV requested with incorrect filters.

STATRBEA (120)
STATTRSA or STATTRSA_64 does not point to a valid STATJQ or STATSE.

STATRSFR (124)
STATSFOR is not valid.

STATRSPR (128)
STATSPRM is not valid.

STATRSP (132)
Function or filter not supported.

STATRSUB (136)
STATSUBR is not valid.

STATRNEX (140)
STATTRSA or STATTRSA_64 points to a nonexistent job.

STATRIDS (144)
STATRIDS indicates STATSSDS is set with either STATSSLC or STATSSNT.

STATRTRS (148)
STATTERS or STATOUTT requested with incorrect token type (SYSOUT token) specified on STATCTKN.

STATRWIL (152)
Same non zero value specified for both STAT1CHR and STATZOMO.

STATRJIL (156)
STATSJIL is set with either STATSJBN, STATSJBI, STATSSCTK, STATSTPI, STATSTPN, STATSOJD or STATSCOR.

STATRJIP (160)
At least one of the JOBIDs in the list pointed to by STATJBNP is not valid.
SSI Function Code 80

**STATRJIZ (164)**
STATSJIL is set and either STATJBNN or STATJBNP is zero.

**STATRJCR (168)**
The job correlator pointed to by STATJCRP is not valid.

**STATRJCO (172)**
STATSCOR is set with STATSJBI, STATSCTK or STATSJIL.

**Note:** When an extended status request is made as a broadcast call, SSOBRETN is set to the maximum return code of all subsystems that service the call, but STATREAS is set to the reason code of the last subsystem that services the call. It is therefore possible in a case where JES2 and JES3 exist in the same complex for SSOBRETN to be set to STATLERR and STATREAS to be set to 0, if JES3 does not support the function or filter and JES2 does. This does not apply in cases where JES2 supports a function or filter and JES3 does not, because the primary subsystem always services a call before any secondary subsystems and JES3, if present, is always the primary.

**STATREA2 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 “Job information elements” on page 213 for a description of each element. If SYSOUT information is requested, the SYSOUT output elements are chained out of the job level output element (of the owning job). See “SYSOUT information elements” on page 228 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>
<tr>
<td>STATNRSU</td>
<td>The number of SYSOUT elements that match the specified filter requirements.</td>
</tr>
<tr>
<td>STATOFG1</td>
<td>Output information flags.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATO1CP</td>
<td>Information was obtained from a copy of the JOB or output queue. (JES2 Only)</td>
</tr>
</tbody>
</table>
Indicates whether 31-bit pointers, 64-bit pointers, or both (in the case of a broadcast SSI) are used. When STATO164 is set on return, the 64-bit versions of the chaining fields should be used. For broadcast SSIs, if STATO164 is on, it is possible to have both 31-bit chained elements anchored from STATJOBF and 64-bit chained elements anchored from STATJOB_64. In this case, the caller should check both queue heads and use the appropriate chain pointers based on which queue is being processed at the time.

STATOHL
IAZOHLD table for processing STSTHRSN.

STATOHIX
IAZOHLD index table for processing STSTHRSN.

STATJOB_64
Address of first Job Queue Element (64 bit)

STATPHTP
Address of text table used by the "Text lookup service (IAZTLKUP)" on page 241 to convert a job phase code to an equivalent text value.

STATJDTP
Address of text table used by the "Text lookup service (IAZTLKUP)" on page 241 to convert a job delay reason bit value to an equivalent text value.

Job information elements
For each job that matches specified filter requirements, an information element is added to the chain pointed to by STATJOB or STATJOB_64. 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. STATJOB or STATJOB_64 points to the start of the first prefix area. Subsequent areas are chained using the STJQNEXT field. Because the size of the prefix area can vary as a result of service being applied, do not use the equate STJQSIZE to access the data that follows the prefix. To obtain the address of subsequent fields, add the field STJQOHDR to the start of the prefix.

The fields in the STATJQ prefix are:

Field Name
Description

STJQEYE
Eycatcher C’SJQE’.

STJQOHDR
Offset from the start of the STATJQ to the first job information data section.

STJQNEXT
31-bit address of the next STATJQ area on the STATJOB chain.

STJQNEXT_64
64-bit address of the next STATJQ area on the STATJOB_64 chain.
SSI Function Code 80

**STJQOSS**
Name of the subsystem that created this entry.

**STJQSE**
If SYSOUT data is requested, this is the 31-bit head of the SYSOUT information elements (STATSE) for this job.

**STJQSE_64**
If SYSOUT data is requested, this is the 64-bit head of the SYSOUT information elements (STATSE) for this job.

**STJQVRBO**
31-bit address of STATVE for this job.

**STJQVRBO_64**
64-bit address of STATVE for this job.

**STJQVSRB**
31-bit address of first STATVO for this job.

**STJQVSRB_64**
64-bit address of first STATVO for this job.

**Information Element Data Sections:** The variable data sections, which contain information about the job, follow the STATJQ prefix. Each section starts with a 2-byte length, a 1-byte section type, and a 1-byte section modifier. The data length can be from 1 through 65535 bytes. The type and modifier are used to determine the mapping needed to access the data in the section. The first section after the STATJQ prefix is a special 4-byte section which describes the length and type of all sections that follow. The DSECTs that map each section are in the IAZSSST macro.

**Job Queue Element 1st Section:** This section is mapped by the STATJQHD DSECT and is identified by a type of STHD1HDR (0) and a modifier of STHD1MOD (0). This is the only fixed-size section with a length of STHDSIZE (4 bytes). The length in this section is the total length of all sections that follow.

The fields in the STATJQHD section are:

**Field Name**
**Description**

**STHDLEN**
Length of all sections which follow (including this section)

**STHDTYPE**
Section type identifier of STHD1HDR (0)

**STHDMD**
Section type modifier of STHD1MOD (0)

**STHDSD**
Length of this section (4 bytes)

**Job Queue Element Terse Section:** This section is mapped by the STATJQTR DSECT and is identified by a type of STTRTERS (1) and a modifier of STTRTMOD (0). All job information elements have at least one section of this type. This section contains information common to all types of jobs.

The fields in the STATJQTR section are:

**Field Name**
**Description**
SSI Function Code 80

STTRLEN
Length of this section

STTRTYPE
Section type identifier of STTRTERS (1)

STTRMOD
Section type modifier of STTRTMOD (0)

STTRNAME
Job name

STTRJID
Job ID

STTROJID
Original job ID. This might be different from STTRJID if the job was sent using NJE.

STTRCLAS
Job execution class.
In JES2, started tasks (STCs) have a job class of ‘$’ and time sharing users (TSUs) have a job class of ‘@’.

STTRONOD
Job’s origin node. 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 user ID 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 user ID 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).
SSI Function Code 80

**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 Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRJNHL</td>
<td>Job is not held</td>
</tr>
<tr>
<td>STTRJHLD</td>
<td>Job is held</td>
</tr>
<tr>
<td>STTRJHLD</td>
<td>Job is held for duplicate job name</td>
</tr>
</tbody>
</table>

**STTRJTyp**
Type of job

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRSTC</td>
<td>Started task</td>
</tr>
<tr>
<td>STTRTSU</td>
<td>Time sharing user</td>
</tr>
<tr>
<td>STTRJOB</td>
<td>Batch job</td>
</tr>
<tr>
<td>STTRAPPC</td>
<td>APPC initiator</td>
</tr>
</tbody>
</table>

**STTRPRIO**
Job's priority

**STTRARMS**
Job's automatic restart manager status

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRARMR</td>
<td>Job is automatic restart manager registered</td>
</tr>
<tr>
<td>STTRARMW</td>
<td>Job is awaiting automatic restart manager restart</td>
</tr>
</tbody>
</table>

**STTRMISC**
Miscellaneous indicators

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRMSPN</td>
<td>JESLOG for this job is spinable</td>
</tr>
</tbody>
</table>
SSI Function Code 80

STTRPEOM
Indicates job is being process for End of Memory

STTRJCLD
JESJCLIN dataset available

STTRSYSL
MVS SYSLOG job

STTRMXRC
The status of job execution. While the job is executing, the STTRMXCC value (if set) is dependent on the JOBRC= value that is specified for the job. By default, STTRMXCC is set to the highest return code of any executed step; however, STTRMXCC can be the return code of a specific step or the last step that executed. The STTRXREQ bit is set to on if the STTRMXCC value is affected by JOBRC processing.

Field Name
Description

STTRXIND
Job completion indicator. The first four bits indicate how to interpret STTRMXCC. The remaining four bits identify the actual completion type.

Bit Value
Description

STTRXAB
If this bit is on, STTRMXCC contains an ABEND code.

STTRXCDE
If this bit is on, STTRMXCC contains a completion code.

STTRXREQ
The JOBRC completion code was set.

STTRXUNK
No completion information is available. This can occur if the job has not completed, or if the job completed but the completion information was not saved.

STTRXNRM
Job executed and ended normally. (+)

STTRXCC
Job executed and ended by completion code. (+)

STTRXJCL
Job had a JCL error.

STTRXCAN
Job was canceled before execution completed.

STTRXABN
Job ABENDed during execution. (+)

STTRXCAB
Converter ABENDed while processing the job.

STTRXSEC
Job failed input processing security checks.
SSI Function Code 80

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRXEOS</td>
<td>Job failed during execution and was processed in end-of-memory. (+)</td>
</tr>
<tr>
<td>STTRXCNV</td>
<td>Job did not execute due to a converter error.</td>
</tr>
<tr>
<td>STTRXSYS</td>
<td>Job was executing when a system failed.</td>
</tr>
<tr>
<td>STTRMXCC</td>
<td>ABEND or completion code for (+)-marked completion types. If STTRXAB is on, then the field contains an ABEND code–either the first 12 bits of STTRMXCC are set to the System ABEND code, or the last 12 bits are set to the User ABEND code. If STTRXCDE is on, then the field contains a return code in the last 12 bits.</td>
</tr>
<tr>
<td>STTRQPOS</td>
<td>Job's position on its queue. This is only returned when the input flag STATSQPS is set. In JES3, when the job has not been selected for main, but has at least reached the converter/interpreter phase, the job queue position is determined relative to jobs in the same Generalized Main Scheduling (GMS) group that are ahead of the job in question.</td>
</tr>
<tr>
<td>STTRJNUM</td>
<td>Binary job number.</td>
</tr>
<tr>
<td>STTRSPUS</td>
<td>Percent SPOOL utilization. The format is xxx.xxxx. Value is <em><strong>.</strong></em> if unknown.</td>
</tr>
<tr>
<td>STTRSLOG</td>
<td>If this is a SYSLOG job (STTRSYSL is on) MVS system name log is for</td>
</tr>
<tr>
<td>STTRJCOR</td>
<td>Job correlator. Refer to z/OS JES Application Programming for more information on the job correlator.</td>
</tr>
<tr>
<td>STTRSPAC</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>
</tbody>
</table>

Job Queue Element JES2 Terse Section: This section is mapped by the STATJ2TR DSECT and is identified by a type of STJ2TERS (2) and a modifier of STJ2TMOD (0). This section is present if the job information came from a JES2 subsystem. This section contains JES2-specific information common to all types of jobs.

The fields in the STATJ2TR section are:

Field Name  
Description  

STJ2LEN  
Length of this section  

STJ2TYPE  
Section type identifier of STJ2TERS (2)  

STJ2MOD  
Section type modifier of STJ2TMOD (0)  

STJ2FLG1  
General flag byte
### Bit Value

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJ21PRO</td>
</tr>
<tr>
<td>Job is protected</td>
</tr>
<tr>
<td>STJ21IND</td>
</tr>
<tr>
<td>Job is set to independent mode</td>
</tr>
<tr>
<td>STJ21SYS</td>
</tr>
<tr>
<td>Job represents a system data set</td>
</tr>
<tr>
<td>STJ21CNW</td>
</tr>
<tr>
<td>Job can only be processed by a converter that can wait for OS resources</td>
</tr>
<tr>
<td>STJ21RBL</td>
</tr>
<tr>
<td>Job is on the JES2 rebuild queue</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJ2JKEY</td>
</tr>
<tr>
<td>The JES2 job key for the JOB</td>
</tr>
<tr>
<td>STJ2SPOL</td>
</tr>
<tr>
<td>The SPOOL token associated with the job</td>
</tr>
<tr>
<td>STJ2SPAC</td>
</tr>
<tr>
<td>In z/OS 2.1 and later versions, STTRSPAC replaces STJ2SPAC.</td>
</tr>
<tr>
<td>STJ2DPNO</td>
</tr>
<tr>
<td>Binary default print node</td>
</tr>
<tr>
<td>STJ2DPRM</td>
</tr>
<tr>
<td>Binary default print remote</td>
</tr>
<tr>
<td>STJ2DPUS</td>
</tr>
<tr>
<td>Default print user ID</td>
</tr>
<tr>
<td>STJ2INPN</td>
</tr>
<tr>
<td>Binary input node</td>
</tr>
<tr>
<td>STJ2XEQN</td>
</tr>
<tr>
<td>Binary execution node (if job has completed execution).</td>
</tr>
<tr>
<td>STJ2JQEI</td>
</tr>
<tr>
<td>Index of JQE</td>
</tr>
<tr>
<td>STJ2OFSL</td>
</tr>
<tr>
<td>Offload status mask</td>
</tr>
<tr>
<td>STJ2BUSY</td>
</tr>
<tr>
<td>Binary busy byte</td>
</tr>
</tbody>
</table>

#### 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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAFLEN</td>
</tr>
<tr>
<td>Length of this section</td>
</tr>
</tbody>
</table>
SSI Function Code 80

**STAFTYPE**
Section type identifier of STAFFIN (3)

**STAFMOD**
Section type modifier of STAFTMOD (0)

**STAFNUM**
Number of members for which the job has affinity

**STAFMEMB**
First member for which job has affinity. Other member names follow after this member name. The number of member names present is in field STAFNUM.

**Job queue element sections**

**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.

The fields in the STATSCHD section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STSCLEN</strong></td>
<td>Length of this section.</td>
</tr>
<tr>
<td><strong>STSCTYPE</strong></td>
<td>Section type identifier of STSCHED ('04'x)</td>
</tr>
<tr>
<td><strong>STSCMOD</strong></td>
<td>Section type modifier of STSCTMOD ('00'x)</td>
</tr>
<tr>
<td><strong>STSCAHLD</strong></td>
<td>Reasons why the job will not run. Also refer to STSCAHL2.</td>
</tr>
</tbody>
</table>

**Bit Value**

<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>
SSI Function Code 80

STSCFLG1
General flag byte

**Bit Value**
Description

**STSCIJC**
JOBCLASS mode: Off is JES mode; On is WLM mode.

**STSCASID**
ASID where job is executing (zero if not active).

**STSCSRVC**
Service class associated with the job

**STSCESTT**
Estimated time to execute (in seconds) for the job. This is only available if the job:
- Is awaiting execution
- Is scheduled to a WLM-managed job class
- Is not held
- Can currently run (STSCAHLD is zero)

If the estimated time is not available, this field is set to negative 1 (-1). The time is calculated on the average queue time for a job in this job class (STSCAVGQ) and the amount of time this job has been queued (STSCQTIM). If the job has been waiting longer than average, STSCESTT will be set to negative 1 (-1).

**STSCSENV**
Scheduling environment required by the job.

**STSCQPOS**
Position of this job on a WLM service class queue (if STATSPOS is on)

**STSCQNUM**
Number of jobs on this WLM service class queue (if STATSPOS is on)

**STSCQACT**
Number of active jobs on this WLM service class queue (if STATSPOS is on)

**STSCAVGQ**
Average queue time for jobs in this WLM service class. STSCAVGQ is one component of STSCESTT. If STSCESTT is not available, this field is zero (0). If the job has already waited more than the average wait time, this field (and STSCQTIM) is set to negative 1 (-1).

**STSCQTIM**
Actual queue time for this job. STSCQTIM is one component of STSCESTT. If STSCESTT is not available, this field is zero (0). If the job has already waited more than the average wait time, this field (and STSCAVGQ) is set to negative 1 (-1).

**STSCPSEQ**
The minimum z/OS level required for this job to run (in the format used for ECVTPSEQ system field)

**STSCAHLD2**
Reasons why the job will not run. Also refer to STSCAHLD.
SSI Function Code 80

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSCMLEV</td>
<td>No system is available with the minimum required z/OS system level (see STSCPSEQ).</td>
</tr>
</tbody>
</table>

**STWSCSIZE**
Length of basic section

**Job Queue Element Schedulable Systems Section:** This section is mapped by the STATSCHS DSECT and is identified by a type of STSCHED ('04'x) and a modifier of STSSTMOD ('01'x). This section is present if the job is scheduled for execution, requires a scheduling environment, and that environment is available on at least one system. This section lists the MVS system names where the scheduling environment is available. This section is not returned by JES3 subsystems.

The fields in the STATSCHS section are:

**Field Name**
**Description**

| STSSLEN  | Length of this section. |
| STSSTYPE | Section type identifier of STSCHED ('04'x) |
| STSSMOD  | Section type modifier of STSSTMOD ('01'x) |
| STSSNUM  | Number of systems that have the required scheduling environment. |
| STSSSYS  | Name of first system that has the required scheduling environment. Other system names follow after this system name. The number of system names present is in field STSSNUM. |
| STSSSIZE | Length of the scheduling systems section. |

**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 STSSTMOD ('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:

**Field Name**
**Description**

| STSSLLEN | Length of this section. |
| STSLTYPE | Section type identifier of STSECLAF ('05'x) |
SSI Function Code 80

**STSLMOD**
Section type modifier of STSLTMOD ('00')

**STSLNUM**
Number of systems where the SECLABEL is active

**STSLSYS**
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.

**STSLSIZE**
Length of the SECLABEL affinity section.

**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 ElementVerbose 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 '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>31-bit address of the associated job queue data element.</td>
</tr>
<tr>
<td>STVEJOB_64</td>
<td>64-bit 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').
### Field name
### Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STFVLEN</strong></td>
<td>Length of entire Job verbose header (Maximum value is 65535)</td>
</tr>
<tr>
<td><strong>STJVTYPE</strong></td>
<td>Header type identifier of STJV1HDR</td>
</tr>
<tr>
<td><strong>STJVMOD</strong></td>
<td>Section type modifier of STJV1MOD (X'0')</td>
</tr>
<tr>
<td><strong>STJVSIZE</strong></td>
<td>Size of 1st Header Section</td>
</tr>
</tbody>
</table>

**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 a STATVER=STATV040.

### Field name
### Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STVBLEN</strong></td>
<td>Length of this section</td>
</tr>
<tr>
<td><strong>STVBTYPE</strong></td>
<td>Section type identifier of STVBVRBO</td>
</tr>
<tr>
<td><strong>STVBMOD</strong></td>
<td>Section type modifier of STVBVMOD (X'0')</td>
</tr>
<tr>
<td><strong>STVBFLG1</strong></td>
<td>Section flag byte</td>
</tr>
<tr>
<td><strong>Bit value</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>STBB1ERR</strong></td>
<td>Error returning verbose data (terse data section returned)</td>
</tr>
<tr>
<td><strong>STVBJCPY</strong></td>
<td>Job copy count</td>
</tr>
<tr>
<td>This value is meaningful in JES2 only. JES3 always returns 1.</td>
<td></td>
</tr>
<tr>
<td><strong>STVBLNCT</strong></td>
<td>Job line count</td>
</tr>
<tr>
<td>This value is meaningful in JES2 only. JES3 always returns 0.</td>
<td></td>
</tr>
<tr>
<td><strong>STVBIDEV</strong></td>
<td>Input device name</td>
</tr>
<tr>
<td><strong>STVBISID</strong></td>
<td>Input system/member</td>
</tr>
</tbody>
</table>

**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.
SSI Function Code 80

STVBJCIN
   Job input count

STVBJLIN
   Job line count

STVBJPAG
   Job page count

STVBJPUN
   Job card (output) count

STVBRTS
   Input start time/date

   Field name
   Description

   STVBRSTS
      Input start time. This is in hundredths of seconds since midnight.

   STVBRRTSD
      Input start date. This is in the form 0cyydddF.

STVBRTS
   Input end time/date

   Field name
   Description

   STVBRTE
      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 JES 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.
SSI Function Code 80

STVBXTED
Execution end date. This is in the form 0cyydddF.

STVBJUSR
JMRUSEID field

STVBMCLS
Message class (Job card)

STVBNOTN
Notify Node

STVBNOTU
Notify user ID

STVBPNAME
Programmer's name (from Job card)

STVBACT
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 user ID

STVBSUBG
Submitter's security group name. In JES3, this field contains the owner's security group name.

STVBMMLRC
The maximum LRECL of the JCLIN stream

STVBMXRC
The status of job execution. This field is not affected by JOBRC processing.

Field Name
Description

STVBXIND
Job completion indicator. The first four bits indicate how to interpret STVBMXCC. The remaining four bits identify the completion type.

Bit Value
Description

STVBXAB
If this bit is on, STVBMXCC contains an ABEND code.

STVBCODE
If this bit is on, STVBMXCC contains a completion code.
### SSI Function Code 80

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVBXUNK</td>
<td>No completion information is available. This can occur if the job has not completed, or if the job completed but the completion information was not saved.</td>
</tr>
<tr>
<td>STVBXNRM</td>
<td>Job executed and ended normally. (+)</td>
</tr>
<tr>
<td>STVBXCC</td>
<td>Job executed and ended by completion code. (+)</td>
</tr>
<tr>
<td>STVBXJCL</td>
<td>Job had a JCL error.</td>
</tr>
<tr>
<td>STVBXCAN</td>
<td>Job was canceled before execution completed.</td>
</tr>
<tr>
<td>STVBXABN</td>
<td>Job ABENDEd during execution. (+)</td>
</tr>
<tr>
<td>STVBXCAB</td>
<td>Converter ABENDEd while processing the job.</td>
</tr>
<tr>
<td>STVBXSEC</td>
<td>Job failed input processing security checks.</td>
</tr>
<tr>
<td>STVBXEOM</td>
<td>Job failed during execution and was processed in end-of-memory. (+)</td>
</tr>
<tr>
<td>STVBXCNV</td>
<td>Job did not execute due to a converter error.</td>
</tr>
<tr>
<td>STVBXSYS</td>
<td>Job was executing when a system failed.</td>
</tr>
<tr>
<td>STVBMXCC</td>
<td>ABEND or completion code for (+)-marked completion types. If STVBXAB is on, then the field contains an ABEND code–either the first 12 bits of STVBMXCC are set to the System ABEND code, or the last 12 bits are set to the User ABEND code. If STVBXCDRE is on, then the field contains a return code in the last 12 bits.</td>
</tr>
</tbody>
</table>

**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').

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSELEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STSETYPE</td>
<td>Section type identifier of STSESEC</td>
</tr>
<tr>
<td>STSEMOD</td>
<td>Section type modifier of STSESMOD (X'0')</td>
</tr>
<tr>
<td>STSEFLG1</td>
<td>Security Section flag byte</td>
</tr>
</tbody>
</table>
SSI Function Code 80

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 strings:
- 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 zero 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)
SSI Function Code 80

- 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. STATJQSE 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 or SYSEJNXT_64 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>31-bit address of the next STATSE area for this job.</td>
</tr>
<tr>
<td>STSEJNXT_64</td>
<td>64-bit address of the next STATSE area for this job.</td>
</tr>
<tr>
<td>STSEJOB</td>
<td>31-bit address of the STATJQ for the job that owns this SYSOUT.</td>
</tr>
<tr>
<td>STSEJOB_64</td>
<td>64-bit address of the STATJQ for the job that owns this SYSOUT.</td>
</tr>
<tr>
<td>STSEVRBO</td>
<td>31-bit address of the STATVO for the job that owns this SYSOUT.</td>
</tr>
<tr>
<td>STSEVRBO_64</td>
<td>64-bit address of the STATVO 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>
</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>
<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 user ID queue</td>
</tr>
<tr>
<td>STS2OGNM</td>
<td>JOE output group name</td>
</tr>
<tr>
<td>STS2CRTM</td>
<td>JOE create time (STCK format system clock time)</td>
</tr>
<tr>
<td>STS2RNOD</td>
<td>Binary destination node</td>
</tr>
<tr>
<td>STS2RRMT</td>
<td>Binary destination remote number</td>
</tr>
<tr>
<td>STS2RUSR</td>
<td>Destination user route code</td>
</tr>
<tr>
<td>STS2TSWB</td>
<td>JOE level SWB MTTR (8 byte field)</td>
</tr>
</tbody>
</table>
SSI Function Code 80

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.

Field name
Description

STS3LEN
Length of this section

STS3TYPE
Section type identifier of

STS3MOD
Section type modifier of

STS3FLG1
Flag byte:

Bit value
Description

STS31XY
Extended keywords used

STS31WSI
Indicates that one SYSOUT element (STATSE) has been returned for the Work Selection Identifier in STS3WSI. The returned STATSE consolidates the information that would normally be returned in multiple SYSOUT elements with identical output characteristics.

STS31FMT
Indicates that a //*FORMAT JCL statement was used to specify processing instructions to JES3 for the SYSOUT data set. The scheduler facilities call cannot be used to modify or obtain characteristics of the SYSOUT data set.

STS3WSI
The Work Selection Identifier assigned to each SYSOUT data set:

• The identifier is a value assigned by JES3 based on the work selection output characteristics of a SYSOUT data set.

• For a job, SYSOUT data sets with identical work selection output characteristics will be assigned the same value.

• The assigned values are unique to the job and cannot be used across jobs.

STS3SIZE
Length of section

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
SSI Function Code 80

('00'x). All job information elements have at least one section of this type. This section contains information common to all types of jobs.

The fields in the STATSETR section are:

**Field Name**
**Description**

**STSTLEN**
Length of this section.

**STSTTYPE**
Section type identifier of STSTTERS ('41'x)

**STSTMOD**
Section type modifier of STSTTMOD ('00'x)

**STSTOUID**
User ID that owns the SYSOUT

**STSTSECL**
SECLABEL assigned to the SYSOUT

**STSTDEST**
Destination of SYSOUT. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**STSTCLAS**
Class assigned to the SYSOUT. If JES3 is the subsystem returning information, and the data set has not been processed by output processing, this is the class of the first instance of the output.

**STSTNREC**
Number of records in the SYSOUT element

**STSTPAGE**
Number of pages in the SYSOUT element

**STSTLNCT**
Number of lines in the SYSOUT element (JES3 only)

**STSTBYCT**
Number of bytes on spool consumed by the SYSOUT element (JES3 only)

**STSTFORM**
Form assigned to the SYSOUT

**STSTFCB**
Forms control buffer (FCB)

**STSTUCS**
Universal character set (UCS)

**STSTXWTR**
External writer name

**STSTPMDE**
Processing mode (PRMODE)

**STSTFLSH**
Flash

**STSTCHAR**
Character sets assigned to the SYSOUT (JES3 only)

232  z/OS V2R1.0 MVS Using the Subsystem Interface
SSI Function Code 80

STSTMODF
MODIFY=(modname) value (JES3 only)

STSTMODC
MODIFY=(,trc) value (JES3 only)

STSTFLG2
General flag byte

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STST2CIV</td>
<td>The token in STSTCTKN cannot be used. It is not valid. Ensure that the token is valid by verifying that bit STST2CIV is not on.</td>
</tr>
</tbody>
</table>

STSTSYS
MVS system name where output currently being processed (blank if not currently active)

STSTMEM
JES member name where output currently being processed (blank if not currently active).

STSTDEVN
Device name on which output currently being processed (blank if not currently active)

STSTHSTA
Current hold status of the SYSOUT

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSTHOPR</td>
<td>An operator hold has been set using an operator command</td>
</tr>
<tr>
<td>STSTHUSR</td>
<td>A user hold has been set using JCL (such as HOLD=YES on the DD statement)</td>
</tr>
<tr>
<td>STSTHSYS</td>
<td>A system (error) hold has been set (see STSTHRSN for hold reason)</td>
</tr>
<tr>
<td>STSTHTSO</td>
<td>SYSOUT is held for TSO (JES3 only)</td>
</tr>
<tr>
<td>STSTHXWT</td>
<td>SYSOUT is held for an external writer (JES3 only)</td>
</tr>
<tr>
<td>STSTHBDT</td>
<td>SYSOUT is held on the BDT queue (JES3 only)</td>
</tr>
<tr>
<td>STSTHTCP</td>
<td>SYSOUT is held on the TCP queue (JES3 only)</td>
</tr>
</tbody>
</table>

STSTHRSN
System hold reason (see fields OHLDJxxx in IAZOHLD for the definition of possible values)

STSTDISP
Current OUTDISP value for the SYSOUT (JES2 only)

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
</table>
SSI Function Code 80

**STSTDHLD**
OUTDISP=HOLD

**STSTDLVE**
OUTDISP=LEAVE

**STSTDWRT**
OUTDISP=WRITE

**STSTDKEP**
OUTDISP=KEEP

**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

**STSTSOID**
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.

**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.

**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’SSE’F)</td>
</tr>
<tr>
<td>STVOOHDR</td>
<td>Offset to first section</td>
</tr>
<tr>
<td>STVOJOB</td>
<td>31-bit address of associated job queue data element - STATJQ</td>
</tr>
<tr>
<td>STVOJOB_64</td>
<td>64-bit address of associated job queue data element - STATJQ</td>
</tr>
<tr>
<td>STVOJNXT</td>
<td>31-bit address of next verbose SYSOUT element for JOB</td>
</tr>
<tr>
<td>STVOJNXT_64</td>
<td>64-bit address of next verbose SYSOUT element for JOB</td>
</tr>
<tr>
<td>STVOSOUT</td>
<td>31-bit address of associated SYSOUT data element - STATSE</td>
</tr>
<tr>
<td>STVOSOUT_64</td>
<td>64-bit address of associated SYSOUT data element - STATSE</td>
</tr>
<tr>
<td>STVOSNXT</td>
<td>31-bit address of next verbose SYSOUT element for STATSE</td>
</tr>
<tr>
<td>STVOSNXT_64</td>
<td>64-bit 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’).
Field name
Description

STSVLEN
Length of entire SYSOUT verbose element (Maximum value is 65535)

STSVYPE
Section type identifier of STSV1HDR

STSVMOD
Section type modifier of STSV1MOD (X'0')

STSVSIZE
Size of 1st Header Section

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').

Field name
Description

STVSLEN
Length of this section

STVSYPE
Section type identifier of STVSVRBO

STVSMOD
Section type modifier of STVSVMOD (X'0')

STVSFLG1
Section flag byte

Bit value
Description

STVS1ERR
Error obtaining verbose data (terse section returned).

STVSDSCL
Line count, page count, byte count, and record count (STVSLNCT, STVSPGCT, STVSBYCT, and STVSRCCT) are accurate. This bit will not be on if there was an abnormal termination or the data was created on a different node.

STVS1SPN
SPIN data set

STVS1JSL
Spun JESLOG data set

STVS1SYS
System data set

STVS1SIN
Instream data set (SYSIN)

STVS1DUM
Dummy data set (SYSOUT data set which will not print)

STVS1ENF
All ENF signals are issued for this data set
SSI Function Code 80

STVSRECF
Record format

STVSPRCD
Procname for the step creating this data set

STVSSTTPD
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)

STVSMRLRL
Maximum logical record length (LRECL)

STVSLNCT
Line count (valid only if STVSDSCL is ON in STVSFLG1)

Note: For JES3, the line, page, byte, and record counts (STVSLNCT, STVSPGCT, STVSBYCT, STVSRCCT) are updated when the data set is unallocated. Prior to then, the returned values, though valid, are not current.

STVSPGCT
Page count (valid only if STVSDSCL is ON in STVSFLG1)

STVSBYCT
Byte count after blank truncation, 63 bit right justified (valid only if STVSDSCL is ON in STVSFLG1)

STVSRCCT
Record count (JES3 only) (valid only if STVSDSCL is ON in STVSFLG1)

STVSDSN
SYSOUT data set name (valid only if STVSDSCL is ON in STVSFLG1)

STVSCOPY
Data set copy count

STVSFLSC
Number of flash copies

STVSCTKN
SYSOUT data set token
SSI Function Code 80

STVSCCHAR
  Printer translate table

STVSMODF
  MODIFY=(modname)

STVSMODC
  MODIFY=(,trc)

STVSSIZE
  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.

Field name
  Description

STO2LEN
  Length of this section

STO2TYPE
  Section type identifier of STO2VRBO

STO2MOD
  Section type modifier of STO2TMOD (X'0')

STO2FLG1
  General flags

  Bit value
  Description

STO21ERR
  Error obtaining verbose data

STO21ORI
  Field override section is populated (JES2 only).

  Note: JES3 does not use this field override section.

STO2SPST
  Data set SPOOL data token

Field Override Section: The following fields are populated if the data set is part of a demand select JOE. A demand select JOE is indicated by the flag STO21ORI being set to ON. If the system or an individual job specifies demand select, then data sets are gathered into JOEs, regardless of whether the following list of characteristics are matching or not:

• Forms
• FCB
• UCS
• Flash
• Burst

STO2FORM
  Form assigned to the data set

STO2FCB
  Forms Control Buffer (FCB)
### SSI Function Code 80

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STO2UCS</td>
<td>Universal Character Set (UCS)</td>
</tr>
<tr>
<td>STO2FLSH</td>
<td>Flash</td>
</tr>
<tr>
<td>STO2FLG2</td>
<td>General flags</td>
</tr>
<tr>
<td>STO21BRT</td>
<td>Indicates BURST=YES</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 STATSEO3 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</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>
<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 STATSESO DSECT and is identified by a type of STSOSEC and a modifier of STSOSMOD (X'0').

<table>
<thead>
<tr>
<th>Field</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>
</tbody>
</table>
SSI Function Code 80

STSOMOD
Section type modifier of STSOSMOD (X’0’)

Field name
Description

STSOSMOD
Security section modifier

STSOFLG1
flag byte

Bit value
Description

STSO1ERR
Error obtaining verbose data

STSOOFFS
Offset to SAF token

STSOTOKN
Mapped SAF token

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.

Field name
Description

STOTJOBN
APPC transaction program job name that created this data set

STOTJID
APPC transaction program job id that created this data set

STOTSTRT
APPC entry start time

STOTSTRD
APPC entry start date

STOTEXST
APPC execution start time

STOTACTO
APPC account number

STOTSIZE
Length of section

JES3 Unsupported Flags and Fields
Table 8 summarizes which flags and fields are not supported by JES3.

Table 8. JES3 Unsupported Flags and Fields

<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>STATSSEQ</td>
<td>STSXEQN</td>
<td>Execution node name for selection</td>
</tr>
</tbody>
</table>
Text lookup service (IAZTLKUP)

For each job that matches specified filter requirements, an information element is added to the chain pointed to by STATJOBF. There are some fields defined in the sections of the job information element that can be further processed.

The STTRPHAZ field is reported within the Job Queue Element Terse Section, and contains a code that identifies the current phase of the job. Job phase codes are documented under the STATPHAZ field. Applications that display information retrieved using SSI 80 can display text to the end user. Rather than have applications interpret the job phase code, the text lookup service (IAZTLKUP) provides text that is equivalent to the code.

The IAZTLKUP text lookup service enables the user to supply extended status SSI 80 information to the service and retrieve a text description for specific fields. The service supports interpreting the job phase code and job delay reasons, which are reported in the Job Queue Element Execution Scheduling section (STATSCHD) flag bytes STSCAHLD and STSCAHL2. Refer to "Job queue element sections" on page 220. The application can perform an extended status SSI 80 call and then invoke the IAZTLKUP text lookup service to retrieve a text description for the job phase code or any job delay reasons for a given job information element.

IAZTLKUP also provides the capability of returning different levels of text for a given code. This provides the capability to report different amounts of detail and information for a given code or a given delay reason. The text lookup tables provide JES-specific text, and also a generalized text description: the macro invoker indicates which level of text to return. The IAZTLKUP macro documents the text levels that are supported by each text table.

Text lookup for flag bytes such as the job delay reasons work slightly differently that the simple lookup for a job phase code, because there can be multiple reasons for a job execution delay. The text lookup service return the text for the first job delay reason that is encountered, indicating to the macro invoker if there are more job delay reasons to interpret. You can invoke the IAZTLKUP text lookup service in a loop to receive all job delay reasons for a given job information element.

Text lookup (IAZTLKUP) macro

The interface to the text lookup service is the IAZTLKUP macro. The service requires a parameter list which is filled in by the expanded macro code. This parameter list is mapped by the IAZTLKDIF macro. The IAZTLKUP text lookup service can interpret the following fields and return equivalent text information:

- Job phase code (STTRPHAZ)
- Job delay reasons (STSCAHLD and STSCAHL2)

IAZTLKUP syntax

The IAZTLKUP macro uses the following syntax:

```plaintext
IAZTLKUP TABLEID=tableid,
LEVEL=level,
SSOB=ssob,
DATASTR=datastr,
OUTAREA=outarea,
OUTLEN=outlen,
MOREFLG=moreflg,
MF=(E,parmlist)
```
tableid Specifies the 3 character ID of the type of lookup to perform. Supported types are:

- PHZ  Job phase code lookup.
- DLY  Job delay reasons lookup.

level Indicates the level of text to be returned:

- 1  JES-specific text.
- 2  Generalized text description.
- 3  SDSF text description. Only supported by the PHZ table.

ssob Specifies the address of the SSOb parameter list used in the Extended Status SSI 80 request to select the job information element being interpreted, which is supplied in the DATASTR parameter.

datastr Specifies the address of the data structure containing the fields to be interpreted by the service. Extended Status SSI 80 can return data for numerous jobs, so the text lookup service requires the job information element to extract data from for the lookup. For a job phase code lookup, supply the address of a job information element (STATJQ), or the job queue element terse section (STATJQTR) within a job information element. For job delay reason lookup, supply STATJQ or the Job Queue Element Execution Scheduling section (STATSCHD) within a job information element. Refer to "Job queue element sections" on page 220.

outarea Specifies the address of the storage location for the text lookup service to place the text derived from the lookup. If the size of the output area is smaller than the derived text, the text will be truncated. If the output area is larger than the derived text, the remaining bytes will be filled with blanks.

outlen Specifies the length of the supplied output area for the derived text. If this parameter is not supplied, the service will use the length of OUTAREA.

moreflg Specifies the address of the 4-byte storage location where remaining flag reason bits can be stored between IAZTLKUP macro invocations. This parameter is used for flag byte text lookup such as the job delay reasons. Set the MOREFLG location to 0 prior to the first invocation. After the first invocation, a return code of 4 indicates that more flag bits are available for interpretation. A return code of 0 indicates the last flag bit, if any, has been processed. If no flag bits are returned on the first invocation of IAZTLKUP, a return code of 0 and blanks in OUTAREA results.

If MOREFLG is not specified, the service only returns the first flag bit reason. If the service is invoked in a loop without MOREFLG specified, the result can be an infinite loop.

If MOREFLG is specified but not set to zero prior to the first IAZTLKUP macro invocation, the results are unpredictable.

MF Indicator for macro execution:

- L  Allocates storage for the TLKUP DSECT that is used as the input parameter list to the IAZTLKUP service. Allocates an additional 256 bytes to be used as a work area by the IAZTLKUP service. MF must be invoked once with this indicator to set aside storage for the parameter list.
- E  Generates the call to the IAZTLKUP service. Requires the list form (L) to have been previously specified.
SSI Function Code 80

**Parmlist**
Label used to reference the input parameter list TLKUP that is passed to the IAZTLKUP service.

**Input register information**
The IAZTLKUP service requires a non-standard save area address in register 13. This save area must be 128 bytes to allow the IAZTLKUP service to save the 16 8-byte registers.

If the lookup being performed is a flag bit lookup (such as job delay reasons) including the MOREFLG parameter, MOREFLG must be set to zero before the first invocation of the IAZTLKUP service, or unpredictable results can occur.

**Output register information**
The IAZTLKUP service affects the registers in the following manner after exiting the macro:

<table>
<thead>
<tr>
<th>Register</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0/AR0</td>
<td>Destroyed. R0 is used as a work register.</td>
</tr>
<tr>
<td>R1/AR1</td>
<td>Destroyed. R1 is used as a work register.</td>
</tr>
<tr>
<td>R2-R13</td>
<td>Unchanged.</td>
</tr>
<tr>
<td>R14</td>
<td>Destroyed. Used as a return address.</td>
</tr>
<tr>
<td>R15/AR15</td>
<td>Destroyed. R15 contains a return code.</td>
</tr>
</tbody>
</table>

**Return code information**
The IAZTLKUP service supplies a return code in register 15 after exit. The following return codes are possible:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning and Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The IAZTLKUP service was called successfully. For a flag bit lookup (such as job delay reasons), this indicates the last reason was interpreted. If no flag bit settings are found, OUTAREA contains blanks. If the code is not defined in the text lookup table, default text is returned in OUTAREA. For example, an unknown job phase code returns “UNKNOWN JOB PHASE”.</td>
</tr>
<tr>
<td>4</td>
<td>The IAZTLKUP service was called successfully for a flag bit interpretation (such as job delay reasons), and more flag bits remain to be interpreted.</td>
</tr>
<tr>
<td>8</td>
<td>No text value was returned. Check the parameter list field TLKRETCD for further information on the error.</td>
</tr>
</tbody>
</table>

**Environment**

**Minimum Authorization:** Problem or Supervisor state, with any PSW key

**Dispatchable unit mode:** Task
Cross Memory Mode: PASN=HASN=SASN

AMODE: 31 or 64 bit

ASC mode: Primary

Locks: none

Restrictions

None.

Text lookup service data definition (IAZTLKDF) macro

The Text Lookup Service data definition macro IAZTLKDF is used to map the input parameter list passed to the IAZTLKUP text lookup service. The TLKUP DSECT defines the service's input parameter list. The IAZTLKDF macro is invoked in the following manner:

IAZTLKDF DSECT=YES | NO

Input parameter list definition:

DSECT=YES

Generates a DSECT statement for the parameter list structure.

DSECT=NO

Does not generate a DSECT statement, and can be used to reserve storage for the parameter list in an existing DSECT.

IAZTLKDF parameter list return codes (TLKRETCD)

The following text lookup service data definition return codes are provided:

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLKRSUCC</td>
<td>Successful completion. For a flag lookup, no additional text is available.</td>
</tr>
<tr>
<td>TLKRLAST</td>
<td>Successful completion. For a flag lookup, additional text is available.</td>
</tr>
<tr>
<td>TLKRNOTF</td>
<td>Code/flag setting not found.</td>
</tr>
<tr>
<td>TLKRNTBL</td>
<td>No text table pointer in SSI output area.</td>
</tr>
<tr>
<td>TLKRBADS</td>
<td>Bad SSOB address, SSI function not supported, or bad SSI parameter list.</td>
</tr>
<tr>
<td>TLKRBADT</td>
<td>Bad text table ID.</td>
</tr>
<tr>
<td>TLKRNOUS</td>
<td>Output area address or length is zeroes.</td>
</tr>
<tr>
<td>TLKRBADE</td>
<td>Bad text table eyecatcher.</td>
</tr>
<tr>
<td>TLKRBADL</td>
<td>Bad text table level requested.</td>
</tr>
</tbody>
</table>
TLKBADY
Bad text table type for request.

IAZTLKUP service input parameter list (TLKUP)

The TLKUP data structure is the input parameter list to the IAZTLKUP text lookup service. It is built by the code expanded by the expansion of the IAZTLKUP text lookup service macro. The parameter list fields are:

Field name
Description

TLKEYE
Eyecatcher.

TLKLEN
Length of TLKUP parameter list.

TLKTBLID
Text table ID used in the lookup.

TLKLEVEL
Level of text to lookup.

TLKSSOBP
Address of the SSOB.

TLKDATAP
Address of the SSI 80 output data where the code or flag to be interpreted is located (8 byte version).

TLKDATA4
Address of the SSI 80 output data where the code or flag to be interpreted is located (4 byte version).

TLKOUTP
Address of the output area where text is returned (8 byte version).

TLKOUT4
Address of the output area where text is returned (4 byte version).

TLKOUTL
Output area length.

TLKRETCOD
Overall return code.

Length of text supplied

The text tables used by the IAZTLKUP text lookup service are defined within the JES at initialization time. The definitions supply the length of text that can be provided for each text level that is requested. Constants are declared in the Text Lookup Service data definition macro IAZTLKDF to identify the size of text that can be returned by the text lookup service. Current sizes are:

Table 9. Job phase text level

<table>
<thead>
<tr>
<th>Job phase text level</th>
<th>IAZTLKDF constant name</th>
<th>Text size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>TPHZLEN1</td>
<td>64 characters</td>
</tr>
<tr>
<td>Level 2</td>
<td>TPHZLEN2</td>
<td>64 characters</td>
</tr>
<tr>
<td>Level 3</td>
<td>TPHZLEN3</td>
<td>20 characters</td>
</tr>
</tbody>
</table>
Table 10. Job phase text level

<table>
<thead>
<tr>
<th>Job delay text level</th>
<th>IAZTLKDF constant name</th>
<th>Text size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>TDLYLEN1</td>
<td>20 characters</td>
</tr>
<tr>
<td>Level 2</td>
<td>TDLYLEN2</td>
<td>64 characters</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.

```assembly
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 SSIB                              *
***********************************************************************

USING SSOB,STSSOB Est SSOB addressability
LA R0,STSSOB Ensure that
LA R1,L'STSSOB the SSOB
SLR R15,R15 area is
MVCL R0,R14 all zero

MVC SSOBID,'C'SSOB' Set SSOB eyecatcher
MVC SSOBLEN,Y(SSOBHSIZ) Set length of SSOB header
MVC SSOBFUNC,Y(SSOBESTA) Set status 2 function code
MVC SSOBSSIB,'F'0' Use LOJ SSIB
LA R0,SSOB+SSOBHSIZ Point to STAT extension
ST R0,SSOBINDV Point base to extension

USING STAT,SSOB+SSOBHSIZ Est STAT extension addr'bility

MVC STATEYE,'C'STAT' Move in the eyecatcher
```

SSI Function Code 80
SSI Function Code 80

MVC STATLEN,=Y(STATSIZE) Set length of extension
MVC STATVER,=AL1(STATCVRL,STATCVRM) Set current version
MVI STATTYPE,STATTERS Set terse data request

***********************************************************************
* Make only filter this userid *
***********************************************************************
OI STATSEL1,STATSOWN Indicate OWNER is a filter
LA R0,STATOWNR Get area in STAT
LA R1,L'STATOWNR and length
LA R14,THISUSER Get this userid
LA R15,L'THISUSER and length
ICM R15,B'1000',=C' ' Pad with blanks
MVCL R0,R14 Copy parm to STAT

***********************************************************************
* Call the subsystem *
***********************************************************************
MODESET MODE=SUP Supervisor state for SSI function
LA R1,STSSOB Get SSOB address
O R1,=X'80000000' Indicate last SSOB
ST R1,PARMPTR Set parm pointer
LA R1,PARMPTR Get R1 for IEFSSREQ
IEFSSREQ Issue extended status SSI call
LTR R15,R15 Any SSI errors?
BNZ SSREQERX Yes, go process errors
MODESET MODE=PROB Return to problem program state

***********************************************************************
* Process results for IEFSSREQ here *
***********************************************************************
USING STATJQ,R4 Est STATJQ addressability
LA R4,STATJQBF-(STJQNEXT-STATJQ) Get 0th STATJQ
LOOPSTJQ ICM R4,B'1111',STJQNEXT Get next area
BZ DONESTJQ No more, done with STATJQs
LH R3,STJQCHDR Get length of STATJQ
LA R5,STATJQ(R3) Point to 1st section
SLR R2,R2 Get total
ICM R2,B'0011',STHDLLEN-STATJQHD(R5) Header length
LA R5,STHDRSZ(R5) Point to 1st variable section
SL R2,=A(STHDSIZE) Decriment for 1st header length
LOOPSECT CLC 2(2,R5),=AL1(STTRTERS,STTRTMOD) Terse section?
BNE NOTTERSE No, check next type

USING STATJQTR,R5 Est Terse section addr'bly
Process terse section data
DROP R5 Drop terse section
B NEXTSECT Go process next section

NOTTERSE CLC 2(2,R5),=AL1(STJ2TERS,STJ2TMOD) JES2 section?
BNE NOTJES2 No, check next type

USING STATJ2TR,R5 Est JES2 section addr'bly
Process JES2 section data
DROP R5 Drop JES2 section
B NEXTSECT Go process next section

NOTJES2 CLC 2(2,R5),=AL1(STAFFIN,STAFTMOD) Affinity section?
BNE NEXTSECT Not known, get next section

Chapter 3. SSI Function Codes Your Program Can Request 247
SSI Function Code 80

USING STATAFFS,R5
* Process JES2 section data
DROP R5

NEXTSECT SLR R15,R15
Get length of current section
ICM R15, B’0011’,0(R5)
Decrement total count
SR R2,R15
BNP LOOPSTIQ None left, loop
ALR R5,R15
Point to next section
B LOOPSECT
Loop for all sections

DONESTIQ DS 0H
Done processing all elements

***********************************************************************
* Return data area passed *
***********************************************************************

MODESET MODE=SUP
Supervisor state for SSI function

MVI STATTYPE,STATMEM
Set memory management call

LA R1,STSSOB
Get SSOB address
O R1,”X’80000000’
Indicate last SSOB
ST R1,PARMPTR
Set parm pointer
LA R1,PARMPTR
Get R1 for IEFSSREQ

IEFSSREQ
Issue extended status SSI call

MODESET MODE=PROB
Return to problem program state
B EXIT
Go exit the command processor

SSREQERX LR R2,R15
Save return code
MODESET MODE=PROB
Return to problem program state
LR R15,R2
Restore return code
B SSREQERR
Go process error

***********************************************************************
* Process IEFSSREQ error return codes *
***********************************************************************

USING GFDSECTD,R1
Est general failure parm list

SSREQERR LA R1,FAILPARM
Get address of fail parm area
ST R1,PARMPTR
Save in pointer word

ST R15,GFRCODE
Save IEFSSREQ return code
MVC GFCALLID,=Y(GFSSREQ)
Indicate IEFSSREQ error
ST R8,GFCPPPLP
Save CPPL pointer addr
MVC ECBADS,=F’0’
Zero ECB address
LA R0,ECBADSS
Set ECB address
ST R0,GFECBP
Into the PPL

LA R1,PARMPTR
Get addr of parm pointer
LINK EP=IKJEFF19
Call TSO GNRLFAIL service
B EXIT
Return to caller

DROP R1
Drop GFDSECTD

***********************************************************************
* Return to the caller *
***********************************************************************

EXIT L R13,SAVEAREA+4
Get callers save area

STORAGE RELEASE,LENGTH=STATWLEN,ADDR=(R18)
Return local work area

248 z/OS V2R1.0 MVS Using the Subsystem Interface
SSI Function Code 80

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
PARMPTTR DS A    Pointer for MVS calls
ECBADS DS F    CMD processor ECB
FAILPARM DS XL(GFLENGF) Parm area for GNRLFAIL
STSSOB DS XL(SSSTLEN8) Enhanced status SSOB
STATWLEN EQU *-STATWORK    Length of local storage area

***********************************************************************
* Equates *
***********************************************************************
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15

***********************************************************************
* TSO and MVS DSECTs *
***********************************************************************
IKJEFFGF GFDSECT=YES
IEFJESCT ,
IEFJSSOB ,
IAZZSST DSECT=YES
IAZJSSAB ,
IHAPSA ,
IHAAASCBO ,
IHAAASSB ,
IKJTCB ,
IHASTCB ,
CVT DSECT=YES

STATUS2 CSECT ,
END ,
SSI Function Code 82

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 11. 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>&quot;NJE Node Information&quot;</td>
<td>SSJPNJOD/SSJPNJRS</td>
<td>IAZJPNJN</td>
</tr>
<tr>
<td>&quot;SPOOL Volume Information&quot; on page 269</td>
<td>SSJPSPOD/SSJSPRS</td>
<td>IAZJPSPL</td>
</tr>
<tr>
<td>&quot;Initiator Information&quot; on page 289</td>
<td>SSJPITOD/SSJPITRS</td>
<td>IAZJPITD</td>
</tr>
<tr>
<td>&quot;JESPLEX Information&quot; on page 308</td>
<td>SSJPJXOD/SSJPXRS</td>
<td>IAZJPLEX</td>
</tr>
<tr>
<td>&quot;Job Class Information&quot; on page 324</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"
- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment" on page 251
- "Input Register Information" on page 252
- "Input Parameters" on page 252
- "Output Register Information" on page 257
- "Return Code Information" on page 257
- "Output Parameters" on page 257

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 SSJPUSER.

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.
SSI Function Code 82

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
- IEFSSIB
- IAZSSJP
- IAZJPNJN (NJE Node Information)

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJP, and IAZJPNJN, control blocks can reside in 24- or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 23 on page 252 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:

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

Input Parameters: Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJP
- IAZJPNJN (NJE Node Information)

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

Field Name Description
SSOBID Identifier 'SSOB'
SSI Function Code 82

SSOBLen
Length of the SSOB (SSOBHSIZ) control block

SSOBFunc
SSI function code 82 (SSOBSSJP)

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:

Field Name
Description

SSIBID
Identifier ‘SSIB’

SSIBlen
Length of the SSIB (SSIBSIZE) control block

SSIBSSNM
Subsystem name: name of the subsystem to which this NJE Node Information Services request is directed

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:

Field Name
Description

SSJPID
Eyecatcher for the control block (set to ‘SSJP’)

SSJPLEN
Length of the IAZSSJP (SSJPSIZE) control block

SSJPVER
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.

SSJPFreq
Function to be performed on this request. Valid functions and the related SSJIUSER area are:

Field Value
SSJPUSER Description

SSJPNJOD
IAZJPNJN NJE Node Information service, obtain data
SSI Function Code 82

SSJPJNRS
IAZJPJNJE Node Information service, release storage

SSJPUSER
Pointer to service specific data area '(IAZJPJN)'

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

NJE Node Information service, IAZJPJN contents: For the NJE Node Information service (function code SSJPJNOD), the caller must set the following fields in the IAZJPJN control block:

Field Name    Description
NJNLEYE        Eyecatcher of the control block (set to ' SSJPJNL ') .
NJNLLNG        Length of the IAZJPJN (NJNLSIZE) control block.
NJNLVRM        Input version of the IAZJPJN control block. Set to NJNLVRM1 for version 1 of the control block. Set to NJNLVRMC for the current (latest) version.
NJNLOPT1        Processing options:
                Bit Value    Description
NJNLODMC        Perform security label dominance check. This check is always performed for non-authorized callers (JES2 Only).
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 IAZJPJN is used and from that point on the field is managed by the subsystem.

The caller can also set the following fields in the IAZJPJN 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.
SSI Function Code 82

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:
  Bit Value  Description
  NJNL1NAM  Select by the node name specified in NJNLNOD1
  NJNL1RNG  Select by a range of node numbers specified in NJNLRNGL and NJNLRNGH (JES2 Only)
  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:
  Bit Value  Description
  NJNL2PMY  Select nodes managed by path manager (JES2 Only)
  NJNL2PMN  Select nodes not managed by path manager (JES2 Only)
  NJNL2TLS  Select nodes using secure sockets (JES3 Only)
NJNLFLTC  Filter by connection status:
  Bit Value  Description
  NJNLCOWN  Select only the own node (JES2) or the home node (JES3). This filter should not be used with any other connection filter.
  NJNLCADJ  Select adjacent nodes. An adjacent node is one hop away from the own (local) node.
SSI Function Code 82

NJNLCDIR
Select directly attached nodes. 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.

NJNCNCN
Select not-connected nodes. A not-connected node is configured, but JES is unable to communicate with it.

NJNLCPDN
Select nodes pending connection.

NJNLCVIA
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>NJNLAJCY</td>
<td>Select nodes with authority to job commands</td>
</tr>
<tr>
<td>NJNLAJCN</td>
<td>Select nodes without authority to job commands</td>
</tr>
<tr>
<td>NJNLANCY</td>
<td>Select nodes with authority to net commands</td>
</tr>
<tr>
<td>NJNLANCN</td>
<td>Select nodes without authority to net commands</td>
</tr>
<tr>
<td>NJNLASCY</td>
<td>Select nodes with authority to system commands</td>
</tr>
<tr>
<td>NJNLASCN</td>
<td>Select nodes without authority to system commands</td>
</tr>
</tbody>
</table>

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>NJNLSSYS</td>
<td>Filter by the MVS System name specified by NJNLSYNSN</td>
</tr>
<tr>
<td>NJNLSMBR</td>
<td>Filter by the JES Member name specified by NJNLMBRN</td>
</tr>
</tbody>
</table>
SSI Function Code 82

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.

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The NJE Node 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 NJE Node 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
- SSJPRETN
- IAZJPNJN (NJE Node Information service)
SSI Function Code 82

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>
<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, see SSJPRETN for reason code.</td>
</tr>
</tbody>
</table>
| 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.                     |

SSJPSTOR Contents: In addition to the return code in SSOBRETN, the field SSJPSTOR contains the service related error or more specific information about the error. SSJPSTOR 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>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>
</tbody>
</table>
SSI Function Code 82

SSJPGLBL (140)
Function not supported on the global (JES3 only).

NJNDSUBF (256)
Function code specified in SSJPFREQ not supported.

NJNDSPTPE (260)
Invalid NJNLSTRP pointer.

NJNDRNGE (264)
The high bound for the range of node numbers that is specified by
NJNLRNGH is less than the low bound specified by NJNLRNGL

NJNDRNGZ (268)
The low bound for the range of node numbers, NJNLRNGL, is set to zero.

NJNDOWNE (272)
The own or home node filter, NJNLCOWN, should not be used with other
connection status filters.

NJNDSTRE (276)
The caller did not provide enough storage to hold all the data returned by
the subfunction call.

NJNDINTE (280)
Internal error building the system information data area.

NJNLSVRM
Subsystem version number (currently 2).

NJNLDPTR
Pointer to data for first NJE node data area.

NJNLMPTR
Pointer to first system information data area.

NJNLNUM
Number of NJE node data areas returned.

NJNLMNUM
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, 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:

DSECT Name
DSECT Description
SSI Function Code 82

NJNHDR
NJE Node Data Header Section

NJNFPREF
Prefix Section

NJNCMN
NJE Node Common Section

In addition to the preceding common sections, JES2 returns the following sections:

DSECT Name
DSECT Description

N2NGEN
JES2 General Data Section

N2NPATH
JES2 Path Information Section

Note: This is an optional section that contains one or more of the following entries:

N2NPTEN
JES2 Path Information Entry

In addition to the common sections listed above, JES3 returns the following sections:

DSECT Name
DSECT Description

N3NGEN
JES3 General Data Section

N3NPATH
JES3 Path Information Section

Note: This is an optional section that contains one or more of the following entries:

N3NPTEN
JES3 Path Information Entry

The following is a layout of the various sections of the NJE Node Information output data area.

```
+----------------+
| NJNHDR | NJNFPREF | NJNCMN |
+----------------+
| | Prefix Section | NJE Node Common Info Section |
+----------------+
| | | |
+----------------+
| N2NGEN | Optional JES2 NJE Node Info Section |
| | | |
+----------------+
| | | |
+----------------+
| NODE INFO SECTION | |
+----------------+
```

Note: Included if JES2
In addition to the Node information, NLNLMPTR will point to the first element in a separate chain of system information elements. 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>
<tr>
<td>NJNHJPLX</td>
<td>Address of system information entry of member reporting this NJE node.</td>
</tr>
</tbody>
</table>

**Prefix Section:** This section contains the total length of the data returned for an NJE node.
SSI Function Code 82

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>
</tbody>
</table>

**NJNCSFLG**

Node status flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNCSLCL</td>
<td>Set if this node is the own or home node</td>
</tr>
<tr>
<td>NJNCSCNC</td>
<td>Set if this node is a connected node where at least one path is connected</td>
</tr>
<tr>
<td>NJNCSPND</td>
<td>Set if this node is a pending node where at least one path is pending</td>
</tr>
<tr>
<td>NJNCSADJ</td>
<td>Set if this node is an adjacent node</td>
</tr>
<tr>
<td>NJNCSDIR</td>
<td>Set if this node is a directly attached node</td>
</tr>
</tbody>
</table>
SSI Function Code 82

NJNCFGLG1
Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNC1SPW</td>
<td>Send the signon password</td>
</tr>
<tr>
<td>NJNC1VPW</td>
<td>Verify the signon password</td>
</tr>
<tr>
<td>NJNC1EPW</td>
<td>Encrypt the job password</td>
</tr>
<tr>
<td>NJNC1PWL</td>
<td>Local password check (JES3 only)</td>
</tr>
<tr>
<td>NJNC1SSG</td>
<td>Secure sign-on</td>
</tr>
<tr>
<td>NJNC1CSG</td>
<td>Compatible sign-on</td>
</tr>
</tbody>
</table>

NJNCFGLG2
More processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNC2TRC</td>
<td>Trace requested</td>
</tr>
<tr>
<td>NJNC2RST</td>
<td>Autoconnect or restart</td>
</tr>
<tr>
<td>NJNC2HDJ</td>
<td>Hold received jobs</td>
</tr>
<tr>
<td>NJNC2HDS</td>
<td>Hold received SYSOUT</td>
</tr>
</tbody>
</table>

NJNCLINE
The associated line name. This line is:
- a dedicated line (JES2 only)
- a default line (JES3 only)

NJNCRINT
Automatic restart (reconnect) interval in minutes

NJNCRETR
Maximum number of reconnection retries. Zero in this field means an indefinite number of retries.

NJNCSECL
Security label (JES2 only)

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>
</table>
SSI Function Code 82

N2NGLNG
Length of this section

N2NGTYPE
Type of this section

N2NGMOD
Modifier for this section

N2NGNUM
Node number

N2NGSFLG
Node status flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NGSPMD</td>
<td>Path manager is down</td>
</tr>
<tr>
<td>N2NGSNOP</td>
<td>Non path manager mode</td>
</tr>
<tr>
<td>N2NGSEND</td>
<td>End node (no forwarding)</td>
</tr>
<tr>
<td>N2NGSPRV</td>
<td>Private node</td>
</tr>
<tr>
<td>N2NGSDIR</td>
<td>Only allow direct connection</td>
</tr>
</tbody>
</table>

N2NGFLG1
Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NG1ADV</td>
<td>Authority to device commands</td>
</tr>
<tr>
<td>N2NG1AJB</td>
<td>Authority to job commands</td>
</tr>
<tr>
<td>N2NG1ANT</td>
<td>Authority to net commands</td>
</tr>
<tr>
<td>N2NG1ASY</td>
<td>Authority to system commands</td>
</tr>
<tr>
<td>N2NG1XMJ</td>
<td>Transmit jobs</td>
</tr>
<tr>
<td>N2NG1XMS</td>
<td>Transmit SYSOUT</td>
</tr>
<tr>
<td>N2NG1RCJ</td>
<td>Receive SYSOUT</td>
</tr>
<tr>
<td>N2NG1RCS</td>
<td>Receive sysout</td>
</tr>
</tbody>
</table>

N2NGFLG2
More processing flags:
### SSI Function Code 82

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NG2ARS</td>
<td>Accept resistance</td>
</tr>
<tr>
<td>N2NGCMPT</td>
<td>Compaction table id</td>
</tr>
<tr>
<td>N2NGREST</td>
<td>Node resistance</td>
</tr>
<tr>
<td>N2NGSUBN</td>
<td>NJE subnet name</td>
</tr>
<tr>
<td>N2NGLOGM</td>
<td>VTAM® logmode</td>
</tr>
<tr>
<td>N2NGLOGN</td>
<td>Logon device name</td>
</tr>
<tr>
<td>N2NGNSVN</td>
<td>NETSRV name</td>
</tr>
<tr>
<td>N2NGLNID</td>
<td>Binary device identifier for NJNCLINE</td>
</tr>
<tr>
<td>N2NGLGID</td>
<td>Binary device identifier for NJNGLOGN</td>
</tr>
<tr>
<td>N2NGNSID</td>
<td>Binary device identifier for NJNGNSVN</td>
</tr>
</tbody>
</table>

**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:
SSI Function Code 82

Field Name
Description

N2NPSFLG
Path status flags:

Bit Value
Description

N2NPSVLN
Connected by line

N2NPSVMB
Connected by member

N2NPSAWR
Awaiting reset

N2NPSSGN
Signon in progress

N2NPSPND
Connection pending

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
- Signon in progress: N2NPSSGN

Set to the associated member name if the path status is set to:
- Connected by member: N2NPSVMB
- Connection pending: N2NPSPND

N2NPREST
Path resistance

JES3 General Data Section: This section contains attributes unique to JES3.

The fields in the N3NGEN section are:

Field Name
Description

N3NGLNG
Length of this section

N3NGTYPE
Type of this section

N3NGMOD
Modifier for this section

N3NGSFLG
Node connection status:
SSI Function Code 82

Bit Value
Description

N3NGSSNA
Connected via SNA

N3NGSBSC
Connected via BSC

N3NGSTCP
Connected via TCP

N3NGSIND
Indirect node

N3NGSALS
Alias of home node

N3NGSCTC
CTC node

N3NGSSGS
Send signature

N3NGSSGV
Verify signature

N3NGFLG1
Processing flags:

Bit Value
Description

N3NG1DFC
Default class

N3NG1XNR
Writer name is required to hold SYSOUT for external writer

N3NG1NTH
Net hold

N3NG1TLS
Secure socket (TLS)

N3NGEPR
NETPR

N3NGEPU
NETPU

N3NGBUFS
Buffer size

N3NGPRCL
PRTDEF class

N3NGTSCL
PRTTSO class

N3NGXWCL
PRTXWTR class

N3NGPUCL
PUNDEF class
SSI Function Code 82

N3NGPART
   Spool partition
N3NGBDTI
   Bulk data transfer (BDT) id
N3NGSTRM
   Stream
N3NGMAXL
   Maximum number of lines
N3NGNRJT
   Number of job transmitters
N3NGNRJR
   Number of job receivers
N3NGNROT
   Number of output transmitters
N3NGNROR
   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>N3NPPLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>N3NPCTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>N3NPMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>N3NPONENT</td>
<td>Offset to first JES3 Path Information entry</td>
</tr>
<tr>
<td>N3NPNENT</td>
<td>Number of entries</td>
</tr>
<tr>
<td>N3NPSENT</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>NJSHOHDR</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”
- “Use Information”
- “Issued to”
- “Related SSI Codes”
- “Related Concepts”
- “Environment”
- “Input Register Information” on page 271
- “Input Parameters” on page 271
- “Output Register Information” on page 275
- “Return Code Information” on page 275
- “Output Parameters” on page 276

**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 SSJPUER.

**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
SSI Function Code 82

- 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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJP, and IAZJPSPL, 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 z/OS MVS Programming: Assembler Services Guide for more information about an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

Figure 24 on page 271 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>
</tbody>
</table>
SSI Function Code 82

**SSOBLEN**
Length of the SSOB (SSOBHSIZ) control block

**SSOBFUNC**
SSI function code 82 (SSOBSSJP)

**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>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
</tr>
<tr>
<td>Identifier 'SSIB'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBLEN</td>
</tr>
<tr>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBSSNMM</td>
</tr>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPID</td>
</tr>
<tr>
<td>Eyecatcher for the control block (set to 'SSJP')</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPLEN</td>
</tr>
<tr>
<td>Length of the IAZSSJP (SSJPSIZE) control block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPVER</td>
</tr>
<tr>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPFREQ</td>
</tr>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPSPOD</td>
</tr>
<tr>
<td>IAZJPSPL SPOOL Volume Information service, obtain data</td>
</tr>
</tbody>
</table>
SSJPSPR
IAZJPSPL SPOOL Volume Information service, release storage

SSJPUER
Pointer to service specific data area '(IAZJPSPL)'

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

SPOOL Volume Information service, IAZJPSPL contents: For the SPOOL Volume Information service (function code SSJPSPOD), the caller must set the following fields in the IAZJPSPL control block:

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
JPSFULL  Filter on spool partitions that are FULL (JES3 only)

JPSPPNAM  Filter on the spool partition name specified in field JPSPPNAM (JES3 only)

JPSFALD  Filter on spool partitions for which spool allocations are allowed (JES3 only)

JPSFNALD  Filter on spool partitions for which spool allocations are not allowed (JES3 only)
SSI Function Code 82

**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)

**JPSPPPOVF**
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.

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSPACT</td>
<td>Extent Active</td>
</tr>
<tr>
<td>JPSPSTRT</td>
<td>Extent Starting (JES2 Only)</td>
</tr>
<tr>
<td>JPSPDRTN</td>
<td>Extent Draining</td>
</tr>
<tr>
<td>JPSPHALT</td>
<td>Extent Halting (JES2 Only)</td>
</tr>
<tr>
<td>JPSPINAC</td>
<td>Extent Inactive (JES2 Only)</td>
</tr>
<tr>
<td>JPSPHLDR</td>
<td>Extent Held (JES3 Only)</td>
</tr>
<tr>
<td>JPSPBADT</td>
<td>Extent holds a Bad Track (JES3 Only)</td>
</tr>
<tr>
<td>JPSSTTT</td>
<td>Extent STT (JES3 Only)</td>
</tr>
</tbody>
</table>

**JPSPELF2**
Extent filters. Each bit corresponds to a filter condition that must be matched before data is returned.

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSPEXI</td>
<td>Filter on the Extent ID specified in the field JPSPEXTI</td>
</tr>
<tr>
<td>JPSPTGUT</td>
<td>Filter on the track group utilization level specified in the field JPSPTGUT (JES2 Only)</td>
</tr>
<tr>
<td>JPSPTGM</td>
<td>Filter on the minimum number of total track groups specified in the field JPSPTGMN (JES2 Only)</td>
</tr>
<tr>
<td>JPSPAMB</td>
<td>Filter on the JES2 affinity member name specified in the field JPSAMBR (JES2 Only)</td>
</tr>
</tbody>
</table>
**SSI Function Code 82**

**JPSPASY**
Filter on the JES2 affinity MVS system name specified in the field JPSPASYS (JES2 Only)

**JPSPEFL3**
Extent filters:

- **Bit Value**
- **Description**

**JPSPXTND**
Extent EXTENDING

**JPSPMIGR**
Extent MIGRATING

**JPSPMAPP**
Extent MAPPED

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 SSJPSPR (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:

- **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 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.
SSI Function Code 82

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>
</tbody>
</table>
| SSJPPARM (16)  | Error in the parameter list. For example, the SSJ extension is an invalid format
                  • It is not an SSJ
                  • The service version number is not supported
                  • The SSJ 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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
</tbody>
</table>
SSI Function Code 82

SSJPNTDS (8)
SSJPUSER does not point to the correct control block.

SSJPUNSD (12)
Version number in the control block pointed to by SSJPUSER is not correct.

SSJPSMLE (16)
Length field in the control block pointed to by SSJPUSER is too small.

SSJPEYEE (20)
Eyecatcher in the control block pointed to by SSJPUSER is not correct.

SSJPGETM (128)
$GETMAIN failed.

SSJPSTGO (132)
STORAGE OBTAIN failed.

SSJPINVA (136)
Invalid filter arguments were specified.

SSJPGLBL (140)
Function not supported on the global (JES3 only).

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 X'0200').</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>
<tr>
<td>JPSPTGAT</td>
<td>Number of active track groups defined across all partitions.</td>
</tr>
<tr>
<td>JPSPTGAI</td>
<td>Number of active track groups in use across all partitions.</td>
</tr>
<tr>
<td>JPSPTKAT</td>
<td>Number of active tracks across all partitions.</td>
</tr>
<tr>
<td>JPSPTKAU</td>
<td>Number of active tracks in use across all partitions.</td>
</tr>
</tbody>
</table>
SSI Function Code 82

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:

**DSECT Name**

**DSECT Description**

**SPPHDR**
- Partition Header Section

**SPPPREF**
- Partition Prefix Section

**SPPGENI**
- Partition General Information Section

**SPPJES3I**
- Partition JES3 Specific Information Section

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:

**DSECT Name**

**DSECT Description**

**SPEHDR**
- Spool Extent Header Section

**SPEPREF**
- Spool Extent Prefix Section

**SPEGENI**
- Spool Extent General Information Section

**SPEJ2I**
- Spool Extent JES2 Specific Information Section

**SPEJ2AI**
- Spool Extent JES2 Affinity Specific Information

**SPEJ2AE**
- Spool Extent JES2 Affinity Array Entry

**SPEJ3I**
- Spool Extent JES3 Specific Information

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 B2 SPOOL DATA PARAMETER LIST:

```
+----------------+
| JPSPL | ... |
+--------+
| JPSPLPTR |
+--------+
| NOTE: The output partition rollup values reside here. |
+----------------+
```

```
+-----------------------+
| SPOOL PARTITION INFO SECTION: |
+--------------------------+
| SPPHDR | SPPNXTP =----------> POINTER TO THE NEXT |
+--------------------------+
```

z/OS V2R1.0 MVS Using the Subsystem Interface
SSPFRSTE =--------+ ZER0 IF END OF CHAIN.
              NOTE: JES2 will always have 1 partition. JES3 can have multiple.
SPPREF     Prefix Section
SPPGENI    General Partition Info Section
SPPJES3I   Optional JES3 Partition Info Section

SPOOL EXTENT INFO SECTION:
SPEHDR     SPEHDR IN THE CHAIN.
            ZERO IF END OF CHAIN.
SPENXTE    SPEHDR IN THE CHAIN.
            ZER0 IF END OF CHAIN.
SPEPREF    Prefix Section
SPEGENI    General Extent Info Section
SPEJ2I     Optional JES2 Extent Info Section
            NOTE: Included if JES2
SPEJ2AI    Optional JES2 Extent Affinity Array Header Section
            NOTE: Included if JES2 and Affinty exists.
SPEJ2AE    (1..N) Optional JES2 Extent Affinity Array Elements
            NOTE: Included if the SPEJ2AI Extent Array header Section is included. #
            of elements is SPE2ANUM.
SPEJ2MI    Optional JES2 Extent Active Migration Info Section
            NOTE: Included if JES2 and extent is associated with an active migration.
SPEJ3I     Optional JES3 Extent Info Section
            NOTE: Included if JES3.
**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>SPPORF</td>
<td>Offset to the prefix section</td>
</tr>
<tr>
<td>SPPNXTP</td>
<td>Address of the next Spool Partition information element</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 SPPOPRF header field to the header (SPPHDR) address.

The fields in the SPPPREF 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>SPPRTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPPRMD</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.

The fields in the SPPGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPPGLN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>SPPGTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPPGMD</td>
<td>Modifier for this section</td>
</tr>
</tbody>
</table>
### SSI Function Code 82

**SPPGNM**
- Partition name. (Always set to blanks for JES2)

**SPPGTGT**
- Total Track Groups for all extents in this Partition

**SPPGTGU**
- Total Track Group in use for all extents in this Partition

**SPPGTKT**
- Total Tracks for all extents in this Partition

**SPPGTKU**
- Total Tracks in use for all extents in this Partition

**SPPGFLG1**
- Partition Indicators
  
<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPPGNSPC</td>
<td>Set ON when no free space currently exists in the partition</td>
</tr>
<tr>
<td>SPPGACTV</td>
<td>Set ON when active extents exist in the partition</td>
</tr>
<tr>
<td>SPPGALOC</td>
<td>Set ON when some extents have space available that is not currently utilized.</td>
</tr>
</tbody>
</table>

**SPPGTGAT**
- Total active Track Groups for all extents in this Partition

**SPPGTGAU**
- Total active Track Group in use for all extents in this Partition

**SPPGTKAT**
- Total active Tracks for all extents in this Partition

**SPPGTKAU**
- Total active Tracks in use for all extents in this Partition

Spool Partition JES3 Specific Information: This section contains a series of flags that describe JES3 Spool Partition information.

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>
SSI Function Code 82

<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>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPP3MRG</td>
<td>Marginal threshold exceeded</td>
</tr>
<tr>
<td>SPP3MIN</td>
<td>Minimal threshold exceeded</td>
</tr>
</tbody>
</table>

**SPP3MRGP**
Marginal SLIM threshold percentage

**SPP3MINP**
Minimal SLIM threshold percentage

**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 SPENXTE 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>
</tbody>
</table>
SPENXTE
Address of the next Extent Header section

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.

The fields in the SPEGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEGTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPEGMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>SPEGSTS</td>
<td>Extent Status string</td>
</tr>
<tr>
<td>SPEGSTSB</td>
<td>Extent Status byte</td>
</tr>
</tbody>
</table>

Field Value
Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>
SSI Function Code 82

SPEGHELD
   HELD status (JES3 Only)
SPEGXTND
   EXTENDING status
SPEGMIGR
   MIGRATING status
SPEGMAPP
   MAPPED status

SPEGFLG1
   Extent General Status Flags

Bit Name
   Description
SPEGNRML
   Set ON when the extent is in ACTIVE status
SPEGRSVD
   Set ON when the extent is in RESERVED status
SPEGNSEL
   Set ON when work on this extent is not selectable
SPEGPERC
   Percent complete for the in-progress $MSPL spool migration command.

SPEGEXTI
   Extent identifier. This is the Volume Name in JES2 and the DDNAME in JES3.
SPEGDSNM
   Data set name on which this extent physically resides
SPEGTGT
   Total track groups
SPEGGU
   Total track groups in use
SPEGTRR
   Total tracks
SPEGTKU
   Total tracks in use
SPEGLCYL
   Low cylinder. Note that this is a normalized value (cccCC)
SPEGHLHD
   Low head
SPEGLMTR
   Low MQTR value for JES2; Low MMRRRR value for JES3
SPEGLM
   Defines JES3 extent number
SPEGLRRN
   Defines JES3 record number
SPEGHCYL
   High Cylinder. Note this is the normalized value (cccCC)
### Spool Extent JES2 Specific Information

This section holds Extent information specific to JES2.

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 Name</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>
</tbody>
</table>
SSI Function Code 82

SPE2FRMT
   FORMAT command

SPE2HALT
   HALT command

SPE2DRN
   DRAIN command

SPE2XTND
   EXTEND command

SPE2MIGR
   MIGRATE command

SPE2FLG1
   Extent Status Indicators

   Bit Name
   Description

SPE2STNT
   If set ON, this extent is stunted

SPE2ALLM
   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.

SPE2MAPT
   This extent is a target of MAPPED extents

SPE2ACTM
   This extent is either the source or target of an active migration.
   Section SPEJ2MI has the details.

SPE2LFTK
   Largest number of contiguous free tracks

SPE2HTRK
   Highest used track relative to the start of the data set

SPE2TARG
   Target Extent Identifier. This is the Volume Name in JES2 where this extent
   is migrating to, or has migrated to.

Spool Extent JES2 Affinity Specific Information Section: This section holds
system affinity information specific to JES2.

This section only exists if the SPE2ALLM indicator in the JES2 info section (SPEJ2I)
is OFF.

The fields in the SPEJ2AI section are:

   Field Name
   Description

SPE2ALN
   Length of this section

SPE2ATY
   Type of this section
SSI Function Code 82

SPE2AMD
Modifier for this section

SPE2ANUM
Number of entries in the Affinity Array.

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:

Field Name
Description

SPE2EMBR
JES2 member Name

SPE2ESYS
MVS System Name

Spool Extent Active Migration JES2 Specific Information Section: only available when the bit SPE2ACTM is ON in the Spool Extent JES2 Specific Information section SPEJ2I.

This section only exists if the SPE2ACTM indicator in the JES2 info section (SPEJ2I) is ON.

The fields in the SPEJ2MI section are:

Field Name
Description

SPE2MLN
Length of this section

SPE2MTY
Type = JES2 Info

SPE2MMD
Type Mod = JES2 Migration

SPE2MFG1
Extent Active Migration Indicators:

Field Name
Description

SPE2M1SR
Extent is source of migration
SSI Function Code 82

SPE2M1TG
Extent is target of migration

SPE2M1MV
MOVE migration

SPE2M1MG
MERGE migration

SPE2NCAN
Migration cannot be cancelled

SPE2MERR
Migration failed and is being cleaned up

The following fields are only valid if SPE2M1SR is ON; they document where the source extent is migrating to.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE2MPH</td>
<td>Migration phase string.</td>
</tr>
<tr>
<td>SPE2MPHB</td>
<td>Migration phase byte. Contains one of the following values that matches the migration phase string:</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>SPE2NOMG</td>
<td>No migration active</td>
</tr>
<tr>
<td>SPE2MPND</td>
<td>PENDING phase</td>
</tr>
<tr>
<td>SPE2MINI</td>
<td>INITIALIZING phase</td>
</tr>
<tr>
<td>SPE2MSET</td>
<td>SETUP phase</td>
</tr>
<tr>
<td>SPE2MCOPY</td>
<td>COPY phase</td>
</tr>
<tr>
<td>SPE2MCUP</td>
<td>CATCHUP phase</td>
</tr>
<tr>
<td>SPE2MCAN</td>
<td>CANCEL phase</td>
</tr>
<tr>
<td>SPE2MBAK</td>
<td>BACKOUT phase</td>
</tr>
<tr>
<td>SPE2MCLN</td>
<td>CLEANUP phase</td>
</tr>
<tr>
<td>SPE2MMGR</td>
<td>Migrator JES2 MAS member name</td>
</tr>
<tr>
<td>SPE2MVSR</td>
<td>Target extent VOLSER that the current extent is migrating to</td>
</tr>
<tr>
<td>SPE2MDSN</td>
<td>Target extent SPOOL data set name that the current extent is migrating to</td>
</tr>
</tbody>
</table>
SSI Function Code 82

SPEJ2MIS
Size of this section

Spool Extent JES3 Specific Information Section: This section holds Extent information specific to JES3.

The fields in the SPEJ3I section are:
Field Name          Description
-------------------   -----------------------------------------------
SPE3LN               Length of this section
SPE3TY               Type of this section
SPE3MD               Modifier for this section
SPE3RCSZ             Extent record size
SPE3FLG1             Extent Status Indicators:
                      Bit Name                 Description
                        SPE3STRK                Set ON for a Single Track Table
                        SPE3BTRK                Set ON when the extent contains a bad track

Initiator Information
The Initiator Information service reports information about the resources associated with job execution. These resources include initiator groups, the systems on which these groups are enabled, the job classes processed by the initiator groups, and the initiators that are allocated for managing jobs.

See the following sections for more information about Initiator Information:
- “Type of Request”
- “Use Information”
- “Issued to” on page 290
- “Related SSI Codes” on page 290
- “Related Concepts” on page 290
- “Environment” on page 290
- “Input Register Information” on page 291
- “Input Parameters” on page 291
- “Output Register Information” on page 295
- “Return Code Information” on page 296
- “Output parameters” on page 296

Type of Request: Directed SSI Call.

Use Information: To use the JES property information services SSI, callers must specify the sub-function they want to perform in SSJPFREQ. Sub-functions used for
SSI Function Code 82

initiator information are SSJPITOD (obtain data) and SSJPITRS (release storage). The appropriate parameter list must be built according to the JPTTD structure mapped by IAZJPITD and pointed to by SSJPUSER.

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:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJP, and IAZJPITD, 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 z/OS MVS Programming: Assembler Services Guide for more information about an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

[Figure 25 on page 291](#) 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:

**Register Contents**

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

**Input Parameters:** Input parameters for the function routine are:

- SSOB
- SSIB
- SSJP mapped by IAZSSJP
- JPITD mapped by 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>
</tbody>
</table>

---

*Figure 25. Environment at Time of Call for SSI Function Code 82, Initiator Information Subfunction*
SSI Function Code 82

**SSOBLEN**
Length of the SSOB (SSOBHSIZ) control block

**SSOFUNC**
SSI function code 82 (SSOBSSJP)

**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 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.

**SSJP 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>
</tbody>
</table>
Initiator Information service, release storage

IAZJPITD Initiator Information service, release storage

SSJPITRD

Pointer to service specific data area (JPITD mapped by IAZJPITD)

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

Initiator Information service parameter list, JPITD 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>JPITLEN</td>
<td>Length of the JPITD parameter list. Depending on the version of the parameter list (see below), set length to JPITSZE (if you use current version) or JPITSZE1 (if you use version 1).</td>
</tr>
<tr>
<td>JPITSVER</td>
<td>Input version of the IAZJPITD parameter list. Set to JPITSVR1 for version 1. Set to JPITSVR2 for version 2. Set to JPITSVR# for the current (latest) version. Set to JPITCVRL for service version level of IAZJPITD. Set to JPITCVRM for service version modifier of IAZJPITD.</td>
</tr>
<tr>
<td>JPITSTRP</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 JPITD is used. From that point on, the field will be managed by the subsystem.</td>
</tr>
</tbody>
</table>

The caller can also set the following fields in the JPITD parameter list 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 for one or more initiator groups that match the group name indicated by JPITGNAM.</td>
</tr>
<tr>
<td>JPIT1NAS</td>
<td>Return initiator information for one or more systems in a sysplex that match the MVS system name indicated by JPITSNAM.</td>
</tr>
<tr>
<td>JPIT1NAM</td>
<td>Return initiator information for one or more systems in a sysplex that match the JES member name indicated by JPITMNAM.</td>
</tr>
</tbody>
</table>
SSI Function Code 82

**Note:** For JES3, the JES member name is synonymous with the MVS system name.

If neither JPIT1NAS nor JPIT1NAM is specified, SSI will only return data from the system where SSI is called. To request information from other systems or members in a JESPLEX, specify the system or member selection filter.

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 the following 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 JPITSCLS 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 JPITSCLS. 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).

**JPIT1JES**
Return JES mode initiators.

**JPIT1WLM**
Return WLM mode initiators.

**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.
SSI Function Code 82

**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.

**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>

Chapter 3. SSI Function Codes Your Program Can Request 295
SSI Function Code 82

**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 Initiator 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 Initiator 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:

- ITSMHDR (JES2 only)
- SSOBRETN
- SSJPRETN
- Various output data areas (mapped by IAZPJITD macro) are chained to JPI TD - Initiator Information Service parameter list

**ITSMHDR:** System Information Header. See “System Information” on page 457 for more information.

**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:

**Value (Decimal)**

**Meaning**

**SSJPOK (0)**

Request successful.

**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.
SSI Function Code 82

SSJPERRJ (12)
Request cannot be completed, SSJPRETN contains internal reason code.

SSJPPARM (16)
SSJP extension is in 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:

Value (Decimal)
Meaning

SSJPUNSF (4)
Unsupported subfunction requested.

SSJPNTDS (8)
SSJPUSER does not point to the correct control block.

SSJPUNSD (12)
Version number in the control block pointed to by SSJPUSER is not correct.

SSJPSMLE (16)
Length field in the control block pointed to by SSJPUSER is too small.

SSJPEYEE (20)
Eyecatcher in the control block pointed to by SSJPUSER is not correct.

SSJPGETM (128)
$GETMAIN failed.

SSJPSTGO (132)
STORAGE OBTAIN failed.

SSJPINVA (136)
Invalid filter arguments were specified.

SSJPGlbl (140)
Function not supported on the global (JES3 only).

JPItJ2SC (256)
User supplied group of JES2 is not valid with the service class

JPItWLJC (260)
User supplied group of WLM is not valid with the job class

JPItWlST (264)
User supplied group of WLM is not valid with a status filter other than 'Active'

JPItEYLN (268)
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.
SSI Function Code 82

JPITJCLN (272)
User supplied jobclass for the JES2 group filter is longer than one character.

JPITSAF (276)
Internal error extracting caller's security token.

JPITINTE (280)
Internal error building system information data area.

JPITSTRE (284)
Not enough storage to return all data.

JPITDSRV (288)
Internal DSERV error.

JPITASDS (292)
ASDS not accessible.

JPITCMNE (296)
JESPLEX communication error.

Initiator Information parameter list JPITD contents: For the Initiator Information service (function code SSJPIITOD), the following parameters are returned in JPITD:

Field Name Description

JPITVERO Subsystem version number (X'0100' for z/OS release V1R11, X'0200' for z/OS release V1R12 (JES2), X'0200' for z/OS release V1R13 (JES3)).

JPITDPTTR Pointer to the first Initiator Group data area in a chain.

JPITNIG Number of Initiator data areas returned.

JPITMPTR Pointer to the first system information data area.

JPITMNUM Number of system information data areas returned.

For each initiator group that passes the filter requirements, an element is added to the chain pointed to by JPITDPTTR. Each element is composed of the following sections:

DSECT Name DSECT Description

ITGHDHDR Initiator Group Header Section

ITGPDGRP Initiator Group Prefix Section

ITGGDGGI Initiator Group General Information Section

In addition to the common sections listed above, JES3 returns these additional initiator group sections:
DSECT Name
DSECT Description

IT3GDG3I
JES3 Initiator Group Information Section

IT3HDG3S
JES3 Initiator Group System Information Section will contain one or more
IT3SDISY JES3 System Information Entry sections.

IT3SDISY
JES3 System Information Entry

IT3JDG3J
IT3JDG3J JES3 Initiator Group Jobclass Information Section will contain
one or more IT3CD3JC JES3 Jobclass Entry sections.

IT3CD3JC
JES3 Jobclass Entry

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 data returned for each Initiator contains these sections:

DSECT Name
DSECT Description

ITIHDIHD
Initiator Header Section

ITIPDINT
Initiator Prefix Section

ITIGD1GI
Initiator General Information Section

IT2IDI2I
JES2 Initiator Information Section

IT2JD12J
JES2 Initiator Jobclass Information Section will contain one or more of the
IT2CDIJJC entries.

IT2CDIJJC
JES2 Jobclass Entry

The following example is a layout of the various sections of the Initiator
Information output data area.
SSI Function Code 82

INITIATOR HEADER SECTION

ITIHDH

ITIHDIHD

ITIHNEXT

ITIHSYSI

PREFIX SECTION

ITIPDINT

GENERAL INFO SECTION

ITIGDIGI

OPTIONAL JES3

GROUP INFO SECTION

IT3GDG3I

NOTE: # elements is
IT3H3SNS

OPTIONAL JES3

GROUP SYSTEM INFO SECTION

NOTE: # elements is
IT3H3SNS

SYSTEM INFO SECTION

IT3SHD3S

NOTE: # elements is
IT3JJCCCT

OPTIONAL JES3

ARRAY ELEMENTS

IT3SDISY (1..N)

OPTIONAL JES3

GROUP JOBCLASS INFO SECTION

NOTE: # elements is
IT3JJCCCT

OPTIONAL JES3

ARRAY ELEMENTS

IT3CD3JC (1..N)

SYSTEM/MEMBER INFORMATION AREA

NOTE: # of elements is
IT2JJCCCT
Initiator Group Header Section: Each Initiator Group 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 ITGHHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITGHEYE</td>
<td>Eyecatcher. Is 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 data area for this group.</td>
</tr>
<tr>
<td>ITGHNINT</td>
<td>The number of initiator data areas for this group.</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.

The fields in the ITGPDGRP section are:

<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>ITGGGNAM</td>
<td>Group Name</td>
</tr>
<tr>
<td>ITGGFLAG</td>
<td>Group Flags</td>
</tr>
</tbody>
</table>

Bit Name Description

- **ITGGWLM**: ON for WLM mode initiator group
  - OFF for JES mode initiator group

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>IT3GTYPE</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>

Barrier value

- 0-15 job priority
- 16 no barrier
- PRTY each job priority is a barrier

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3GFLAG</td>
<td>JES3 group flag</td>
</tr>
</tbody>
</table>
SSI Function Code 82

Bit Name
Description
IT3GDEFG
ON if this is the JES3 default group
OFF if this is not the JES3 default group

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:
Field Name
Description
IT3HLEN
Length of this section
IT3HTYPE
Type of this section
IT3HMOD
Modifier for this section
IT3H3SOS
Offset to the first of the JES3 System Information Entry sections
IT3H3SNS
Number of JES3 System Information Entry sections
IT3H3SLS
Length of a JES3 System Information entry

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 IT3SDISY section are:
Field Name
Description
IT3SSYSN
System Name
IT3SDICT
Count of initiators defined for this system
IT3SAICT
Count of initiators allocated for this system
IT3SUICT
Count of initiators in use for this system
IT3SFLAG
Flag Byte
Bit Name
Description
IT3SMANA
ON for Manual allocation
SSI Function Code 82

OFF for Dynamic Allocation

**IT3SMANU**
ON for Manual unallocation
OFF for Dynamic unallocation

**IT3SENBS**
ON - Group is enabled for scheduling on this system
OFF - Group is disabled for scheduling on this system

**IT3SJSIE**
Address of JPSYSIFE associated with this JES3 System Information entry.

**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>IT3JCOF</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>IT3JJCLN</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>
<tr>
<td>IT3CSENB</td>
<td>Bitmap relative to system entries in the Group System Information section. If bit is ON, Job-Class is enabled on the corresponding system.</td>
</tr>
<tr>
<td>IT3CSDEF</td>
<td>Bitmap relative to system entries in the Group System Information section. If bit is ON, Job-Class is defined on the corresponding system.</td>
</tr>
<tr>
<td>IT3CSEN2</td>
<td>Bitmap relative to JPSYSIFE system entries, which is anchored by JPITMPTR. Job-Class is enabled on the corresponding system.</td>
</tr>
</tbody>
</table>
SSI Function Code 82

IT3CSDE2
   Bitmap relative to JPSYSIFE system entries, which is anchored by
   JPITMPTR. Job-Class is defined on the corresponding system.

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:
Field Name    Description
ITIHIEYE      Eyecatcher. Is set to ‘DINITDTA’.
ITIHOHDR      Offset to the prefix section
ITIHNEXT      Address of the next individual initiator information section.
ITIH SYSI     Address of JPSYSIFE for this initiator.

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:
Field Name    Description
ITIPILEN      Combined length of the individual initiator sections. This does not include
              the length of the header section.
ITIPITYP      Type of this section
ITIPIMOD      Modifier for this section

Initiator General Information Section: This section holds general information
about an individual initiator.

The fields in the ITIGDIGI section are:
Field Name    Description
ITIGIILN      Length of this section
ITIGIITY      Type of this section
ITIGIIMD      Modifier for this section
ITIGASID      Address Space Identifier for the initiator job
ITIGSTAT
Initiator Status flag

Bit Value Description

ITIGIDRI Draining
ITIGDDRD Drained
ITIGIHLI Halting
ITIGIHLD Halted
ITIGIINA Inactive
ITIGIACT Active
ITIGISTR Starting

ITIGMVSN MVS System name

ITIGSID JES Member name

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.

Note: Information on the job that is currently active on the initiator is only returned if the SSI caller is authorized to access data for the job, specifically, if the caller’s security label dominates the security label of the job.
**SSI Function Code 82**

**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>IT2IITID</td>
<td>Initiator partition identifier</td>
</tr>
<tr>
<td>ITIIJID</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>ITJJJCOS</td>
<td>Offset to the first job-class entry</td>
</tr>
<tr>
<td>ITJJJCCT</td>
<td>Number of job-class entries</td>
</tr>
<tr>
<td>ITJJCLN</td>
<td>Length of a single job-class entry</td>
</tr>
</tbody>
</table>

**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 or job class group name</td>
</tr>
<tr>
<td>IT2CFLAG</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>
### SSI Function Code 82

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT2CJCWY</td>
<td>ON if the job class is WLM eligible; OFF if the job class is not WLM eligible</td>
</tr>
<tr>
<td>IT2CJGRP</td>
<td>ON if IT2CJCNM is a job class group name; OFF if it is a job class name</td>
</tr>
</tbody>
</table>

#### 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 309
- "Input Parameters" on page 310
- "Output Register Information" on page 313
- "Return Code Information" on page 313
- "Output Parameters" on page 314

**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 SSIPUSER.

**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
- IEFSSIB
- IAZSSJP
- IAZJPLEX (JESPLEX Information)
SSI Function Code 82

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 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 z/OS MVS Programming: Assembler Services Guide for more information about an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

Figure 26 shows the environment at the time of the call for SSI function code 82, JESPLEX Information Subfunction.

Figure 26. Environment at Time of Call for SSI Function Code 82, JESPLEX Information Subfunction

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

Register Contents
SSI Function Code 82

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
- 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>SSOBRID</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>
<tr>
<td>SSOBSIB</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="#">&quot;Subsystem Identification Block (SSIB)&quot; on page 8</a> for more information about the life-of-job SSIB</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (IAZSSJP 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 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>
</table>
**SSI Function Code 82**

**SSJPID**
Eyecatcher for the control block (set to ‘SSJP’)

**SSJPLEN**
Length of the IAZSSJP (SSJPSIZE) control block

**SSJPVER**
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.

**SSJPREQ**
Function to be performed on this request. Valid functions and their related SSJPUSER area are:

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPXOD</td>
<td>IAZJPLEX JESPLEX Information service, obtain data</td>
</tr>
<tr>
<td>SSJPXRS</td>
<td>IAZJPLEX JESPLEX Information service, release storage</td>
</tr>
</tbody>
</table>

**SSJPUSER**
Pointer to service specific data area ‘(IAZJPLEX)’

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

**JESPLEX Information service, IAZJPLEX contents:** For the JESPLEX Information service (function code SSJPXOD), the caller must set the following fields in the IAZJPLEX 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 IAZJPLEX (JPLXSZE) control block</td>
</tr>
<tr>
<td>JPLXVER</td>
<td>Input version of the IAZJPLEX control block. Set to JPLXSVR# for the current (latest) version.</td>
</tr>
<tr>
<td>JPLXSTRP</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 IAZJPLEX is used and from that point on the field will be managed by the subsystem.</td>
</tr>
</tbody>
</table>

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></td>
<td></td>
</tr>
</tbody>
</table>
SSI Function Code 82

**JPLXFLTR**
Flag byte for filtering results. Each bit corresponds to a filter condition that must be matched before data is returned.

<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>JPLXFMNM</td>
<td>Filter on the JES member name specified in the field JPLXMNAM</td>
</tr>
</tbody>
</table>

**JPLXSTS1**
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.

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>JPLOUDEF</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>
</tbody>
</table>

**JPLXSPEC**
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.

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPLXINDP</td>
<td>Return JESPLEX information only if the system or member is Independent (JES2 Only)</td>
</tr>
<tr>
<td>JPLXBOSS</td>
<td>Return JESPLEX information only if the system or member is BOSS (JES2 Only)</td>
</tr>
<tr>
<td>JPLXPRIM</td>
<td>Return JESPLEX information only if the system or member is the Primary Subsystem (JES2 Only)</td>
</tr>
</tbody>
</table>
SSI Function Code 82

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:

<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 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.
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
SSI Function Code 82

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.

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>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>
</tbody>
</table>
SSJPSMLE (16)
Length field in the control block pointed to by SSJPUSER is too small.

SSJPEYEE (20)
Eyecatcher in the control block pointed to by SSJPUSER is not correct.

SSJPGETM (128)
$GETMAIN failed.

SSJPSTGO (132)
STORAGE OBTAIN failed.

SSJPINVA (136)
Invalid filter arguments were specified.

SSJPGLBL (140)
Function not supported on the global (JES3 only).

JPLXINVA (132)
Invalid search arguments

JPLXINV1 (136)
Status filter invalid

JPLXINV2 (140)
System name filter invalid

JPLXINV3 (144)
Member name filter invalid

JPLXINV4 (148)
JES Specific filter invalid

**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 X'0200')</td>
</tr>
<tr>
<td>JPLXLPTTR</td>
<td>Pointer to first Member data buffer</td>
</tr>
<tr>
<td>JPLXNMBR</td>
<td>Number of Member data buffers 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>
<tr>
<td>JPLXTRAT</td>
<td>Total number of active SPOOL tracks defined across all partitions.</td>
</tr>
<tr>
<td>JPLXTRAU</td>
<td>Total number of active 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 JPLXLPTTR. Each element is composed of the following sections:
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.

**JESPLEX Info Section**

```
+----------------+                  +----------------+
| JPXHDR | JPXNXTP =-----------> | JPXHDR | JPXNXTP =----------->
|        | JPXHDR IN THE CHAIN.  |        | JPXHDR IN THE CHAIN.
+----------------+                  |        | ZERO IF END OF CHAIN.
| JPXPREF | Prefix Section |        |...
+----------------+                  +----------------+
| JPXGENI | General JESPLEX Data |        |...
|        | Section |        |...
+----------------+                  +----------------+
| JPXJES3I | Optional JES3 JESPLEX |        |...
|        | Info Section |        |...
+----------------+                  +----------------+
| JPXJES2I | Optional JES2 JESPLEX |        |...
|        | Info Section |        |...
+----------------+                  +----------------+
| JPXCPREF | Command prefix array section |        |...
+----------------+                  +----------------+
```

**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:
SSI Function Code 82

Field Name
   Description

JPXEYE
   Eyecatcher. Should be set to ‘JPXHDR’.

JPXOPRF
   Offset to the prefix section

JPXNXTP
   Address of the next JESPLEX information element

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:

Field Name
   Description

JPXPRLN
   Length of the entire element, not including the JESPLEX header section

JPXPRTP
   Type of this section.

JPXPRMD
   Modifier for this section.

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 (JPXPRSZ) to the header (JPXHDR) address.

The fields in the JPXGENI section are:

Field Name
   Description

JPXGLN
   Length of this section.

JPXGTY
   Type of this section

JPXGMD
   Modifier for this section

JPXSBSNM
   Subsystem name

JPXSTIME
   Last start date and time in STCK format

JPXSTAT1
   Member status

Field Value
   Description

JPXDRAIN
   Drained or Down member
SSI Function Code 82

JPXINITZ
Initializing (JES2 Only)

JPXACTIV
Active member

JPXDRING
Draining member (JES2 Only)

JPXNATCH
Not Attached (JES3 Only)

JPXSTATC
Character string representation of the member status

JPXMVSNM
MVS system name

JPXMEMNM
JES member name

JPXVERSN
Product version in character format

JPXSMFID
SMF identifier

JPXSYSLG
Syslog indicator flag

Field Value Description
JPXSLOGY
Release 11 syslog support is Active for this member

JPXMEMNO
Member number (JES2 Only)

JPXSTPE
Type of last start

Field Value Description
JPXCOLD
Cold start

JPX2COLF
Cold start with format (JES2 Only)

JPXWARM
Warm start

JPX2SRMS
Single member warm start (JES2 Only)

JPX3WRMD
Warm start to replace a spool dataset (JES3 Only)

JPX3WRMA
Warm start with analysis (JES3 Only)

JPX3WDA
Warm start to replace a spool dataset with analysis (JES3 Only)
SSI Function Code 82

JPXHOT
  Hot start

JPX3HOTR
  Hot start with refresh (JES3 Only)

JPX3HOTA
  Hot start with analysis (JES3 Only)

JPX3HTRA
  Hot start with refresh and analysis (JES3 Only)

JPX2QUICK
  Quick start (JES2 Only)

JPX3LCL
  Local start (JES3 Only)

JPXPRODL
  Product level in binary format

JPXSERVL
  Service level in binary format

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>
</tbody>
</table>
SSI Function Code 82

JPX3FIX
Fixed PUFs

JPX3USR
User pages per open SYSOUT dataset

JPX3SELM
Selection mode constant

JPX3SP1
Spool partition name

JPX3MPFX
Message prefix

JPX3MDST
Message destination

JPX3FLG1
Flag byte

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX3GBL</td>
<td>Global node</td>
</tr>
<tr>
<td>JPX3ONL</td>
<td>Online</td>
</tr>
<tr>
<td>JPX3FLSH</td>
<td>Flushed</td>
</tr>
<tr>
<td>JPX3CNN</td>
<td>Connected</td>
</tr>
<tr>
<td>JPX3NCNN</td>
<td>Not connected</td>
</tr>
<tr>
<td>JPX3DOWN</td>
<td>Down</td>
</tr>
<tr>
<td>JPX3ATT</td>
<td>Attached</td>
</tr>
<tr>
<td>JPX3NATT</td>
<td>Not attached</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>
</tbody>
</table>
### SSI Function Code 82

**JPX2MD**
Modifier for this section

**JPX2FLG1**
JES2 indicators

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX21IND</td>
<td>Independent mode</td>
</tr>
<tr>
<td>JPX21BOS</td>
<td>BOSS indicator</td>
</tr>
<tr>
<td>JPX21PRI</td>
<td>Primary subsystem indicator</td>
</tr>
</tbody>
</table>

**JPX2ITIM**
Time of last checkpoint access

**JPX2FLG2**
Current command being processed indicator

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX21P</td>
<td>$P command</td>
</tr>
<tr>
<td>JPX21PXQ</td>
<td>$PXEQ command</td>
</tr>
</tbody>
</table>

**JPX2HOLD**
Current setting for MASDEF HOLD

**JPX2MIND**
Current setting for MASDEF MIN DORMANCY

**JPX2MAXD**
Current setting for MASDEF MAX DORMANCY

**JPX2SYNC**
Current setting for MASDEF SYNCTOL

**JPX2AHLD**
Actual HOLD value on the last checkpoint

**JPX2ADRM**
Actual DORMANCY value on the last checkpoint

**JPX2RSID**
Name of the member doing the reset

**JPX2STAT**
Specific Status Indicator

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX2DOWN</td>
<td>DOWN</td>
</tr>
<tr>
<td>JPX2DEF</td>
<td>DEFINED</td>
</tr>
</tbody>
</table>
SSI Function Code 82

JPX2INU
INUSE

JPX2FAIL
FAILED

JPX2UNDF
Member UNDEFINED

JPX2UPND
Member UNDEFINED-PENDING

JPX2ACTV
Member ACTIVE

JPX2INAC
Member TERMINATED

JPX2INIT
Member INITIALIZING

JPX2TERM
Member TERMINATING

JPX2JESF
Member JES2-FAILED

JPX2XCFF
Member JESXCF-FAILED

JPX2MVSG
Member MVS-GONE

JPX2DORM
Member DORMANT

JPX2DRAN
Member DRAINED

JPX2ALIC
Member awaiting ALICE processing

**JPX2STA2**
Second status byte

**Bit Value**
**Description**

JPX2EDEL
Member deleted

JPX2$IND
Member in independent mode

JPX2SIOT
SPIN IOT being purged

JPX2NMAL
Member has two checkpoint data sets allocated

JPX2EGON
XCF system gone

JPX2PRIM
Member is a primary subsystem
SSI Function Code 82

**JPX2SPLX**

Command prefix has SYSplex scope

**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>JPXFPXL</td>
<td>Length of each command prefix array entry</td>
</tr>
<tr>
<td>JPXPFXO</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 (JPXOPRF), the prefix size (JPXPRSZ), the general information section size (JPXGENSZ), the sizes of any optional sections supplied, and the array offset (JPXPFXO) to the header (JPXHDR) address.

The fields in the JPXCPRFE section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXCPFXS</td>
<td>Scope Flags</td>
</tr>
<tr>
<td></td>
<td>Field Value</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>JPXCSYSP</td>
<td>SYSPLEX Scope</td>
</tr>
<tr>
<td>JPXCSYST</td>
<td>System Scope</td>
</tr>
<tr>
<td>JPXCPFXP</td>
<td>Command prefix value</td>
</tr>
</tbody>
</table>
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"
- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment"
- "Input Register Information" on page 325
- "Input Parameters" on page 325
- "Output Register Information" on page 328
- "Return Code Information" on page 328
- "Output Parameters" on page 329

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 SSJPUSER.

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
- IEFSSIB
- IAZSSJP
- IAZJPCLS (Job Class Information)

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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>
</tbody>
</table>
SSI Function Code 82

Caller variable: Caller value
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 shows the environment at the time of the call for SSI function code 82, Job Class Information Subfunction.

Figure 27. Environment at Time of 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:

**Register Contents**

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

**Input Parameters:** Input parameters for the function routine are:

- SSOB
- SSIB
SSI Function Code 82

- 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>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 82 (SSOBSSJP)</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 about the life-of-job SSIB.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (IAZSSJP 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 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>
</tbody>
</table>
SSI Function Code 82

SSJPVER
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.

SSJPFREQ
Function to be performed on this request. Valid functions and their related SSJPUSER area are:

<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>
</tbody>
</table>

SSJPUSER
Pointer to service specific data area '(IAZJPCLS)'

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>
<tr>
<td>JPCLVER</td>
<td>Input version of the IAZJPCLS control block. Set to JPCLV010 for version 1 of the control block. Set to JPCLVER# for the current (latest) version</td>
</tr>
<tr>
<td>JPCLSTRP</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 IAZJPCLS is used and from that point on the field will be managed by the subsystem.</td>
</tr>
</tbody>
</table>

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>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
</table>

SSI Function Code 82

**JPCL1CLS**
Return job class information for those job classes that match the class name indicated by JPCLCNAM.

**JPCL1GRP**
Filters on a job class group.

**JPCLGNAM**
The group name to filter on.

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.

**Return Code (Decimal)**  
**Meaning**

**SSRTOK (0)**
The Job Class 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 Class 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.
SSI Function Code 82

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>
<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>
</tbody>
</table>
SSI Function Code 82

SSJPGETM (128)
$GETMAIN failed.

SSJPSTGO (132)
STORAGE OBTAIN failed.

**Job Class Information service, IAZJPCLS contents:** For the Job Class Information service (function code SSJPCOD), 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 first Class data buffer</td>
</tr>
<tr>
<td>JPCLNCLS</td>
<td>Number of Class data buffers 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 CLSFSTM in the Job Class Header section. Each member element is composed of the following sections:
SSI Function Code 82

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

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. |
| | CLSFRRSTM =--------+ ZERO IF END OF CHAIN. |
| | +----------------+ |
| CLSPREF | Prefix Section |
| +----------------+ |
| CLSGENI | General |
| | Class Info |
| | Section |
| +----------------+ |
| CLSJES2I | Optional JES2 |
| | Class Info |
| | Section |
| +----------------+ |
| CLSJES3I | Optional JES3 |
| | Class Info |
| | Section |
| +----------------+ |
| CLS3TLIM | Optional JES3 |
| | TLIMIT Array |
| | Elements |
| +----------------+ |
| | NOTE: Number of elements |
| | specified in |
| | CLS3TLCT. |
| +----------------+ |
+----------------+

+----------------+
| CLASS MEMBER INFO SECTION |
+----------------+
| CLMHDR | CLMNXTM =-----------> POINTER TO THE NEXT |
+----------------+
```
**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><strong>CLSID</strong></td>
<td>Eyecatcher. Should be set to 'CLASSHDR'.</td>
</tr>
<tr>
<td><strong>CLSOPRF</strong></td>
<td>Offset to prefix section.</td>
</tr>
<tr>
<td><strong>CLSNXTP</strong></td>
<td>Address of the next Job Class Information element.</td>
</tr>
<tr>
<td><strong>CLSFRSTM</strong></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><strong>CLSPRLN</strong></td>
<td>Length of the entire element, not including the Job Class Header.</td>
</tr>
<tr>
<td><strong>CLSPRTP</strong></td>
<td>Type of this section.</td>
</tr>
<tr>
<td><strong>CLSPRMD</strong></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.
SSI Function Code 82

The fields in the CLSGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSGLN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>CLSGTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>CLSGMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>CLSGNAME</td>
<td>Class name</td>
</tr>
<tr>
<td>CLSGFLG1</td>
<td>Class flag 1</td>
</tr>
</tbody>
</table>

#### CLSGFLG1

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSG1WLM</td>
<td>Class is in WLM mode</td>
</tr>
<tr>
<td>CLSG1JRN</td>
<td>No journal option</td>
</tr>
<tr>
<td>CLSG1LRC</td>
<td>JOBRC=LASTRC was specified for this class.</td>
</tr>
<tr>
<td>CLSG1NAC</td>
<td>Job class is not active.</td>
</tr>
</tbody>
</table>

#### CLSGREST

Restart options

<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>
<tr>
<td>CLSGRPT</td>
<td>Print output, then hold the job (JES3 only)</td>
</tr>
<tr>
<td>CLSGRSTR</td>
<td>Allow warmstart to re-queue to Execution Phase</td>
</tr>
</tbody>
</table>

#### CLSGJFLG

JESLOG default settings

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSGELIG</td>
<td>Spin eligible</td>
</tr>
<tr>
<td>CLSGTIMI</td>
<td>Spin on time interval</td>
</tr>
<tr>
<td>CLSGTIMD</td>
<td>Spin on time of day</td>
</tr>
</tbody>
</table>
**SSI Function Code 82**

**CLSGLINE**
Spin upon line delta

**CLSGSUP**
Suppress

**CLSGNOSP**
No spin

**CLSGJVAL**
Spin value. This is the number of seconds in the interval if CLSGJFLG is set to CLSGTIMI. This is the number of seconds past midnight if CLSGJFLG is set to CLSGTIMD. This is the number of lines if CLSGJFLG is set to CLSGLINE.

**CLSGMAXJ**
Maximum number of concurrently executing jobs of this class (TDEPTH for JES3 if specified).

**CLSGCURJ**
Current number of concurrently executing jobs of this class.

**CLSGQSZ**
Number of jobs of this class that are eligible for execution (awaiting job selection) (JES2 only).

**CLSGHELD**
Number of jobs of this class that are held (JES2 only).

**CLSGGRP**
The job class group that the current job class is in.

**JES2 Job Class Information Section:** This section contains job class attributes that are unique to JES2.

The fields in the CLSJES2I section are:

**Field Name**
- **Description**

**CLS2LN**
Length of this section

**CLS2TY**
Type of this section

**CLS2MD**
Modifier for this section

**CLS2JBFL**
Job class flag

**Bit Value**
- **Description**

**CLS2BCH**
Batch job

**CLS2TSU**
Time sharing user

**CLS2STC**
Started task
SSI Function Code 82

**CLS2NOUT**
No output option

**CLS2TYP**
TYPRUN setting

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2HOLD</td>
<td>TYPRUN = HOLD</td>
</tr>
<tr>
<td>CLS2SCAN</td>
<td>TYPRUN = SCAN</td>
</tr>
<tr>
<td>CLS2COPY</td>
<td>TYPRUN = COPY</td>
</tr>
</tbody>
</table>

**CLS2CACT**
Accounting information

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CSWA</td>
<td>SWA above 16M line</td>
</tr>
<tr>
<td>CLS2CNUM</td>
<td>Account number required</td>
</tr>
<tr>
<td>CLS2CNAM</td>
<td>Programmer name required</td>
</tr>
<tr>
<td>CLS2CNON</td>
<td>No information required</td>
</tr>
<tr>
<td>CLS2CALL</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
SSI Function Code 82

Field Value
   Description
CLS2CEXE
   Pass the command through
CLS2CDSP
   Display and then pass the command
CLS2CVER
   Ask operator disposition
CLS2CIGN
   Ignore the command

CLS2CBLP
   Bypass label processing

Bit Value
   Description
CLS2CBLY
   Process bypass label parameter

CLS2COCG
   Operator command group

Bit Value
   Description
CLS2CGSY
   Group 1 commands (SYS)
CLS2CGIO
   Group 2 commands (I/O)
CLS2CGCO
   Group 3 commands (CONS)
CLS2CGAL
   All command groups

CLS2CJCL
   Default MSGLEVEL, JCL listing if not MSGLEVEL

CLS2CMMSG
   Allocation termination messages value of MSGLEVEL

CLS2JOPT
   Job options flag

Bit Value
   Description
CLS2NLOG
   No joblog indicator

CLS2XBMI
   XBM II job class

CLS2QHLD
   Class queue is held

CLS2XBM
   Procedure name for XBM II jobs
SSI Function Code 82

<table>
<thead>
<tr>
<th>Procedure library number</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SMF flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bit Value</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>CLS2NUSO</td>
</tr>
<tr>
<td>Do not take IEFUSO exit</td>
</tr>
<tr>
<td>CLS2NTY6</td>
</tr>
<tr>
<td>Do not produce Type 6 SMF record</td>
</tr>
<tr>
<td>CLS2NUJP</td>
</tr>
<tr>
<td>Do not take IEFUJP exit</td>
</tr>
<tr>
<td>CLS2NT26</td>
</tr>
<tr>
<td>Do not produce Type 26 SMF record</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default performance group</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Default message class, TSU and STC classes only</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Normal output disposition for JES data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bit Value</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>CLS21CDP</td>
</tr>
<tr>
<td>Conditionally purge output for jobs in this class</td>
</tr>
<tr>
<td>CLS21NOP</td>
</tr>
<tr>
<td>NORMAL OUTDISP=PURGE</td>
</tr>
<tr>
<td>CLS21NOW</td>
</tr>
<tr>
<td>NORMAL OUTDISP=WRITE</td>
</tr>
<tr>
<td>CLS21NOH</td>
</tr>
<tr>
<td>NORMAL OUTDISP=HOLD</td>
</tr>
<tr>
<td>CLS21NOK</td>
</tr>
<tr>
<td>NORMAL OUTDISP=KEEP</td>
</tr>
<tr>
<td>CLS21NOL</td>
</tr>
<tr>
<td>NORMAL OUTDISP=LEAVE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Abnormal output disposition for JES data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bit Value</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>CLS22AOP</td>
</tr>
<tr>
<td>ABNORMAL OUTDISP=PURGE</td>
</tr>
<tr>
<td>CLS22AOW</td>
</tr>
<tr>
<td>ABNORMAL OUTDISP=WRITE</td>
</tr>
<tr>
<td>CLS22AOH</td>
</tr>
<tr>
<td>ABNORMAL OUTDISP=HOLD</td>
</tr>
</tbody>
</table>
**SSI Function Code 82**

**CLS22AOK**
ABNORMAL OUTDISP=KEEP

**CLS22AOL**
ABNORMAL OUTDISP=LEAVE

**CLS2FLG3**
Processing flags

<table>
<thead>
<tr>
<th>Bit Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
</tbody>
</table>

| **CLS23SPC** |
| Special class (STC/TSU) |

| **CLS23SNV** |
| Default SCHENV (CLS2SCHE) no longer defined to WLM |

**CLS23DOK**
Duplicate job names OK for this job class

**CLS2SCHE**
Default SCHENV

**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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
</tbody>
</table>

| **CLS3LN** |
| Length of this section, including the variable length TLIMIT information. |

| **CLS3TY** |
| Type of this section |

| **CLS3MD** |
| Modifier for this section |

| **CLS3GRP** |
| Job class group name |

| **CLS3PART** |
| Spool partition name |

| **CLS3TRK1** |
| Primary track group allocation |

| **CLS3TRK2** |
| Secondary track group allocation |

| **CLS3SDEP** |
| SDEPTH setting |

| **CLS3PTY** |
| JES3 priority |

| **CLS3FLG1** |
| Flag byte |

<table>
<thead>
<tr>
<th>Bit Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
</tbody>
</table>
SSI Function Code 82

CLS31DEF
  Default class

CLS3JOPT
  Bit Value
  Description

CLS3NLOG
  Suppress JESMSG

CLS3LOG
  Log JESMSG

CLS3TLOF
  Offset to first JES3 TLIMIT entry

CLS3TLC
  JES3 TLIMIT entry count

CLS3TLSI
  Size of a JES3 TLIMIT entry

JES3 TLIMIT Entry: This section contains the JES3 TLIMIT information for the job class.

The fields in the CLS3TLIM section are:

Field Name
  Description

CLS3TCLS
  Controlling class name

CLS3TMAX
  Maximum jobs in controlling class

CLS3TCUR
  Current jobs in controlling class

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:

Field Name
  Description

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.
SSI Function Code 82

The fields in the CLMPREF section are:

Field Name  
Description

CLMPRLN  
Length of the entire element, not including the length of the Job Class Member Header.

CLMPRTP  
Type of this section.

CLMPRMD  
Modifier for this section.

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:

Field Name  
Description

CLMGLN  
Length of this section.

CLMGTY  
Type of this section.

CLMGMD  
Modifier for this section.

CLMGMNAM  
Member name

CLMGSNAM  
MVS System name

CLMGFLG1  
Flag byte

Bit Value  
Description

CLMG1ENB  
Class is enabled or active on the member

CLMG1ACT  
Member is active

CLMG1PXQ  
Class is on halted member, $PXEQ issued (JES2 only)

CLMG1DRN  
Class is on draining member (JES2 only)

CLMG1DEF  
Class is defined on member (JES3 only)

CLMGJMAX  
Maximum job count for this class on member (MDEPTH for JES3 if specified)

CLMGJCUR  
Current active job count for this class on member
JES3 Job Class Member Information Section: This section contains the member information that is unique to JES3.

The fields in the CLM3I section are:

**Field Name**
**Description**

CLM3LN
Length of this section, including the variable length MLIMIT information

CLM3TY
Type of this section

CLM3MD
Modifier for this section

CLM3SELM
Selection mode name

CLM3MLOF
Offset to first JES3 MLIMIT entry

CLM3MLCT
JES3 MLIMIT entry count

CLM3MLSI
Size of a JES3 MLIMIT entry

JES3 MLIMIT Entry: This section contains the JES3 MLIMIT information for the member.

The fields in the CLM3MLIM section are:

**Field Name**
**Description**

CLM3MCLS
Controlling class name

CLM3MMAX
Maximum jobs in controlling class

CLM3MCUR
Current jobs in controlling class

JES device information services — SSI function code 83

JES device information services (SSI function code 83) allow an application program to obtain information on devices managed by JES.

The JES device information interface has a JESPLEX scope – information on all devices managed by all members of a JESPLEX is available to an application program running on any system in a JESPLEX.

The interface has extensive filtering capabilities that allow an application program to subset device information by attribute values such as device type, name, and operational state, and by which a JESPLEX member manages the devices.

Requesting device information services processing

All data structures used by the JES device information SSI are mapped by the IAZSSJD macro. The names of data structures and fields referenced below can be found in this macro, unless otherwise indicated.
The primary data structure used for communication with the JES device information interface is a parameter list (SSJD structure). Fields in the parameter list are differentiated as input and output fields. Input fields are used by an application program to provide detailed information upon request, such as options and device selection filters. On return from the call to the JES device information SSI, output fields of the parameter list contain the information that allows access to device data returned by the interface. Actual data structures, which contain device information, are located in the storage areas managed by the interface.

Considerations for 64-bit addressable virtual storage: The JES device information service optionally supports returning device data in 64-bit addressable virtual storage (see option SSJDPOD64 in the SSJDPOPT field). A caller must be in 24 or 31-bit addressing mode when making a call to the IEFSSREQ macro. However, a caller might request that device data is returned in interface-controlled storage areas, located above the bar and only addressable through 64-bit address.

To access this data, a caller must switch to 64-bit addressing mode after the call to the IEFSSREQ macro and operate appropriately with the 64-bit pointers embedded in the data. Note that a caller running in 64-bit addressing mode can also access any data returned in 31-bit addressable storage (below the bar) using 64-bit pointers, although 64-bit mode is not required to do so.

JES device information services request types
The JES device information services support two request types (SSJDFREQ field):

- Data retrieval request (function code SSJOBTD)
  This request returns device information according to options and filters specified in the parameter list. Device information is returned in the storage, allocated and managed by the interface. Repeated requests of this type without intervening calls to release storage will add more device information to the information returned by the prior calls.

- Release storage request (function code SSJRSTG)
  This request clears results of the prior data retrieval requests and releases all storage areas managed by the interface.

Type of request: Directed SSI Call.

Use information: To use the JES property information services SSI, callers must meet the following requirements:

- Prepare Subsystem Options Block Header (SSOB).
- Allocate parameter list (SSJD) and place its address in the SSOBINDV field of SSOB.
- Set parameter list eyecatcher, length and version fields. See the description below.
- Decide a request to perform and place the code in the SSJDFREQ field.
- If necessary, fill in additional options and filters in the parameter list. See the description below.
- Invoke the IEFSSREQ macro to make a call to the interface.
- Analyze the return code and data returned by the interface.

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.
SSI Function Code 83

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
- IAZSSJD

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key</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 SSOB, SSIB, IAZSSJD control blocks can reside in 24 or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 28 on page 344 shows the environment at the time of the call for SSI function code 83, the JES device information services.
**Input register information:** Before issuing the IEFSSREQ macro to start the JES device information services, 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 JES information services routine are:
- SSOB
- SSIB
- IAZSSJD

**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 control block (SSOBHISZ)</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 83 (SSOBSSJD)</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 about the life-of-job SSIB.</td>
</tr>
</tbody>
</table>
SSOBINDV
  Address of the function-dependent area (IAZSSJD)

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

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

Field Name          Description
SSIBID              Identifier 'SSIB'
SSIBLEN             Length of the SSIB control block (SSIBSIZE)
SSIBSSNM            Subsystem name: name of the JES subsystem to which this Device
                    Information Services request is directed

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

IAZSSJD Contents: Before the first use, the caller should set all fields in the
parameter list (IAZSSJD) to binary zeroes. Also, when requesting function code
SSJPRSTG (release storage), the caller should not alter any output fields in the
parameter list (SSJD) returned on the prior calls to SSJOBDT (obtain data).
Failure to do so might result in a storage leak. On input, the caller must set the
following fields in the IAZSSJD structure (parameter list):

Field Name          Description
SSJDEYE             Eyecatcher for the parameter list (must be set to 'SSJPL')
SSJDLNG             Length of the parameter list (SSJDSIZE)
SSJDRVRM            Version of the parameter list, supplied by the caller. This is a two-byte
                    field, which combines version and modifier. Use the SSJDRVRC constant
                    to set this field to the most current version defined in the IASSSJD macro.
SSJDFREQ            Function to be performed on this request. Valid values of SSJDFREQ are:
                    Field Value          Description
SSJDOBTD            The obtain device data function returns device data in the storage
                    managed by the SSI for that purpose. This function can be called
                    as many times as needed. Each successive call will add more data
to the output, unless the SSJDPRLS option is used to release
storage.
SSJDRSTG            Release interface-managed storage. This function will release
SSI Function Code 83

storage used by the data returned by the SSI, ignoring all options and filters in the SSI parameter list.

SSJDSTRP
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 when the first time parameter list is used. And from that point, the field will be managed by the subsystem.

The JES device information services provide a number of processing options and selection filters, which impact the way the returned information is presented and allow to subset the returned device information according to a number of criteria.

If no filters are specified by the caller, all available device information 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 will be returned.

Implicit OR is performed between filters in the same filter group. For example, if SSJDFLT1 (device status filter) is set to SSJD1ACT in addition to SSJD1INA, the SSI will return data for all active devices in addition to data for all inactive devices.

Implicit AND is performed between filters in the different filter groups. For example, if SSJDFLT1 (device status filter) is set to SSJD1ACT and SSJDFLT6 is set to SSJD6NAM (device name filter), the SSI will return only those active devices that also have names, matching the name selection filter.

If a filter is not recognized or does not apply, it will not have an impact on the result of the SSI call. For example, JES3-only filters will not have impact on the SSI output from JES2.

Field Name
Description

**SSJDPOPT**
Processing options:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
</table>

**SSJDPRLS**
Release storage used by old data before returning new data.

**SSJDPD64**
Return data in 64-bit virtual private storage (above the bar). If the caller’s job is prevented by the job attributes to allocate storage above the bar (see MEMLIMIT parameter on JOB JCL statement), the interface will allocate storage in the 31-bit virtual private storage (below the bar). If this option is not selected, the data is always returned in the 31-bit virtual private storage (below the bar).

**SSJDPDMC**
Perform security label dominance check. In order for the caller to obtain device data, the caller’s security label should dominate the security label of the device. Otherwise, the device is invisible to the caller. This check is always performed for non-authorized callers. This option is only valid for JES2.

**SSJDPOP2**
Processing options (2):
SSI Function Code 83

Bit Value
Description

SSJDP2AD
Return additional data (line, logon or NETSRV device) with remote
WS and NJE connection

SSJDP2NF
Apply name filter (see SSJD6NAM) to NJE connections rather than
devices of other types

SSJDP2SD
Return all subdevices for the selected device regardless of filtering
(applies to offloads, lines and NJE connections)

SSJDFOPT
Output formatting options:

Bit Value
Description

SSJDFLIN
"Line view" data for remote workstations and NJE connections is
arranged according to lines that are used to access them. To access
"line view" data, use the SSJDLIN8/SSJDLINP pointer. Otherwise,
data is arranged according to device type or class; to access this
data, use pointers other than SSJDLIN8/SSJDLINP.

SSJDFDRM
Destination filter (see SSJD9DST) should also be checked against
remote number for remote print/punch devices and against device
number for local print/punch devices. This option is only valid for
JES2.

The SSJD1CHR and SSJDZOMO fields can be used to specify the EBCDIC wildcard
characters for selection strings that support wildcards. If SSJD1CHR and
SSJDZOMO are not specified (both set to hex zeroes), the default wildcard
characters are ? (question mark) for SSJD1CHR and * (asterisk) for SSJDZOMO. For
example, if either value is specified, both of the provided values are used even if
one value is hex zero. It is an error to specify equal values for SSJD1CHR and
SSJDZOMO, unless the equal values are hex zeroes.

SSJD1CHR
Wildcard character - matches one character.

SSJDZOMO
Wildcard character - matches 0 or more characters.

Device status filter group includes all filters in SSJDFLT1 and SSJDFLT2 (see
descriptions of device statuses).

SSJDFLT1
Filter by device status (1):

Bit Value
Description

SSJD1ACT
Select active devices

SSJD1INA
Select inactive devices
SSI Function Code 83

SSJD1HOT
Select printers with hot writer (JES3)

SSJD1DRG
Select draining devices

SSJD1DRN
Select drained devices (JES2) or offline devices (JES3)

SSJD1NRM
Composite (multi-bit) value for status filter - select devices in a "normal" state. Devices in this state are available to process work

SSJDFLT2
Filter by device status (2):

Bit Value Description
SSJD2STE Select startable devices
SSJD2STG Select starting devices
SSJD2HTD Select halted devices
SSJD2PAU Select paused devices
SSJD2HTG Select halting devices
SSJD2INT Select devices requiring intervention or attention

SSJD2PRB Composite (multi-bit) value for status filter selects devices in a "problem" state. Devices in this state might require operator attention.

SSJD2NRS Select unresponsive devices

SSJD2END Select devices with processors ended due to error (JES2)

Device type filter group includes all filters in SSJDFLT3 and SSJDFLT4:

Field Name Description
SSJDFLT3 Filter by device type (1):

Bit Value Description
SSJD3PRT Select printers
SSJD3PUN Select punches
SSI Function Code 83

SSJD3RDR
Select readers

SSJD3CON
Select consoles

SSJD3JXM
Select job transmitters

SSJD3JRC
Select job receivers

SSJD3SXM
Select SYSOUT transmitters

SSJD3SRC
Select SYSOUT receivers

SSJD3XMT
Composite device type filter–select all transmitters

SSJD3RCV
Composite device type filter–select all receivers

SSJD4LIN
Select line devices

SSJD4LGN
Select logon devices

SSJD4NSV
Select NETSRV devices

SSJD4OFL
Select OFFLOAD devices

SSJD4NJE
Select NJE connections

Device class filter group includes all filters in SSJD5LT5.

Attention: Do not use SSJD5LT5 to filter for line devices. If any combination of SSJD5LT5 bits are selected other than all off or all on, lines will not be returned.

Field Name
Description

SSJD5LT5
Filter by device class:

Bit Value
Description

SSJD5LCL
Select local devices

SSJD5RMT
Select remote devices
SSI Function Code 83

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJD5OFL</td>
<td>Select OFFLOAD devices (transmitters and receivers)</td>
</tr>
<tr>
<td>SSJD5NJE</td>
<td>Select NJE devices (transmitters and receivers)</td>
</tr>
<tr>
<td>SSJD5IFC</td>
<td>Select interface devices (logon and NETSRV devices)</td>
</tr>
</tbody>
</table>

Remaining filters are independent and are not combined in filter groups. (There is an implicit AND relationship between these filters.)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJD6NAM</td>
<td>Select by device name (see SSJDDVNM)</td>
</tr>
<tr>
<td>SSJD6DGN</td>
<td>Select by device group name (see SSJDDGNM) (JES3)</td>
</tr>
<tr>
<td>SSJD6SYS</td>
<td>Select by owning system name (see SSJDSYSN)</td>
</tr>
<tr>
<td>SSJD6MBR</td>
<td>Select by owning member name (see SSJDMBRN)</td>
</tr>
<tr>
<td>SSJD6LIN</td>
<td>Select by related line name (see SSJDLNNM) (JES2)</td>
</tr>
</tbody>
</table>

Note that if none of the system/member selection filters (SSJD6SYS and SSJD6MBR) are specified, data is only returned for the local JESPLEX member (for example, the subsystem that processes the SSI call). To see data for other JESPLEX members, some selection through these filters must be specified. For example, to obtain data for all members in the current JESPLEX, set SSJD6SYS bit on and set value of SSJDSYSN to “match all” wildcard selection. For JES3, information is always reported by the global system.

The following filters only apply to specific device types, which support the relevant attribute. Applying these filters to other devices will not impact the result. For example, if a device type filter is used to select punches, specifying JES-mode printer filter will not have any effect on the result.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJD7RWN</td>
<td>Select remote devices by remote workstation name (see SSJDRWNM)</td>
</tr>
</tbody>
</table>
SSJD7NJN
Select NJE devices by adjacent node name (see SSJDADJN) (JES2)

SSJD7NJA
Select SNA NJE connections by SNA application name (see SSJDAPNM) (JES2)

SSJD7NJK
Select TCP NJE connections by TCP socket name (see SSJDSDKNM) (JES2)

SSJD7NJB
Select remote and NJE devices connected via BSC

SSJD7NJS
Select remote and NJE devices connected via SNA

SSJD7NJT
Select remote and NJE devices connected via TCP/IP

SSJDFLT8
Filters that only apply to specific device types (2):

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJD8JES</td>
<td>Select JES mode printers</td>
</tr>
<tr>
<td>SSJD8FSS</td>
<td>Select FSS mode printers</td>
</tr>
</tbody>
</table>

SSJDFLT9
Filters that only apply to specific device types (3) - filter by attributes used in the work selection criteria in effect for a device:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJD9CLS</td>
<td>Select by output class name in the work selection criteria (see SSJDWSCL)</td>
</tr>
<tr>
<td>SSJD9FRM</td>
<td>Select by form name in the work selection criteria (see SSJDWSFM)</td>
</tr>
<tr>
<td>SSJD9JBN</td>
<td>Select by job name in the work selection criteria (see SSJDWSJN)</td>
</tr>
<tr>
<td>SSJD9DST</td>
<td>Select by destination ID in the work selection criteria (see SSJDWSDS). Also see option SSJDFDRM in SSJDOPT.</td>
</tr>
<tr>
<td>SSJD9WRT</td>
<td>Select by writer name in the work selection criteria (see SSJDWSWR)</td>
</tr>
<tr>
<td>SSJD9PRM</td>
<td>Select by processing mode in the work selection criteria (see SSJDWSPR)</td>
</tr>
</tbody>
</table>

SSJDFLTZ
Filters that only apply to specific device types (4) - filter by attributes of the work unit currently processed by a device:
SSI Function Code 83

Bit Value
Description

**SSJDZJOB**
Select by name of the job currently processed by a device (see SSJDAAJOB)

**SSJDZCRT**
Select by name of the owner of the job currently processed, or by name of creator of the SYSOUT dataset currently processed by a device (see SSJDACRT)

The following fields provide actual selection values for filters that are requested by the filter selection flags described above.

Device name filter consists of two parts:
- Single name filter (SSJDDVNM). This part supports wildcards.
- Name range list filter (SSJDDVNA and SSJDDVN#). This part does not support wildcards.

The device will pass the device name filter if it matches any of these two parts. Both parts of the device name filter are controlled by the same bit (SSJD6NAM in SSJDFLT6).

Name range list is specified by a pointer to a list (SSJDDVNA) and the number of elements in the list (SSJDDVN#). Each element in the list is a pair of 10-character device names that define a range of device names. To disable the single name filter (SSJDDVNM), set this field to blank. To disable the name range list filter, set the pointer (SSJDDVNA) or the number of elements (SSJDDVN#) to binary zero. For example, one way to select devices with the name R1.PRI and devices with the names in the range of PRT100-PRT200 is to set single name filter (SSJDDVNM) to 'R1.PRI' (this will match a single device name), and create a list with one element - a pair of values 'PRT100' and 'PRT200' (this will match a range of device names).

The function of the device name filter is independent of the device type and the device class filters. There is an implicit OR relationship between the device name filter and the type and class filters. For example, if the device name filter is set to select devices with name 'PRT5*', SSJD3PRT is set to select printers, and SSJD5RMT is set to select remote devices, the interface will return all remote printers and any other device that matches 'PRT5*'.

Field Name
Description

**SSJDDVNM**
Single device name for selection. This field supports wildcards (used with SSJD6NAM)

**SSJDDVNA**
Pointer to device name range list (used with SSJD6NAM)

**SSJDDVN#**
Number of elements in the device name range list (used with SSJD6NAM)

Device group name filter is supported by JES3 and consists of two parts:
- Single group name filter (SSJDDGNM). This part supports wildcards.
- Group name range list filter (SSJDDGNA and SSJDDGN#). This part does not support wildcards.
JES3 device group will pass the filter if it matches any of these two parts. Both parts of the device group name filter are controlled by the same bit (SSJD6DGN in SSJDFLT6).

Group name range list is specified by a pointer to a list (SSJDDGNA) and number of elements in the list (SSJDDGN#). Each element in the list is a pair of 8-character device group names that define a range of device group names.

To disable the single name part of the filter (SSJDDGNM), set this field to blank.

To disable the name range list part of the filter, set pointer (SSJDDGNA) or the number of elements (SSJDDGN#) to binary zero.

For example, to select devices with the device group name DGRP1 and devices with the device group names in the range of DGRP100-DGRP200, set a single name filter (SSJDDGNM) to 'DGRP1' (this will match a single device group), and create a list with one element - a pair of values 'DGRP100' and 'DGRP200' (this will match a range of device group names).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJDDGNM</td>
<td>Single device group name for selection. This field supports wildcards. (used with SSJD6DGN)</td>
</tr>
<tr>
<td>SSJDDGNA</td>
<td>Pointer to device group name range list. (used with SSJD6DGN)</td>
</tr>
<tr>
<td>SSJDDGN#</td>
<td>Number of elements in the device group name range list. (used with SSJD6DGN)</td>
</tr>
</tbody>
</table>

More selection values for filters:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJDSYSN</td>
<td>System name for selection (used with SSJD6SYS in SSJDFLT6). This field supports wildcards.</td>
</tr>
<tr>
<td>SSJDMBRN</td>
<td>Member name for selection (used with SSJD6MBR in SSJDFLT6). This field supports wildcards.</td>
</tr>
<tr>
<td>SSJDLNNM</td>
<td>Line name for selection (used with SSJD6LIN in SSJDFLT6). This field supports wildcards.</td>
</tr>
<tr>
<td>SSJDRWNM</td>
<td>Remote workstation name for selection (used with SSJD7RWN in SSJDFLT7). This field supports wildcards.</td>
</tr>
<tr>
<td>SSJADJN</td>
<td>Adjacent node name for selection (used with SSJD7NJA in SSJDFLT7). This field supports wildcards.</td>
</tr>
<tr>
<td>SSJDAPNM</td>
<td>SNA application name for selection (used with SSJD7NJA in SSJDFLT7). This field supports wildcards.</td>
</tr>
</tbody>
</table>
SSI Function Code 83

**SSJDSKNM**
TCP socket name for selection (used with SSJD7NJK in SSJDFLT7). This field supports wildcards.

**SSJDWSCL**
Output class name for selection (used with SSJD9CLS in SSJDFLT9). This field supports wildcards.

**SSJDWSFM**
Form name for selection (used with SSJD9FRM in SSJDFLT9). This field supports wildcards.

**SSJDWSJN**
Job name for selection (used with SSJD9JBN in SSJDFLT9). This field supports wildcards.

**SSJDWSDS**
Single destination name for selection. This field supports wildcards.

**SSJDWSPR**
Processing mode for selection (used with SSJD9PRM in SSJDFLT9). This field supports wildcards.

**SSJDWSDS**
Name of the job currently processed by device (used with SSJDZJOB in SSJDFLTZ). This field supports wildcards.

**SSJDACRT**
Owner of the job that is currently processed by device or creator of SYSOUT dataset that is currently processed by device (used with SSJDZCRM in SSJDFLTZ). This field supports wildcards.

Destination filter consists of two parts:
- Single destination name filter (SSJDWSDS). This part of the filter supports wildcards.
- Destination name list filter (SSJDDSTA and SSJDDDST#). This part of the filter does not support wildcards.

Device will pass the filter if its destination name matches any of these two parts. Both parts of the destination name filter are controlled by the same bit (SSJD9DST in SSJDFLT9).

Destination name list is specified by a pointer to a list (SSJDDSTA) and the number of elements in the list (SSJDDDST#). Each element in the list is an 18-character destination name.

To disable the single name part of the filter, set the field (SSJDWSDS) to blank.

To disable the list part of the filter, set pointer (SSJDDSTA) or the number of elements (SSJDDDST#) to binary zero.

**Field Name**
**Description**

**SSJDWSDS**
Single destination name for selection. This field supports wildcards.
### SSI Function Code 83

**SSJDDSTA**
- Pointer to destination name list.

**SSJDDST#**
- Number of elements in the destination name list.

**Output Register Information:** When control returns to the SSI 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 whether the request was processed.

#### Return Code (Decimal)

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Return Code (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The JES Device Information services request completed. Check the SSOBRETN field for function-specific return code.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the JES Device 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
- SSJPRETN
- SSJD
SSI Function Code 83

**SSOBRETN Contents:** When control returns to the caller and register 15 contains a zero, the JES Device 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>SSJDOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJDERRW (4)</td>
<td>Request completed with possible errors. Some data might be returned. See reason code in SSJDRETN.</td>
</tr>
<tr>
<td>SSJDERRU (8)</td>
<td>Request cannot be completed because of a user error. See reason code in SSJDRETN.</td>
</tr>
<tr>
<td>SSJDERRJ (12)</td>
<td>Request cannot be completed because of an internal (JES) error. See reason code in SSJDRETN.</td>
</tr>
</tbody>
</table>
| SSJDParm (16)   | Parameter list, SSOB extension, has an invalid format:
|                 | • Eyecatcher is invalid. |
|                 | • The version number is not supported by JES. |
|                 | • SSJD length is incorrect. |
| SSJDSTOR (20)   | Request cannot be processed because the required storage cannot be obtained. No data can be returned to the caller. |

**SSJDSRETN Contents:** In addition to the return code in SSOBRETN, the field SSJDSRETN contains a reason code that contains more specific information about the error. SSJDSRETN 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>SSJDFTRE (4)</td>
<td>Invalid or contradictory filter is specified.</td>
</tr>
<tr>
<td>SSJDSpTPE (8)</td>
<td>Invalid storage pointer in the parameter list.</td>
</tr>
<tr>
<td>SSJDSTRE (12)</td>
<td>No enough virtual storage to complete the request.</td>
</tr>
<tr>
<td>SSJDSUBF (16)</td>
<td>Invalid request type.</td>
</tr>
<tr>
<td>SSJDINTE (24)</td>
<td>Internal error building system information data area.</td>
</tr>
<tr>
<td>SSJDEYE (28)</td>
<td>Incorrect eyecatcher for the parameter list SSJD.</td>
</tr>
<tr>
<td>SSJDUNSD (32)</td>
<td>Invalid or unsupported version of the parameter list SSJD.</td>
</tr>
<tr>
<td>SSJDSML EQU (36)</td>
<td>SSJD Control block is wrong size</td>
</tr>
</tbody>
</table>
SSJDSMAP EQU (128)

Error with storage addressed by storage management anchor pointer. For example, not key 1, fetch protected, incorrect eyecatcher.

SSJDSTGO (132)

STORAGE OBTAIN failed.

SSJDLGLBL (136)

JES3 global system is down level. JES Device Information services (SSI 83) is not supported on the global system (JES3 only).

SSJPOST (140)

No response data received from the JES3 global system (JES3 only).

SSJDNVL (144)

Invalid response received from the JES3 global system (JES3 only).

SSJDRWIL (152)

Wildcard specification error: SSJD1CHR = SSJDZOMO, and neither of them is hex zero.

**Parameter list, SSJD contents:** The following output fields in the parameter list are set on return from the JES Device Information services:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSJDSVRM</strong></td>
<td>Subsystem version: version of the JES Device Information services, implemented by the subsystem that respond to the request. This is a two-byte field that combines version and modifier. The caller might use this information to interpret the data returned by a subsystem.</td>
</tr>
<tr>
<td><strong>SSJDLCL8</strong></td>
<td>Pointer to the first and most recent data area in a chain of data areas for local devices, managed by JES subsystem. The interface adds data to this area whenever there is a local device that passes selection filters. This is a 64-bit pointer. <strong>SSJDLCLP</strong> is the 31-bit part of this pointer. The following data areas can be found in this chain:</td>
</tr>
<tr>
<td>Data area for printer/punch devices</td>
<td>[Printer/punch data header](page 381)</td>
</tr>
<tr>
<td>Data area for reader devices</td>
<td>[Reader device data header](page 391)</td>
</tr>
<tr>
<td><strong>SSJDRMT8</strong></td>
<td>Pointer to the first and most recent data area in a chain of data areas for remote RJE workstations. The interface adds data to this area whenever there is a remote device that passes selection filters. Remote workstation data area (see description of [Remote Workstation data header](page 376) structure) provides data on RJE workstation and also serves as an anchor point for the data areas that contain data for remote devices attached to that workstation. This is a 64-bit pointer. <strong>SSJDRMTP</strong> is the 31-bit part of this pointer.</td>
</tr>
<tr>
<td><strong>SSJDNJE8</strong></td>
<td>Pointer to chain of NJE connections (see [NJE connection data header](page 422). <strong>SSJDNJEP</strong> is the 31-bit part of this pointer.</td>
</tr>
</tbody>
</table>
SSI Function Code 83

SSJDOFL8
Pointer to chain of OFFLOAD devices (see "OFFLOAD device data header (JDOHOFLD)" on page 395). SSJDOFLP is the 31-bit part of this pointer.

SSJDIFC8
Pointer to chain of interface devices (see "Logon device data header (JDGHLOGN)" on page 366 and "NETSRV device header (JDNHNSRV)" on page 368). SSJDIFCP is the 31-bit part of this pointer.

SSJDLIN8
Pointer to chain of line devices (see JDLHLINE). SSJDLINP is the 31-bit part of this pointer.

SSJDSIN8
Pointer to the first and most recent data area in a chain of system/member information data areas that provide information on JESPLEX members that pass system and member selection filters (see SSJD6SYS and SSJD6MBR in SSJDFLT6). The interface adds one such data area to the chain for each request for device data SSJDOBTD. See the description of "System information header (JDSIHDR)" on page 359 structure. This is a 64-bit pointer. SSJDSINP is the 31-bit part of this pointer.

SSJDLCL#
Number of the data areas returned in SSJDLCL8 chain. This number includes all data that are returned since the last release storage request (SSJDRSTG).

SSJDRMT#
Number of the data areas returned in SSJDRMT8 chain. This number includes all data that are returned since the last release storage request (SSJDRSTG).

SSJDRDV#
Number of the data areas returned for all remote RJE workstations in SSJDRMT8 chain. This number includes all data areas on all sub-device chains that are anchored to all JDWHRMTW data areas. This number also includes all data that are returned since the last release storage request (SSJDRSTG).

SSJDNJJE#
Number of NJE connections returned (in SSJDNJJE8/SSJDNJJE8 chain)

SSJDJDV#
Number of NJE subdevices returned (on all JDJHDEV8 chains)

SSJDOFL#
Number of OFFLOAD devices returned (in SSJDOFL8/SSJDOFLP chain)

SSJDOFDV#
Number of offload subdevices returned (on all JDOHDEV8 chains)

SSJDSRV#
Number of interface devices returned (in SSJDIFC8/SSJDIFCP chain)

SSJDLIN#
Number of line devices returned (in SSJDLIN8/SSJDLINP chain)

SSJDNDV#
Number of line subdevices returned (on all JDLHDEV8 chains)

SSJDSIN#
Number of the data areas returned in SSJDSIN8. This number includes all data that are returned since the last release storage request (SSJDRSTG).
SSJDSTRP

Storage management anchor internally used by the interface. The caller must set this field to zero before the first call to the interface. And after that this field is managed by the subsystem.

Data structures returned by the JES Device Information services:

**System/member information:** In addition to the device information, the JES Device Information services returns basic information on JESPLEX members that match the system and member selection filters (see SSJD6SYS and SSJD6MBR in SSIJDFLT6). Note that complete information on JESPLEX members is available from the JESPLEX information subfunction of the JES Property Information services ("JESPLEX Information" on page 308).

The JES Device Information services return one system/information data area in the SSJDSIN8 chain for each request to obtain device data. This data area contains an entry for each system/member that meets the selection filters. Each data area consists of the following contiguous data structures:

- System information header mapped by JDSIHDR.
- Prefix section mapped by JPSYSPRF in macro IAZJPLXI.
- System information section mapped by JPSYSINF in macro IAZJPLXI.

**System information header (JDSIHDR):** This header provides a container for system/member information data:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSIEYE</td>
<td>Eyecatcher. Set to 'JDSIHDR' by the subsystem.</td>
</tr>
<tr>
<td>JDSIOHDR</td>
<td>Offset to the first section in this container. The prefix section mapped by JPSYSPRF in macro IAZJPLXI.</td>
</tr>
<tr>
<td>JDSINEX8</td>
<td>Pointer to the next JDSIHDR in this chain. The next header in the chain is returned by the next most recent call to the SSI. This is a 64-bit pointer. JDSINEXT is the 31-bit part of this pointer.</td>
</tr>
</tbody>
</table>

Prefix section (JPSYSPRF in macro IAZJPLXI) and system information section (JPSYSINF in macro IAZJPLXI) are not unique for this interface and are described in "Output Parameters" on page 257.

**Device information:** The interface returns a number of data areas that contain various data about devices managed by the JES. There are some common considerations that apply to all or most of these data areas.

**Common fields:** Device status value:

In the context of JES information services, device status is represented as a two-byte value. Full device status may be a combination of more than one basic status bit.

Bits in the two status bytes are defined as follows:

First status byte:
SSI Function Code 83

JDST1ACT
Active—device is currently busy with processing work.

JDST1INA
Inactive—device is ready for work but is not processing any work now (JES2).

Available—(JES3).

JDST1DRG
Draining—device is active but will stop after the current unit of work (JES2).

Ending—(JES3).

JDST1DRN
Drained—device is configured but is not available (JES2).

Offline—(JES3).

JDST1ACO
Device is active but varied offline (JES3).

Second status byte:

JDST2STG
Startable—device is not ready for work but has enough resource to start.

JDST2STG
Starting—device is being started.

JDST2HTD
Halted—device is halted by the HALT command. (JES2)

JDST2PAU
Paused—device is paused by the PAUSE command. (JES2)

JDST2HTG
Halting—device is being halted. (JES2)

JDST2INT
Intervention required—device needs operator attention.

JDST2NRS
Device is not responding.

JDST2END
Ended—JES2 processor (PCE) for this device has ended because of an error (JES2).

Device type value: The following device types are supported:

JDDTPRT
Printer (local or remote)

JDDTPUN
Punch (local or remote)

JDDTCONS
Console (remote)

JDDTLOGN
Logon device

JDDTNSRV
NETSRV device
Attributes used in work selection criteria: Many JES devices support work selection based on the various attributes of the work selected for processing. The work selection list is a variable-size list of attributes that are used for work selection by this device. This list is represented in two ways in the output data, as a character string that is similar to how the appropriate JES command can display it, and as an encoded vector that is easier to process programmatically.

Each element of an encoded work selection list represents a code for a work selection attribute. The attributes in the list are ordered in the same way as they are used by the device. The actual values of the attributes are defined in the sections that report specific device attributes.

Here is a list of work selection attributes used in the encoded vector representation:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDWSCLAS</td>
<td>Job class or output class</td>
</tr>
<tr>
<td>JDWSCRTN</td>
<td>Name of the owner or creator of the unit of work</td>
</tr>
<tr>
<td>JDWSFCBN</td>
<td>Forms Control Buffer (FCB) name</td>
</tr>
<tr>
<td>JDWSFLID</td>
<td>Print flash ID</td>
</tr>
<tr>
<td>JDWSFORM</td>
<td>Output form name</td>
</tr>
<tr>
<td>JDWSJBID</td>
<td>Range of JES job ids or job numbers</td>
</tr>
<tr>
<td>JDWSJBNM</td>
<td>Name of jobs to process</td>
</tr>
<tr>
<td>JDWSOPTY</td>
<td>Output priority</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDWSPRMD
Output processing mode

JDWSUCSN
Universal Character Set (UCS) name

JDWSBRST
Burst setting for output (JES2)

JDWSHLDI
Hold indicator (JES2)

JDWSJBLM
Job size limit - in records (JES2)

JDWSMBAF
JES2 MAS member affinity (JES2)

JDWSOUTD
OUTDISP setting (JES2)

JDWSRCJ2
Route code or destination name (JES2)

JDWSSCHE
Job scheduling environment (JES2)

JDWSSLISH
“Slash”—this code separates “must have” attributes from preferences (JES2)

JDWSSOSP
SYSOUT dataset size of limit (pages) (JES2)

JDWSSOSR
SYSOUT dataset size limit (records) (JES2)

JDWSSRVC
WLM service class of a job (JES2)

JDWSSVAF
Job spool volume affinity (JES2)

JDWSUSRD
User-defined criteria (JES2)

JDWSWRTN
Writer name (JES2)

JDWSCHRS
Coded font name (CHARS) setting (JES3)

JDWSCPID
Copy modification ID (JES3)

JDWSDEVT
Device type (JES3)

JDWSRCJ3
Route code or destination ID (JES3)

JDWSTAK
Stacker setting (JES3)
Representation of variable size data: Some output data for devices have variable size and do not fit easily in a simple structure mapping. These data are physically located after the owning structure and are represented in a fixed part of a structure with a triple:

- Offset from the beginning of the structure to the first byte of variable-size data.
- Number of elements in the array.
- Length of each element of the array.

In this case, the length of the structure includes the length of the fixed part of the structure and the length of all variable-size elements belonging to that structure.

Data structures returned by the interface

After the successful call to the JES Device Information services, data areas with data for devices that pass the filters specified in the parameter list are added to chains anchored in the output fields of the parameter list.

The data area for each device is a contiguous area in storage that consists of the following elements:

- A header, which has an eyecatcher and contains all pointers used for chaining data areas together.
- A prefix section, which defines the type of data contained within this header and accounts for the length of all sections within the header. This length does not include the length of the header.
- One or more specific data sections, which contain the data that are unique for a particular device.

Each section has a section type and section type modifier. Together they uniquely identify the data section. Modifier 0 is reserved for a prefix section.

Common prefix section (JDCXPREF): This section describes the total length of data returned within a given header. Inspect the section type field to determine what kind of data are returned within this header.

In addition, the mapping for the common prefix section (JDCXPREF) can be used to access common fields at the top of all sections. The caller can navigate through the sections using these common fields and checking section types and modifiers without having to look inside data sections which are not relevant.

Field Name
   Description

JDCXLNG
   In a prefix section—the total length of all sections for this header (not including the length of the header). In all other sections—the length of that section.

JDCXTYPE
   Section type

JDCXMOD
   Section type modifier (0 for the prefix section)

JDCXDATA
   Beginning of the section-unique data (the prefix sections do not have data)

Remote workstation console data header (JDCHCONS): This section describes the header structure for remote workstation console data. The total length of the
SSI Function Code 83

data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For console devices the data (section) type is JDTYCONS. The following sections could be returned within this header:

- Prefix section (always first) [Common prefix section (JDCXPREF)” on page 363]
- Console common section [Console device common section (JDCCCONS)]
- Console JES2 section [JES2 console device section (JDC2CONS)” on page 365]
- Console JES3 section [JES3 console device section (JDC3CONS)” on page 365]

Field Name  
Description

JDCHEYE  
Eye catcher

JDCHOHDR  
Offset to first (prefix) section

JDCHJPL8  
Address of IAZJPLXI for this device

JDCHJPLX  
31-bit part of a pointer

JDCHNEX8  
Address of header of the next device (this remote)

JDCHNEXT  
31-bit part of a pointer

JDCHPAR8  
Address of parent device (remote or line)

JDCHPARN  
31-bit part of the pointer

Console device common section (JDCCCONS):  This section describes the common attributes of console devices. Console devices are uniquely identified by the combination of device type (JDCCDEVT), device class (JDCCDEVCL), device name (JDCCNAME) and name of the owning system (JDCCSYSN).

Field Name  
Description

JDCCCLNG  
Length of this section

JDCCTYPE  
Section type

JDCCMOD  
Section type modifier

JDCCDEVT  
Device type

JDCCDEVCL  
Device class

JDCCNAME  
Device name

JDCCSTAT  
Device status (see common device status flags):
SSI Function Code 83

JDCCSTA1
First status byte

JDCCSTA2
Second status byte

JDCCSYSN
Owning MVS system name

JDCCMBRN
JESPLEX member name

JDCCSECL
Security label

JDCCCSTA
Status, character value

JES2 console device section (JDC2CONS): This section describes attributes that are specific to JES2 console devices:

Field Name  Description

JDC2LNG  Length of this section

JDC2TYPE  Section type

JDC2MOD  Section type modifier

JDC2DVID  Binary device ID

JES3 console device section (JDC3CONS): This section describes attributes that are specific to JES3 console devices:

Field Name  Description

JDC3LNG  Length of this section

JDC3TYPE  Section type

JDC3MOD  Section type modifier

JDC3AUTH  Authority level (0-15)

The following variable size array of fixed-size character strings represent routing codes:

JDC3RTC0  Offset from the beginning of DSECT to the first Routing Code

JDC3RTC#  Number of elements in array

JDC3RTCL  Length of each element
SSI Function Code 83

The following variable size array of fixed-size character strings represent destination classes:

**JDC3DSTO**
Offset from the beginning of DSECT to the first Destination Class

**JDC3DST#**
Number of elements in array

**JDC3DSTL**
Length of each element

**Logon device data header (JDGHLOGN):** This section describes the header for logon device data. The total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For logon devices the data (section) type is JDTYLOGN. The following sections could be returned within this header:

- Prefix section (always first) [“Common prefix section (JDCXPREF)” on page 363]
- Logon device common section [“Logon device common section (JDGCLOGN)”]
- Logon device JES2 section [“Logon device JES2 section (JDG2LOGN)” on page 368]
- Logon device JES3 section [“Logon device JES3 section (JDG3LOGN)” on page 368]
- SNA application section (NJE) [“SNA application section (JDAPPLIC)” on page 425 attributes of local application]
- SNA application JES2 section [“SNA application JES2 section (JDA2APPL)” on page 426]

**Field Name**
Description

**JDGHEYE**
Eye catcher

**JDGHOHDR**
Offset to first (prefix) section

**JDGHJPL8**
Address of IAZJPLXI for this device

**JDGHJPLX**
31-bit part of the pointer

**JDGHNEX8**
Address of header of the next device

**JDGHNEXT**
31-bit part of the pointer

**JDGHPAR8**
Address of parent device (remote, NJE connection or none)

**JDGHPARN**
31-bit part of a pointer

**Logon device common section (JDGCLOGN):** This section describes the logon device common attributes. Logon devices are uniquely identified by the combination of device type (JDGCDEV'T), device class (JDGCDEV'C), device name (JDGCNAME) and name of the owning system (JDGCSYSN).
Field Name
Description
JDGCLNG
Length of this section
JDGCTYPE
Section type
JDGCMOD
Section type modifier
JDGDEVT
Device type
JDGCDEVC
Device class
JDGCNAME
Device name
JDGCAPPL
SNA application name
JDGCSTAT
Device status (see common device status flags):
  JDGCSTA1
  First status byte
  JDGCSTA2
  Second status byte
JDGCSYSN
Owning MVS system name
JDGCMBRN
JESPLEX member name
JDGCSECL
Security label
JDGCFLAG
Processing flags:
  Bit  Value  Description
    0  JDGCFERR  Device error (not available)
    1  JDGCFPWD  Device password set
    2  JDGCFAUT  Auto restart
    3  JDGCFTRC  Device trace requested
    4  JDGCFLOG  Device activity is logged (JES2)
JDGCRINT
Auto restart interval (minutes)
SSI Function Code 83

**JDGCRETR**
Maximum number of restart retries (0 - indefinite retry)

**JDGCCSTA**
Status, character value

**Logon device JES2 section (JDG2LOGN):** Logon device JES2 section. Contains attributes specific for logons managed by JES2.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDG2LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDG2TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDG2MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDG2DVID</td>
<td>Binary device ID</td>
</tr>
</tbody>
</table>

**Logon device JES3 section (JDG3LOGN):** Logon device JES3 section. Contains attributes of logon device, which are specific to JES3.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDG3LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDG3TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDG3MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDG3DSPJ</td>
<td>JES3 DSP job ID</td>
</tr>
<tr>
<td>JDG3SNLM</td>
<td>Session limit</td>
</tr>
</tbody>
</table>

**NETSRV device header (JDNHNSRV):** Header for NETSRV device data. Total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For NETSRV devices the data (section) type is JDTYNSRV. The following sections could be returned within this header:

- Prefix section (always first) "Common prefix section (JDCXPREF)" on page 363
- NETSRV common section "NETSRV common section (JDNCNSRV)" on page 369
- NETSRV JES2 section "NETSRV device JES2 section (JDN2NSRV)" on page 370
- NETSRV JES3 section "NETSRV JES3 section (JDN3NSRV)" on page 370
- TCP socket data section "TCP socket section (JDSKSOCK)" on page 426
- TCP socket JES2 section "TCP socket JES2 section (JDK2SOCK)" on page 427
SSI Function Code 83

JDNHEYE
Eye catcher

JDNHOHDR
Offset to first (prefix) section

JDNHJPL8
Address of IAZJPLXI for this device

JDNHJPLX
31-bit part of the pointer

JDNHNEX8
Address of header of the next device

JDNHNEXT
31-bit part of the pointer

JDNHPAR8
Address of parent device (NJE connection or 0)

JDNHPARN
31-bit part of the pointer

NETSRV common section (JDNCNSRV): NETSRV common section. Contains common attributes of NETSRV device. NETSRV devices are uniquely identified by the combination of device type (JDNCDEVT), device class (JDNCDEVC), device name (JDNCNAME) and name of the owning system (JDNCSYSN).

Field Name
Description

JDNCLNG
Length of this section

JDNCTYPE
Section type

JDNCMOD
Section type modifier

JDNCDEVT
Device type

JDNCDEVC
Device class

JDNCNAME
Device name

JDNCSKNM
Local socket name

JDNCSTAT
Device status (see common device status flags):

JDNCSTA1
First status byte

JDNCSTA2
Second status byte

JDNCSYSN
Owning MVS system name
SSI Function Code 83

**JDNCMBRN**
JESPLEX member name

**JDNCSECL**
Security label

**JDNCFLG1**
Processing flags:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDNC1AUT</td>
<td></td>
<td>Auto restart</td>
</tr>
<tr>
<td>JDNC1TRB</td>
<td></td>
<td>Basic trace requested</td>
</tr>
<tr>
<td>JDNC1TCM</td>
<td></td>
<td>Common code trace requested</td>
</tr>
<tr>
<td>JDNC1TEX</td>
<td></td>
<td>Extended trace requested</td>
</tr>
</tbody>
</table>

**JDNCASID**
NETSRV address space ID

**JDCSTAK**
TCP/IP stack name

**JDNCNSVJ**
NETSRV job ID

**JDNCREFR**
Auto restart interval (minutes)

**JDNCRTR**
Maximum number of restart retries (0 - indefinite retry)

**JDNCCTSTA**
Status, character value

**NETSRV device JES2 section (JDN2NSRV):** The NETSRV device JES2 section contains attributes of NETSRV device, which are specific to JES2.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDN2LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDN2TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDN2MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDN2DVID</td>
<td>Binary device ID</td>
</tr>
</tbody>
</table>

**NETSRV JES3 section (JDN3NSRV):** The NETSRV JES3 section contains attributes of NETSRV device, which are specific to JES3.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>

SSI Function Code 83

JDN3LNG
Length of this section

JDN3TYPE
Section type

JDN3MOD
Section type modifier

JDN3DSPJ
JES3 DSP job ID

**Line device data header (JDLHLINE):** The header for line device data returns data for the line devices with various communication protocols. The total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For line devices the data (section) type is JDTYLINE. The following sections could be returned within this header:

- Prefix section (always first) “Common prefix section (JDCXPREF)” on page 363
- Line common section “Line device common section (JDLCLINE)” on page 372
- Line JES2 section “Line device JES2 section (JDL2LINE)” on page 374
- Line JES3 section “Line device JES3 section (JDL3LINE)” on page 375
- SNA application section (NJE over SNA) “SNA application section (JDAPPLIC)” on page 425 (attributes of a peer application)
- SNA application JES2 section “SNA application JES2 section (JDA2APPL)” on page 426
- TCP socket section (NJE over TCP/IP) “TCP socket section (JDSKSOCK)” on page 426 (attributes of a peer socket) - when “line view” was requested (SSJDFLIN)
- TCP socket JES2 section “TCP socket JES2 section (JDK2SOCK)” on page 427

In the line view (see SSJDFLIN formatting option), data for remote (RJE) and NJE devices will be chained to this header using JDLHDEV pointer. Number of devices in the chain is in JDLHDEV#. Data returned for each related device will have its own unique header structure. Depending on the type of the line, devices related to line device include:

- Job transmitters (NJE) (see “Job transmitter data header (JDXHJXMT)” on page 409)
- Job receivers (NJE) (see “Job receiver data header (JDBHJRCV)” on page 398)
- SYSOUT transmitters (NJE) (see “SYSOUT transmitter data header (JDYHSXMT)” on page 415)
- SYSOUT receivers (NJE) (see “SYSOUT receiver device data header (JDSHSRCV)” on page 403)
- Printer/punch devices (RJE) (see “Printer/punch data header (JDPHPRPU)” on page 381)
- Reader devices (RJE) (see “Reader device data header (JDRHRDR)” on page 391)
- Console devices (RJE) (see “Remote workstation console data header (JDCHCONS)” on page 363)

**Field Name**
**Description**

JDLHEYE
Eye catcher
SSI Function Code 83

**JDLHOHDR**
Offset to first (prefix) section

**JDLHDEV#**
Number of related devices in the chain (see JDLHDEV8)

**JDLHJPL8**
Address of IAZJPLXI for this device

**JDLHJPLX**
31-bit part of a pointer

**JDLHNEX8**
Address of header of the next device

**JDLHNEXT**
31-bit part of the pointer

**JDLHPAR8**
Address of parent device (remote, NJE connection or none)

**JDLHPARN**
31-bit part of the pointer

**JDLHDEV8**
Address of header of the first related device

**JDLHDEVC**
31-bit part of the pointer

**Line device common section (JDLCLINE):** The line device common section contains common attributes of line devices. Line devices are uniquely identified by the combination of device type (JDLCDEVT), device class (JDLCDEVC), device name (JDLCNAME) and name of the owning system (JDLCSYSN).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDLCLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDLCTYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDLCMOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDLCDEVT</td>
<td>Device type</td>
</tr>
<tr>
<td>JDLCDEVC</td>
<td>Device class</td>
</tr>
<tr>
<td>JDLCNAME</td>
<td>Device name</td>
</tr>
<tr>
<td>JDLCUNIT</td>
<td>Device unit name/number</td>
</tr>
<tr>
<td>JDLCSTAT</td>
<td>Device status (see common device status flags):</td>
</tr>
<tr>
<td></td>
<td>First status byte</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDLCSTA2
Second status byte

JDLCSYSN
Owning MVS system name

JDLCMBRN
JESPLEX member name

JDLCSECL
Security label

JDLCPROT
Line protocol:

JDLCPSNA
SNA

JDLCPTCP
TCP/IP

JDLCFLG1
Processing flags (1):

Bit Value
Description

JDLC1RJA
Line can be used for RJE

JDLC1NJA
Line can be used for NJE

JDLC1RJE
Line currently used for RJE

JDLC1NJE
Line currently used for NJE

JDLC1CMP
Line is capable of compression

JDLC1DPX
Full duplex (if not set - half duplex)

JDLC1PWD
Line password set

JDLC1AUT
Auto restart

JDLCFLG2
Processing flags (2):

Bit Value
Description

JDLC2AB
Use interface B for this BSC line (if not set - use interface A)

JDLC2TRP
Transparency indicator
SSI Function Code 83

**JDLC2TRB**
Basic trace requested

**JDLC2TCM**
Common code trace requested

**JDLC2TEX**
Extended trace requested

**JDLC2CNA**
Auto connect required (CONNECT=YES)

**JDLC2CNN**
Auto connect not required (CONNECT=NO) if both JDLC2CNA and JDLC2CNN are off, CONNECT=DEFAULT

**JDLCDISC**
Disconnect behavior:

- **JDLDNO**
  No disconnect

- **JDLDINT**
  Immediate disconnect (interrupt)

- **JDLDQUI**
  Disconnect after current activity is complete (quiesce)

**JDLCRINT**
Auto restart interval (minutes)

**JDLCRETR**
Maximum number of restart retries (0 - indefinite retry)

**JDLCINT**
Auto connect interval (minutes)

**JDLCSTA**
Status, character value

---

**Line device JES2 section (JDL2LINE):** The line device JES2 section contains attributes of line devices that are specific to JES2.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDL2LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDL2TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDL2MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDL2CNAM</td>
<td>Connected remote workstation name or NJE node name</td>
</tr>
<tr>
<td>JDL2NJEN</td>
<td>Associated NJE node name</td>
</tr>
<tr>
<td>JDL2REST</td>
<td>Line resistance</td>
</tr>
</tbody>
</table>

Counts of the number of transmitters/receivers. X'FF' indicates a value of DEFAULT:
SSI Function Code 83

JDL2JT#
Number of job transmitters
JDL2JR#
Number of job receivers
JDL2ST#
Number of SYSOUT transmitters
JDL2SR#
Number of SYSOUT receivers
JDL2FLAG
Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDL2FADS</td>
<td>Auto disconnect indicator</td>
</tr>
<tr>
<td>JDL2FSHR</td>
<td>Line is shared</td>
</tr>
<tr>
<td>JDL2FSPH</td>
<td>High speed line greater than 9600 bps (if not set - low speed line)</td>
</tr>
<tr>
<td>JDL2FAB</td>
<td>Use code B for this dual code BSC line (if not set - use code A)</td>
</tr>
<tr>
<td>JDL2FASC</td>
<td>Use ASCII control characters (if not set - use EBCDIC)</td>
</tr>
<tr>
<td>JDL2FLOG</td>
<td>Device activity is logged</td>
</tr>
<tr>
<td>JDL2FLEA</td>
<td>Line is leased</td>
</tr>
</tbody>
</table>

JDL2DVID
Binary device ID

Line device JES3 section (JDL3LINE):
The line device JES3 section contains attributes of line devices that are specific to JES3.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDL3LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDL3TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDL3MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDL3BPS</td>
<td>Line speed (bps)</td>
</tr>
</tbody>
</table>

Remote Workstation data: If the data retrieval request to the JES Device Information services results in the data being returned for remote devices, devices attached to remote (RJE) workstations, the interface returns remote workstation data area for each RJE workstation. This data area is chained to the SSJDRMT8 output field in the parameter list.
Remote workstation data area contains the detailed information on the RJE workstation and also serves as an anchor for data areas that represent remote devices attached to that workstation.

Remote workstation data area consists of the following data structures:

- **Header** ("Remote Workstation data header (JDWHRMTW)"
- **Common prefix section** ("Common prefix section (JDCXPREF)" on page 363): this section accounts for the total length of all other sections (header is not included). The prefix section also identifies the type of data returned in this header. For remote workstation, the data (section) type is JDTYRMTW.
- **Remote workstation common section** ("Remote workstation common section (JDWCRMTW)" on page 377)
- **Optional remote workstation BSC section** ("Remote workstation BSC section (JDWBSC)" on page 379)
- **Optional remote workstation SNA section** ("Remote workstation SNA section (JDWNSNA)" on page 378)
- **Optional remote workstation JES2 section** ("Remote workstation JES2 section (JDW2RMTW)" on page 380)

Data areas for devices attached to this remote workstation are chained to this header using JDWHDEV8 pointer in JDWHRMTW header. Data returned for each related device have their own unique header structures.

Depending on the type of the remote workstation, the following related devices can be returned for this workstation:

- Printer and punch devices (see "Printer/punch data header (JDPHPRPU)" on page 381)
- Reader devices (see "Reader device data header (JDRHRDR)" on page 391)
- Console devices (see "Remote workstation console data header (JDCHCONS)" on page 363)
- Line device (see "Line device data header (JDLHLINE)" on page 371)
- Logon device (see "Logon device data header (JDGHLOGN)" on page 366)

**Remote Workstation data header (JDWHRMTW):**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDWHEYE</td>
<td>Eyecatcher. Set to 'JDWHRMTW' by the subsystem.</td>
</tr>
<tr>
<td>JDWHOHHDR</td>
<td>Offset to the first section in this container. That is the common prefix section mapped by &quot;Common prefix section (JDCXPREF)&quot; on page 363.</td>
</tr>
<tr>
<td>JDWHNEX8</td>
<td>Address of the data header (JDWHRMTW) of the next remote workstation in the chain. This is a 64-bit pointer. JDWHNEXT is the 31-bit part of this pointer.</td>
</tr>
<tr>
<td>JDWHDEV8</td>
<td>Address of the data area for the first device attached to this workstation. This is a 64-bit pointer. JDWHDEV8 is the 31-bit part of this pointer.</td>
</tr>
</tbody>
</table>
### SSI Function Code 83

**JDWHDEV#**
Number of data areas in the chain of the attached devices (JDWHDEV8).

**JDWHJPL8**
Address of the system information entry for the JESPLEX member that owns this workstation. This entry is contained in the system/member information data area pointed to by SSJDSIN8 output field in the parameter list and is mapped by "System Information" on page 457 in IAZJPLXI macro.

This is a 64-bit pointer. JDWHJPLX is the 31-bit part of this pointer.

*Remote workstation common section (JDWCRMTW):* This section contains the attributes that are common to all remote workstations. Remote workstation is uniquely identified by the combination of the workstation name (JDWCNAME), communication protocol type (JDWCPROT), and the name of the owning system (JDWCSYSN).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDWCLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDWCTYPE</td>
<td>Section type. Set to JDTYRMTW by the subsystem.</td>
</tr>
<tr>
<td>JDWCMOD</td>
<td>Section type modifier. Set to JDMDRWCM by the subsystem.</td>
</tr>
<tr>
<td>JDWCNAME</td>
<td>Remote workstation name</td>
</tr>
<tr>
<td>JDWCSYSN</td>
<td>Owning MVS system name</td>
</tr>
<tr>
<td>JDWCMBRN</td>
<td>Owning JESPLEX member name</td>
</tr>
<tr>
<td>JDWCDEV</td>
<td>Remote workstation device type</td>
</tr>
<tr>
<td>JDWCBSTAT</td>
<td>Remote workstation status (see common device status flags)</td>
</tr>
<tr>
<td>JDWCPROT</td>
<td>Connection protocol type</td>
</tr>
<tr>
<td>JDWCPBSC</td>
<td>Connected through BSC</td>
</tr>
<tr>
<td>JDWCP5NA</td>
<td>Connected through SNA</td>
</tr>
<tr>
<td>JDWCFK</td>
<td>Processing flags:</td>
</tr>
<tr>
<td>Bit Value</td>
<td>Description</td>
</tr>
<tr>
<td>JDWCFMP</td>
<td>Compression is supported (SNA only)</td>
</tr>
<tr>
<td>JDWCFCS</td>
<td>Workstation has the console</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDWCFMSG
Messages will be printed if the console is not available

JDWCFPWD
Password set indicator

JDWCLINE
Associated line device name

JDWCBUFS
Buffer size (in bytes)

JDWCDSCI
Disconnect interval (in seconds)

JDWCRTC
Route code

JDWCCRTC
Console route code

JDWCWTIM
Wait time (in seconds)

JDWCPRTR
Number of attached printers

JDWCPUNR
Number of attached punches

JDWCRDRR
Number of attached readers

JDWCCSTA
Remote workstation status, character value

Remote workstation SNA section (JDWNSNA): This section contains the remote workstation attributes that are specific for SNA communication protocol.

Field Name
Description

JDWNLNG
Length of this section

JDWNLUNM
SNA LU name

JDWNLOGN
Logon device name

JDWNFLAG
Processing flags:

Bit Value
Description

JDWNFLGN
Enable automatic logon
SSI Function Code 83

JDWNFCMP
Use compaction

JDWNFMSC
Send setup request through message
(If not set - send setup request through Peripheral Data Information Record (PDIR))

Remote workstation BSC section (JDWBSC): This section contains the remote workstation attributes that are specific for BSC communication protocol.

Field Name Description
JDWBLNG Length of this section
JDWBTYP Section type. Set to JDYRMTW by the subsystem.
JDWBMOD Section type modifier. Set to JDRWBS by the subsystem.

JDWBFLG1 Processing flags (1):
Bit Value Description
JDWB1BEX Buffer expansion feature
JDWB1BXA Additional buffer expansion feature
JDWB1BLK Blocked data record format
JDWB1HTB Horizontal tabs feature
JDWB1MFJ Add job name to messages
JDWB1MFT Add time stamp to messages
JDWB1MRF Multi-record feature
JDWB1MLV Multi-leaving capability

JDWBFLG2 Processing flags (2):
Bit Value Description
JDWB2VAR Variable length record format
(If not set - fixed length record format)
JDWB2TPY Text transparency feature
**SSI Function Code 83**

**JDWB2SHR**
Shared line definition (multiple workstations can use the same line definition)

*Remote workstation JES2 section (JDW2RMTW):* This section contains the remote workstation attributes that are specific for JES2.

**Field Name**
**Description**

**JDW2LNG**  
Length of this section

**JDW2TYPE**  
Section type. Set to JDTYRMTW by the subsystem.

**JDW2MOD**  
Section type modifier. Set to JDMDRWJ2 by the subsystem.

**JDW2FLAG**  
Processing flags:

**Bit Value**
**Description**

**JDW2F150**  
Send message HASP150 to this workstation and local operator

**JDW2F190**  
The type of message HASP190 is ACTION  
(if not set — the type of message HASP190 is INFO)

**Printer/punch device data:** If a data retrieval request to the JES Device Information services results in the data being returned for printer or punch devices, the interface returns one printer/punch data area for each printer or punch device.

For local printers and punches, this data area is chained to the SSJDLCL8 output field in the parameter list.

For remote printers and punches, this data area is chained to the JDWHDEV8 field in the data area (*Remote Workstation data header (JDWHRMTW)* on page 376) for the remote workstation that owns this printer or punch.

The printer/punch data area consists of the following data structures:

- Header (*Printer/punch data header (JDHPRPU)* on page 381)
- Common prefix section (*Common prefix section (JDCXPREF)* on page 363): this section accounts for the total length of all other sections (header is not included). The prefix section also identifies the type of data returned in this header. For printer/punch devices, data (section) type is JDTYPRPU.
- Printer/punch common section (*Printer/punch common section (JDPCPRPU)* on page 381)
- Printer/punch work selection section (*Printer/punch work selection section (JDPWRKSL)* on page 384)
- Optional non-impact printer section (*Non-impact printer section (JDPFPRT)* on page 387)
- Optional printer/punch JES2 section (*Printer/punch JES2 section (JDP2PRPU)* on page 388)
SSI Function Code 83

- Optional printer/punch JES3 section
- Optional remote printer section
- Optional job information section
- Optional output information section
- Optional output information JES2 section
- Optional output information JES3 section

Printer/punch data header (JDPHPRPU): The header for printer/punch device data can return data for local and remote printers and local and remote punches. The total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For printer/punch devices the data (section) type is JDTYPRPU.

Field Name | Description
---|---
JDPHEYE | Eye-catcher. Set to JDPHPRPU by the subsystem.
JDPHOHDR | Offset to the first section in this container, the common prefix section mapped by “Common prefix section (JDCXPREF)” on page 363.
JDPHJPL8 | Address of the system information entry for the JESPLEX member that owns this device. This entry is contained in the system/member information data area pointed to by SSJDSIN8 output field in the parameter list and is mapped by “System Information” on page 457 in IAZJPLXI macro.

This is a 64-bit pointer. JDPHJPLX is the 31-bit part of this pointer.

JDPHNEX8 | Address of the data area of the next device in the chain.

This is a 64-bit pointer. JDPHNEXT is the 31-bit part of this pointer.

JDPHPAR8 | Address of the data area of the parent device (remote, line or none).

This is a 64-bit pointer. JDPHPARN is the 31-bit part of this pointer.

Printer/punch common section (JDPCPRPU): This section contains the attributes that are common to all printer or punch devices. The device is uniquely identified by the combination of device type (JDPCDEVT), device class (JDPCDEVC), device name (JDPCNAME), and the name of the owning system (JDPCSYSN).

Field Name | Description
---|---
JDPCLNG | Length of this section
JDPCTYPE | Section type
SSI Function Code 83

JDPCMOD
Section type modifier

JDPCDEVT
Device type: JDDTPRT for printer devices JDDTPUN for punch devices

JDPCDEVC
Device class: JDDCLCL for local devices JDDCRMT for remote devices

JDPCNAME
Device name

JDPCUNIT
Device unit name/number

JDPCSTAT
Device status (see common device status flags):
  JDPCSTA1
  First status byte
  JDPCSTA2
  Second status byte

JDPCSYSN
Owning MVS system name

JDPCMBRN
JESPLEX member name

JDPCSECL
Security label

JDPCMOD1
Processing flags (1):
  Bit  Value         Description
  JDPC1FSS          FSS mode printer (if not set - JES mode printer)
  JDPC1EDG          Mark edge of separator page (3800 printer)
  JDPC1HTR          Honor TRC parameter on OUTPUT JCL statement (JES mode printer)
  JDPC1PAU          Pause between data sets (JES mode printer)
  JDPC1DSS          Print separator page between data sets
  JDPC1GPS          Print JESNEWS dataset between output groups
  JDPC1TRC          Trace requested
  JDPC1TRK          Read one track cell at a time from spool (if not set - read one record at a time)
## SSI Function Code 83

### JDPCMOD2
Processing flags (2):

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPC2VUC</td>
<td>Verify UCS</td>
</tr>
<tr>
<td>JDPC2SCH</td>
<td>Use current character arrangement table for separator pages (3800) (if not set - use default table)</td>
</tr>
<tr>
<td>JDPC2NIP</td>
<td>Non-impact printer</td>
</tr>
<tr>
<td>JDPC2FLU</td>
<td>For punches, add a blank card after each data set</td>
</tr>
<tr>
<td>JDPC2SP2</td>
<td>SPACE=DOUBLE override for this data set</td>
</tr>
<tr>
<td>JDPC2SP1</td>
<td>SPACE=SINGLE override for this data set</td>
</tr>
<tr>
<td>JDPC2SP3</td>
<td>SPACE=TRIPLE override for this data set</td>
</tr>
</tbody>
</table>

### JDPCCKML
Maximum number of lines in a logical page (used for checkpoint)

### JDPCCKPG
Number of pages between checkpoints

### JDPCDFCB
Default FCB name

### JDPCNEWP
Processing of skip-to-channel commands:

- **JDPCNPDF**
  - Use PRINTDEF statement (DEFAULT)

- **JDPCNP1**
  - Skip to channel 1 is treated as new page (ONE)

- **JDPCNPAL**
  - Skip to any channel is treated as new page (ALL)

### JDPCTRNS
TRANS= parameter processing:

- **JDPCTYES**
  - Always perform translation

- **JDPCTNO**
  - Never perform translation

- **JDPCTDEF**
  - Use TRANS= parameter from PRINTDEF statement

### JDPCFLID
Default flash ID

### JDPCMODF
N/I-printer modify identifier
SSI Function Code 83

JDPCMOD3
Processing flags (3)

Bit Value Description

JDPC3CKP
Checkpoints are based on page count (see JDPCCKPG)

JDPC3CKR
Checkpoints are based on record count (JES3) (see JDP3CKRC)

JDPC3CKS
Checkpoints are based on elapsed time (see JDPFCSEC)

JDPC3SUP
Halt with SETUP message between units of work

JDPCCSTA
Status, character value

The array of CHARS settings is a variable size array of fixed-size character strings that represent CHARS settings associated with the printer:

Field Name Description

JDPCCHRO
Offset from the beginning of the section to the first CHARS value

JDPCCHR#
Number of elements in array

JDPCCHRL
Length of each element

Printer/punch work selection section (JDPWRKSL): This section contains the attributes that are pertaining to work selection and work selection criteria used by this device.

Field Name Description

JDPWLNG
Length of this section

JDPWTYPE
Section type. Set to JDTYPRPU by the subsystem

JDPWMOD
Section type modifier. Set to JDMDDPPWS by the subsystem

Values for attributes used for work selection:

Field Name Description

JDPWOWNN
Name of the owner/creator of a SYSOUT dataset

JDPWFCBN
Forms Control Buffer (FCB) name

JDPWFLSH
Flash ID.
### SSI Function Code 83

Array of UCS names. This is variable size array of fixed-size character strings which represent form names.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWUCSO</td>
<td>Offset from the beginning of the section to the first UCS name</td>
</tr>
<tr>
<td>JDPWUCS#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDPWUCSL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

**JDPWFLG1**

Work selection attributes represented by bits:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPW1BRS</td>
<td>Select output with BURST=YES</td>
</tr>
<tr>
<td></td>
<td>(If not set - select output with BURST=NO)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWRCLL</td>
<td>Low limit of dataset size in records</td>
</tr>
<tr>
<td>JDPWRCLH</td>
<td>High limit of dataset size in records</td>
</tr>
<tr>
<td>JDPWPGLL</td>
<td>Low limit of dataset size in pages</td>
</tr>
<tr>
<td>JDPWPGLH</td>
<td>High limit of dataset size in pages</td>
</tr>
</tbody>
</table>

The array of output class names is a variable size array of fixed-size character strings which represent names of output classes. The class array can also be viewed as a single string, which starts JDPWCLSO bytes from JDPWRKSL and has a length of JDPWCLS#.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWCLSO</td>
<td>Offset from the beginning of the section to the first class name</td>
</tr>
<tr>
<td>JDPWCLS#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDPWCLSL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

Array of form names: This is variable size array of fixed-size character strings that represent form names used for work selection.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWFRCNO</td>
<td>Offset from the beginning of the section to the first form name</td>
</tr>
</tbody>
</table>
### SSI Function Code 83

**JDPWFRM#**
- Number of elements in array

**JDPWFRML**
- Length of each element

Array of processing mode names: This is variable size array of fixed-size character strings that represent the names of processing modes used for work selection.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWPRCO</td>
<td>Offset from the beginning of the section to the first processing mode name</td>
</tr>
<tr>
<td>JDPWPRC#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDPWPRCL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

Array of routing codes/destination IDs. This is a variable size array of fixed-size character strings that represent routing codes/destination IDs used for work selection.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWDSTO</td>
<td>Offset from the beginning of the section to the first route code/destination ID</td>
</tr>
<tr>
<td>JDPWDST#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDPWDSTL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

Work selection criteria in printable form: The work selection criteria string for this device is represented in the format that is used by appropriate JES configuration command.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWPWSO</td>
<td>Offset from the beginning of the section to the work selection string</td>
</tr>
<tr>
<td>JDPWPWSL</td>
<td>Length of the work selection string</td>
</tr>
</tbody>
</table>

Work selection criteria in encoded form: The work selection criteria for this device is encoded as an array (vector) of bytes, where the value of each byte represents some attribute used for work selection. The attributes are listed in the order of their processing for selection.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPWEWSO</td>
<td>Offset from the beginning of the section to the encoded work selection array</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDPWEWSL
Length of the work selection array

Non-impact printer section (JDPFPRT): This section contains the attributes that are specific for non-impact (N/I) printers.

Field Name
Description

JDPFLNG
Length of this section

JDPFTYPE
Section type. Set to JDTYPRPU by the subsystem.

JDPFMOD
Section type modifier. Set to JDMPPFU by the subsystem.

JDPFSSNM
Functional subsystem name

JDPFPROC
FSS procedure name

JDPFDEVN
FSS device (FSA) name

JDPFNPRO
Non-process runout (NPRO) time in seconds (if 0 - NPRO is not used)

JDPFFLAG
Processing flags:

Bit Value
Description

JDPFFTTRC
Rolling trace requested

JDPFPRS
JES preselects datasets for device

JDPFCPMK
Copy mark increment:
Possible values:

JDPFCDF
Use PRINTDEF settings (DEFAULT)

JDPFCCON
Do not increment (COPYMARK=CONSTANT)

JDPFCDS
Increment on a dataset level (COPYMARK=DATASET)

JDPFCJOB
Increment on a job level (COPYMARK=JOB)

JDPFCON
No copy marks to be used (COPYMARK=NONE)

JDPFCSEC
Checkpoint seconds (when JDPC3CKP flag is not set)

JDPFSSYS
Name of MVS system where FSA is active
Printer/punch JES2 section (JDP2PRPU): This section contains the attributes that are specific for printers and punches managed by JES2.

Field Name
Description
JDP2LNG
Length of this section
JDP2TYPE
Section type. Set to JDTYPRPU by the subsystem.
JDP2MOD
Section type modifier. Set to JMDPPJ2 by the subsystem.
JDP2FLAG
Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDP2FSEP</td>
<td>Print separator pages between data set groups</td>
</tr>
</tbody>
</table>
JDP2DVID
Binary device ID assigned by JES2

Attributes used by JES2 for work selection for this device, in addition to work selection attributes and work selection criteria found in the "Printer/punch work selection section (JDPWRKSL)” on page 384.

Field Name
Description
JDP2WFLG
Work selection flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDP2SFJR</td>
<td>Apply job number range selection to jobs</td>
</tr>
<tr>
<td>JDP2SFST</td>
<td>Apply job number range selection to started tasks (STC)</td>
</tr>
<tr>
<td>JDP2SFTS</td>
<td>Apply job number range selection to time-sharing users (TSU)</td>
</tr>
</tbody>
</table>
JDP2WJBN
Job name for work selection
JDP2WJIL
Low limit of job/STC/TSU numbers for work selection
JDP2WJIH
High limit of job/STC/TSU numbers for work selection
JDP2WRTN
Writer name for work selection

Array of spool volume names for work selection (spool volume affinity). This is a variable size array of fixed-size character strings that represent spool volume names used for work selection.
SSI Function Code 83

Field Name  Description
JDP2WVLO  Offset from the beginning of the section to the first volume name
JDP2WVL#  Number of elements in array
JDP2WVL2  Length of each element

Array of binary route codes for work selection: This is a variable size array of fixed-size elements that represent binary route codes used for work selection. Each element of this array is mapped by the "Binary route code structure (JDD2DEST)" on page 431 structure.

Field Name  Description
JDP2WRCO  Offset from the beginning of the section to the first route code
JDP2WRC#  Number of elements in array
JDP2WRC2  Length of each element

Printer/punch JES3 section (JDP3PRPU): This section contains the attributes that are specific for printers and punches managed by JES3.

Field Name  Description
JDP3LNG  Length of this section
JDP3TYPE  Section type. Set to JDTYPRPU by the subsystem.
JDP3MOD  Section type modifier. Set to JDMDPJ3 by the subsystem.
JDP3GRPN  Device group name
JDP3DEVT  Device type name
JDP3DSPJ  DSP job ID
JDP3HFLG  H/R Flags:
  Bit Value  Description
JDP3HFCB  Hold FCB option on device
JDP3HCHR  Hold CHARs option on device
### SSI Function Code 83

**JDP3HUCS**
- Hold UCS option on device

**JDP3HMOD**
- Hold CPYMOD option on device

**JDP3HFLS**
- Hold FLASH option on device

**JDP3HFRM**
- Hold FORMS option on device

**JDP3HBUR**
- Hold STACKER (BURST) option

**JDP3FLG1**
- Processing flags:
  
<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDP31DYN</td>
<td>Device can be started dynamically</td>
</tr>
<tr>
<td>JDP31OLG</td>
<td>Log operator commands in output</td>
</tr>
<tr>
<td>JDP31BPG</td>
<td>Print burst pages at the end of job</td>
</tr>
<tr>
<td>JDP31DGY</td>
<td>Device cannot process local data sets</td>
</tr>
<tr>
<td>JDP31PDC</td>
<td>PDEFAULT=CHARS</td>
</tr>
<tr>
<td>JDP31PDF</td>
<td>PDEFAULT=FCB</td>
</tr>
</tbody>
</table>

**JDP3CKRC**
- Number of records between checkpoints

**JDP3TRC**
- TRC

**JDP3CGS**
- Amount of character generation storage in type 3800 printer:
  
<table>
<thead>
<tr>
<th>Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDP3CGS1</td>
</tr>
<tr>
<td>128 characters</td>
</tr>
<tr>
<td>JDP3CGS2</td>
</tr>
<tr>
<td>256 characters</td>
</tr>
</tbody>
</table>

**JDP3CB**
- Clear print indicator:
  
<table>
<thead>
<tr>
<th>Possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDP3CBD</td>
</tr>
<tr>
<td>Clear after each data set</td>
</tr>
<tr>
<td>JDP3CBJ</td>
</tr>
<tr>
<td>Clear after each job</td>
</tr>
</tbody>
</table>
**Remote printer section (JDRPRRT):** This section contains the attributes that are specific for remote printers.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDPRLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDPRTYPE</td>
<td>Section type. Set to JDTYPRPU by the subsystem.</td>
</tr>
<tr>
<td>JDPRTYPE</td>
<td>Section type modifier. Set to JDMDPPRM by the subsystem.</td>
</tr>
<tr>
<td>JDPRCMPT</td>
<td>Compaction table name/number</td>
</tr>
<tr>
<td>JDPRECS</td>
<td>Transmission record size</td>
</tr>
<tr>
<td>JDPWIDTH</td>
<td>Print width</td>
</tr>
<tr>
<td>JDPDPT</td>
<td>Remote device type and subaddress (PRINTnn, EXCHnn or BASICnn)</td>
</tr>
<tr>
<td>JDPF</td>
<td>Processing flags:</td>
</tr>
<tr>
<td>Bit Value</td>
<td>Description</td>
</tr>
<tr>
<td>JDPFASI</td>
<td>Send print data “as is”</td>
</tr>
<tr>
<td>JDPFCMT</td>
<td>Printer has compaction capability</td>
</tr>
<tr>
<td>JDPFCMP</td>
<td>Printer has compression capability</td>
</tr>
<tr>
<td>JDPFFCB</td>
<td>JES will load FCB on this device</td>
</tr>
<tr>
<td>JDPFSSP</td>
<td>Printer has suspend/interrupt capability</td>
</tr>
<tr>
<td>JDPFCCTL</td>
<td>send carriage control</td>
</tr>
</tbody>
</table>

**Reader device data header (JDRHRDR):** Header for reader device data. Data for the following devices may be returned within this header: local and remote and internal readers. Total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For reader devices the data (section) type is JDTYRDR. The following sections could be returned within this header:

- Prefix section (always first) [Common prefix section (JDCXPREF)](#) on page 363
- Reader common section [Reader common section (JDRCRDR)](#) on page 392
## SSI Function Code 83

- Reader JES2 section
- Reader JES3 section
- Job information section

### Reader JES2 section (JDR2RDR)

Field Name | Description
--- | ---
JDRHEYE | Eye catcher
JDRHOHDR | Offset to first (prefix) section
JDRHJPL8 | Address of IAZJPLXI for this device
JDRHJPLX | 31-bit part of a pointer
JDRHNEX8 | Address of header of the next device
JDRHNEXT | 31-bit part of the pointer
JDRHPAR8 | Address of parent device (remote, line or none)
JDRHPARN | 31-bit part of the pointer

### Reader JES3 section (JDR3RDR)

Field Name | Description
--- | ---
JDRCLNG | Length of this section
JDRCTYPE | Section type
JDRCTYPE | Section type
JDRCMOD | Section type modifier
JDRCDEV1 | Device type
JDRCDEV1 | Device class: JDDCLCL for local devices JDDCRMT for remote devices
JDRCNAME | Device name
JDRCUNIT | Device unit name/number
JDRCSTAT1 | Device status (see common device status flags):
JDRCSTA1 | First status byte

### Reader common section (JDRCRDR)

Reader common section. Contains common attributes of reader device. Reader devices are uniquely identified by the combination of device type (JDRCDEV1), device class (JDRCDEV1), device name (JDRCNAME) and name of the owning system (JDRCYSN).

Field Name | Description
--- | ---
JDRCLNG | Length of this section
JDRCTYPE | Section type
JDRCTYPE | Section type
JDRCMOD | Section type modifier
JDRCDEV1 | Device type
JDRCDEV1 | Device class: JDDCLCL for local devices JDDCRMT for remote devices
JDRCNAME | Device name
JDRCUNIT | Device unit name/number
JDRCSTAT1 | Device status (see common device status flags):
JDRCSTA1 | First status byte
**SSI Function Code 83**

**JDRCSTA2**
Second status byte

**JDRCSYSN**
Owning MVS system name

**JDRCMBRN**
JESPLEX member name

**JDRCSECL**
Security label

**Progress counters**

**JDRCTJB**
Total jobs processed by this reader

**JDRCTRC**
Total number of records (card images) processed

**JDRCPRC**
Number of records (card images) processed for the current job

**JDRCDFJC**
Default job class

**JDRCDFMC**
Default message class

**JDRCCSTA**
Status, character value

*Reader JES2 section (JDR2RDR):* Reader JES2 section. Contains attributes specific for readers managed by JES2.

**Field Name**
**Description**

**JDR2LNG**
Length of this section

**JDR2TYPE**
Section type

**JDR2MOD**
Section type modifier

**JDR2PRDS**
Default print destination

**JDR2PUDS**
Default punch destination

**JDR2NODE**
Default execution node in NJE network

**JDR2PTLM**
Priority limit

**JDR2PTIN**
Priority increment

**JDR2FLAG**
Processing flags:
### Bit Value

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDR2FHLDD</td>
<td>Hold jobs when processed</td>
</tr>
<tr>
<td>JDR2FDVA</td>
<td>Authorized for device commands</td>
</tr>
<tr>
<td>JDR2FJBA</td>
<td>Authorized for job commands</td>
</tr>
<tr>
<td>JDR2FSYA</td>
<td>Authorized for system commands</td>
</tr>
<tr>
<td>JDR2FTRC</td>
<td>Trace requested</td>
</tr>
<tr>
<td>JDR2FANY</td>
<td>Default member affinity ANY</td>
</tr>
<tr>
<td>JDR2FIND</td>
<td>Default member affinity IND</td>
</tr>
</tbody>
</table>

### JDR2DVID

Binary device ID

Array of MAS member names (default member affinity) This is variable size array of fixed-size character strings which represent names of MAS members which can be used for job execution.

### JDR2MBRO

Offset from the beginning of DSECT to the first member name

### JDR2MBR#

Number of elements in array

### JDR2MBRL

Length of each element

---

**Reader JES3 section (JDR3RDR):** Reader JES3 section. Contains attributes specific for readers managed by JES3.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDR3LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDR3TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDR3MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDR3GRPN</td>
<td>Device group name</td>
</tr>
<tr>
<td>JDR3DEV</td>
<td>Device type name</td>
</tr>
<tr>
<td>JDR3DSPJ</td>
<td>DSP job ID</td>
</tr>
</tbody>
</table>

**JDR3FLAG:**

Processing flags:
Bit Value
Description

JDR3FACT
Account number required on JOB card

JDR3FPGM
Programmer name required on JOB card

JDR3FABV
SWA should be located above the line

JDR3FBLP
BLP label setting is respected (if not set - BLP setting is ignored)

JDR3DPTY
Default job priority

JDR3JLVL
Default job message level

JDR3ALVL
Default allocation message level

JDR3TIML
Default time limit for a job step in seconds (144000 - no limit)

JDR3REGL
Default region size in KBytes

OFFLOAD device data header (JDOHOFLD): Header for OFFLOAD device data. Total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For OFFLOAD devices the data (section) type is JDTYOFLD. The following sections could be returned within this header:

- Prefix section (always first) "Common prefix section (JDCXPREF)" on page 363
- OFFLOAD device common section "OFFLOAD device common section (JDOCOFLD)" on page 396
- OFFLOAD device JES2 section "OFFLOAD device JES2 section (JDO2OFLD)" on page 397

Data for devices related to that OFFLOAD device is chained to this header using JDOHDEVVC pointer. Number of devices in the chain is in JDOHDEV#. Data returned for each related device will have its own unique header structure. Devices related to OFFLOAD device include:

- Job transmitters (see "Job transmitter data header (JDXHJXMT)" on page 409)
- Job receivers (see "Job receiver data header (JDBHJRCV)" on page 398)
- SYSOUT transmitters (see "SYSOUT transmitter data header (JDYHSXMT)" on page 415)
- SYSOUT receivers (see "SYSOUT receiver device data header (JDSHSRCV)" on page 403)

Field Name
Description

JDOHEYE
Eye catcher

JDOHOHDR
Offset to first (prefix) section
SSI Function Code 83

JDOHDEV#
Number of related devices in the chain (see JDOHDEV8)

JDOHJPL8
Address of IAZJPLXI for this device

JDOHJPLX
31-bit part of a pointer

JDOHNEX8
Address of header of the next device

JDOHNEXT
31-bit part of the pointer

JDOHDEV8
Address of header of the first related device

JDOHDEVC
31-bit part of the pointer

OFFLOAD device common section (JDOCOFLD): OFFLOAD device common section. Contains common attributes of OFFLOAD device. OFFLOAD devices are uniquely identified by the combination of device type (JDOCDEVT), device class (JDOCDEVC), device name (JDOCNAME) and name of the owning system (JDOCSYSN).

Field Name
Description

JDOCLNG
Length of this section

JDOCTYPE
Section type

JDOCMOD
Section type modifier

JDOCDEVT
Device type

JDOCDEVC
Device class

JDOCNAME
Device name

JDOCUNIT
Device unit name/number or type

JDOCSTAT
Device status (see common device status flags):

JDOCSTA1
First status byte

JDOCSTA2
Second status byte

JDOCSYSN
Owning MVS system name

JDOCMBRN
JESPLEX member name
SSI Function Code 83

JDOCSECL
  Security label

JDOCCSTA
  Status, character value

OFFLOAD device JES2 section (JDO2OFLD): OFFLOAD device JES2 section. Contains attributes of OFFLOAD device, which are specific to JES2.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDO2LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDO2TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDO2MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDO2NRUN</td>
<td>Number of units to use</td>
</tr>
<tr>
<td>JDO2NRVL</td>
<td>Number of volumes to use</td>
</tr>
<tr>
<td>JDO2RETD</td>
<td>Retention period (days)</td>
</tr>
<tr>
<td>JDO2DSN</td>
<td>Dataset name</td>
</tr>
</tbody>
</table>

JDO2FLG1
  Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDO21XMT</td>
<td>Started as transmitter</td>
</tr>
<tr>
<td>JDO21RCV</td>
<td>Started as receiver</td>
</tr>
</tbody>
</table>

JDO2FLG2
  More processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDO22ARC</td>
<td>ARCHIVE=ALL (if not set - ARCHIVE=ONE)</td>
</tr>
<tr>
<td>JDO22CRT</td>
<td>Preserve creation time (if not set assign new creation time after restore)</td>
</tr>
<tr>
<td>JDO22SAF</td>
<td>Protect via SAF</td>
</tr>
<tr>
<td>JDO22TRC</td>
<td>Trace requested</td>
</tr>
<tr>
<td>JDO22VAL</td>
<td>Validate logical record length</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDO2TLAB
Tape label processing type:

JDO2TNL
label=NL

JDO2TSL
label=SL

JDO2TNSL
label=NSL

JDO2TSUL
label=SUL

JDO2TBLP
label=BLP

JDO2TAL
label=AL

JDO2TAUL
label=AUL

JDO2DVID
Binary device ID

Job receiver data header (JDBHJRCV): Header for job receiver data. Data for the following devices may be returned within this header:

- NJE job receivers
- OFFLOAD job receivers

The total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For job receiver devices the data (section) type is JDTYJBR.

The following sections could be returned within this header:

- Prefix section (always first) "Common prefix section (JDCXREF)" on page 363
- Job receiver common section "Job receiver common section (JDBCJRCV)" on page 393
- Job receiver JES2 section "Job receiver device JES2 section (JDBJ2RCV)" on page 403
- Job receiver OFFLOAD section "Job receiver OFFLOAD section (JDBOJRCV)" on page 400
- Job information section "Common sections" on page 425

Field Name
Description

JDBHEYE
Eye catcher

JDBHOHDR
Offset to first (prefix) section

JDBHJPL8
Address of IAZJPLXI for this device

JDBHJPLX
31-bit part of a pointer
SSI Function Code 83

JDBHNEX8
Address of header of the next device

JDBHNEXT
31-bit part of the pointer

JDBHPAR8
Address of parent device (offload, line or NJE connection)

JDBHPARN
31-bit part of the pointer

Job receiver common section (JDBCJRCV): Contains common attributes of job receiver device. Job receiver devices are uniquely identified by the combination of device type (JDBCDEVT), device class (JDBCDEVC), device name (JDBCNAME) and name of the owning system (JDBCSYSN).

Field Name | Description
---|---
JDBCLNG | Length of this section
JDBCTYPE | Section type
JDBCMOD | Section type modifier
JDBCDEVT | Device type
JDBCDEVC | Device class: JDDCNJE for NJE devices JDDCOFLD for OFFLOAD devices
JDBCNAME | Device name
JDBCSTAT | Device status (see common device status flags):
JDBCSTA1 | First status byte
JDBCSTA2 | Second status byte
JDBCSYSN | Owning MVS system name
JDBCMBRN | JESPLEX member name
JDBCSECL | Security label
JDBCFLG1 | Processing flags:
Bit Value | Description
JDBC1HLD | hold received jobs (HOLD=YES)
JDBCIRLS
release received jobs (HOLD=NO) if neither bit set, status is not
changed (HOLD=NONE)

JDBCCSTA
Status, character value

Job receiver OFFLOAD section (JDBOJRCV): Job receiver OFFLOAD section.
Contains attributes specific for JES2 OFFLOAD job receivers.

JDBOLNG
Length of this section

JDBOTYPE
Section type

JDBOMOD
Section type modifier

JDBOFLG1
Processing flags:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDB01NFY</td>
<td></td>
<td>send notification message to TSO userid as requested</td>
</tr>
<tr>
<td>JDB01STR</td>
<td></td>
<td>start this receiver when OFFLOAD device is started</td>
</tr>
<tr>
<td>JDBOEANY</td>
<td></td>
<td>Job execution member affinity is ANY</td>
</tr>
</tbody>
</table>

Modification settings - job attributes will be changed in a specified way when job
is successfully received.

Field Name
Description

JDBOMJBC
New job class

JDBOMROU
New route code/destination

Array of member MAS names (new member affinity). This is variable size array of
fixed-size character strings which represent names of MAS members which can be
used for job execution.

Note: Check the JDBOEANY bit first before using these fields.

Field Name
Description

JDBOMMBO
Offset from the beginning of DSECT to the first member name

JDBOMMB#
Number of elements in array

JDBOMMBL
Length of each element
Values for attributes used for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWOWN</td>
<td>Name of job owner</td>
</tr>
<tr>
<td>JDBOWJBN</td>
<td>Job name</td>
</tr>
<tr>
<td>JDBOWSVN</td>
<td>Service class name</td>
</tr>
<tr>
<td>JDBOWSCH</td>
<td>Scheduling environment</td>
</tr>
</tbody>
</table>

JDBOWFGLG

Work selection flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWHLD</td>
<td>Select held (HOLD=YES)</td>
</tr>
<tr>
<td>JDBOWRLS</td>
<td>Select non held (HOLD=NO) If neither bit set, select none (HOLD=NONE)</td>
</tr>
<tr>
<td>JDBOWANY</td>
<td>Work selection member affinity is ANY (also see section starting with field JDBOWMBO)</td>
</tr>
<tr>
<td>JDBOWJOB</td>
<td>Job ID range is for JOB</td>
</tr>
<tr>
<td>JDBOWSTC</td>
<td>Job ID range is for STC</td>
</tr>
<tr>
<td>JDBOWTSU</td>
<td>Job ID range is for TSU</td>
</tr>
</tbody>
</table>

Job ID range for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWJIL</td>
<td>Job ID low limit</td>
</tr>
<tr>
<td>JDBOWJIH</td>
<td>Job ID high limit</td>
</tr>
</tbody>
</table>

Array of job classes. This is a variable size array of fixed-size character strings which represent names of job classes. The class array can also be viewed as a single string, which starts JDBOWCLO bytes from JDBOJRCV and which length is JDBOWCL#:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWCLO</td>
<td>Offset from the beginning of DSECT to the first job class or job class group</td>
</tr>
<tr>
<td>JDBOWCL#</td>
<td>Number of elements in array</td>
</tr>
</tbody>
</table>
SSI Function Code 83

**JDBOWCLL**
Length of each element

Array of routing codes/destination IDs. This is variable size array of fixed-size character strings which represent routing codes or destination IDs.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWDSO</td>
<td>Offset from the beginning of DSECT to the first route code/dest ID</td>
</tr>
<tr>
<td>JDBOWDS#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDBOWDSL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

Array of binary route codes for work selection. This is variable size array of fixed-size structures, mapped by the JDD2DEST structure, of binary route codes used for work selection.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWRCO</td>
<td>Offset from the beginning of DSECT to the first route code</td>
</tr>
<tr>
<td>JDBOWRC#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDBOWRCL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

Array of MAS member names for work selection (member affinity). This is variable size array of fixed-size character strings which represent MAS member names used for work selection. NOTE: Check the JDBOWANY bit first before using these fields.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWMBO</td>
<td>Offset from the beginning of DSECT to the first member name</td>
</tr>
<tr>
<td>JDBOWMB#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDBOWMBL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

Work selection criteria in printable form. The work selection criteria string is represented in the format which would be used by appropriate JES configuration command.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBOWSCO</td>
<td>Offset from the beginning of DSECT to the work selection string</td>
</tr>
<tr>
<td>JDBOWSCL</td>
<td>Length of the work selection string</td>
</tr>
</tbody>
</table>
Work selection criteria in encoded form. The work selection criteria is encoded as an array of bytes, where the value of each byte represents an attribute used for work selection. (See symbol definitions for work selection attributes - JDWSxxx.)

Field Name
  Description
JDBOWSEO
  Offset from the beginning of DSECT to the work selection array
JDBOWSEL
  Length of the work selection array

Job receiver device JES2 section (JDB2JRCV): The job receiver device JES2 section contains the job receiver attributes:

Field Name
  Description
JDB2LNG
  Length of this section
DB2TYPE
  Section type
JDB2MOD
  Section type modifier
JDB2DVID
  Binary device ID

SYSOUT receiver device data header (JDSHSRCV): Header for SYSOUT receiver data. Data for the following devices may be returned within this header: - NJE SYSOUT receivers - OFFLOAD SYSOUT receivers Total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For SYSOUT receiver devices the data (section) type is JDTYSYRC. The following sections could be returned within this header:

  • Prefix section (always first) “Common prefix section (JDCXPREF)” on page 363
  • SYSOUT receiver common section “SYSOUT receiver common section (JDSCSRCV)” on page 404
  • SYSOUT receiver JES2 section “SYSOUT receiver JES2 section (JDS2SRCV)” on page 409
  • SYSOUT receiver OFFLOAD section “SYSOUT receiver OFFLOAD section (JDSOSRCV)” on page 405
  • Job information section “Common sections” on page 425
  • Output information section “Current output information section (JDUTINFO)” on page 429
  • Output information JES2 section “Current output information JES2 section (JDU2INFO)” on page 430

Field Name
  Description
JDSHEYE
  Eye catcher
JDSHOHDR
  Offset to first (prefix) section
SSI Function Code 83

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSHJPL8</td>
<td>Address of IAZJPLXI for this device</td>
</tr>
<tr>
<td>JDSHJPLX</td>
<td>31-bit part of a pointer</td>
</tr>
<tr>
<td>JDSHNEX8</td>
<td>Address of header of the next device</td>
</tr>
<tr>
<td>JDSHNEXT</td>
<td>31-bit part of the pointer</td>
</tr>
<tr>
<td>JDSHPAR8</td>
<td>Address of parent device (offload, line or NJE connection)</td>
</tr>
<tr>
<td>JDSHPARN</td>
<td>31-bit part of the pointer</td>
</tr>
</tbody>
</table>

SYSOUT receiver common section (JDSCSRCV): The SYSOUT receiver common section contains common attributes of SYSOUT receiver devices. SYSOUT receiver devices are uniquely identified by the combination of device type (JDSCDEVT), device class (JDSCDEVC), device name (JDSCNAME) and name of the owning system (JDSCSYSN).

Field Name               Description
JDSCCLNG                      Length of this section
JDSCCTYPE            Section type
JDSCCMOD                Section type modifier
JDSCDEVT               Device type
JDSCDEVC          Device class: JDDCNJE for NJE devices JDDCOFLD for OFFLOAD devices
JDSCNAME            Device name
JDSCSTAT              Device status (see common device status flags):
                      JDSCSTA1    First status byte
                      JDSCSTA2    Second status byte
JDSCSYSN              Owning MVS system name
JDSCMBRN                JESPLEX member name
JDSCSECL               Security label
JDSCCSTA               Status, character value
SYSOUT receiver OFFLOAD section (JDSOSRCV): SYSOUT receiver OFFLOAD section. Contains attributes specific for JES2 OFFLOAD SYSOUT receivers.

Field Name  
Description  

JDSOLNG  
Length of this section  

JDSOTYPE  
Section type  

JDSOMOD  
Section type modifier  

JDSOFLG1  
Processing flags:
  - Bit Value  
  - Description  
  - JDSO1NFY: send notification message to TSO userid as requested  
  - JDSO1STR: start this receiver when OFFLOAD device is started  

Modification settings - SYSOUT dataset attributes will be changed in a specified way when data set is successfully received.

Field Name  
Description  

JDSOMFCB  
New FCB name  

JDSOMFLH  
New flash ID  

JDSOMFRM  
New form name  

JDSOMPRM  
New processing mode  

JDSOMCLS  
New output class/queue  

JDSOMDST  
New route code/destination  

JDSOMUCS  
New UCS name  

JDSOMWTR  
New writer name  

JDSOFLG2  
Modification settings:
  - Bit Value  
  - Description  
  - JDSO2BRS: Set BURST=YES
SSI Function Code 83

JDSO2BRN
Set BURST=NO if neither bit set, do not change the attribute

JDSO2HLD
Hold output (HOLD=YES)

JDSO2RLS
Release output (HOLD=NO) if neither bit set, status is not changed (HOLD=NONE)

JDSO2ODH
Set OUTDISP=HOLD

JDSO2ODK
Set OUTDISP=KEEP

JDSO2ODL
Set OUTDISP=LEAVE

JDSO2ODW
Set OUTDISP=WRITE

Values for attributes used for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSOWFCB</td>
<td>FCB name</td>
</tr>
<tr>
<td>JDSOWFLH</td>
<td>Flash ID</td>
</tr>
<tr>
<td>JDSOWCRT</td>
<td>SYSOUT creator/owner name</td>
</tr>
<tr>
<td>JDSOWJBN</td>
<td>Job name</td>
</tr>
</tbody>
</table>

Job ID range for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSOWJIL</td>
<td>Job ID low limit</td>
</tr>
<tr>
<td>JDSOWJIH</td>
<td>Job ID high limit</td>
</tr>
<tr>
<td>JDSOWUCS</td>
<td>UCS name</td>
</tr>
<tr>
<td>JDSOWWTR</td>
<td>Writer name</td>
</tr>
</tbody>
</table>

JDSOFLG3
Work selection flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSO3BRS</td>
<td>Select jobs with BURST=YES (if not set - select BURST=NO)</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDSO3HLD
  Select jobs which are held (if not set - select jobs which are not held)

JDSO3ODH
  Select output with OUTDISP=HOLD

JDSO3ODK
  Select output with OUTDISP=KEEP

JDSO3ODL
  Select output with OUTDISP=LEAVE

JDSO3ODW
  Select output with OUTDISP=WRITE

JDSO3BNS
  BURST value was not set (ignore JDSO3BRS)

JDSO3HNS
  HOLD value was not set (ignore JDSO3HLD)

JDSOFLG4
  More work Selection flags:
  Bit Value
  Description

JDSO4JOB
  Job ID range is for JOB

JDSO4STC
  Job ID range is for STC

JDSO4TSU
  Job ID range is for TSU

Array of output classes
This is a variable size array of fixed-size character strings which represent names of output classes.

Field Name
Description

JDSOWCLO
  Offset from the beginning of DSECT to the first class

JDSOWCL#
  Number of elements in array

JDSOWCLL
  Length of each element

Array of form names
This is a variable size array of fixed-size character strings which represent names of output forms.

Field Name
Description

JDSOWFMO
  Offset from the beginning of DSECT to the first form name

JDSOWFM#
  Number of elements in array
**SSI Function Code 83**

### Array of processing mode names

This is a variable size array of fixed-size character strings which represent names of processing modes.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSOWPMO</td>
<td>Offset from the beginning of DSECT to the first processing mode name</td>
</tr>
<tr>
<td>JDSOWPM#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDSOWPML</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

### Array of routing codes/destination ids

This is a variable size array of fixed-size character strings which represent routing codes or destination ids.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSOWDSO</td>
<td>Offset from the beginning of DSECT to the first route code/dest ID</td>
</tr>
<tr>
<td>JDSOWDS#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDSOWDSL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

### Array of binary route codes for work selection

This is a variable size array of fixed-size structures, mapped by the JDD2DEST structure, of binary route codes used for work selection.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSOWRCO</td>
<td>Offset from the beginning of DSECT to the first route code</td>
</tr>
<tr>
<td>JDSOWRC#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDSOWRCL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

### Work selection criteria string

The work selection criteria string is represented in the format which would be used by appropriate JES configuration command.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSOWSCO</td>
<td>Offset from the beginning of DSECT to the work selection string</td>
</tr>
<tr>
<td>JDSOWSCL</td>
<td>Length of the work selection string</td>
</tr>
</tbody>
</table>
Work selection criteria in encoded form. The work selection criteria is encoded as an array of bytes, where the value of each byte represents an attribute used for work selection. (See symbol definitions for work selection attributes - JDWSxxxx.)

Field Name
Description

JDSOWSEO
Offset from the beginning of DSECT to the work selection array

JDSOWSEL
Length of the work selection array

SYSOUT receiver JES2 section (JDS2SRCV): The SYSOUT receiver device JES2 section contains attributes of SYSOUT receiver, which are specific to JES2.

Field Name
Description

JDS2LNG
Length of this section

JDS2TYPE
Section type

JDS2MOD
Section type modifier

JDS2DVID
Binary device ID

Job transmitter data header (JDXHJXMT): Header for job transmitter data. Data for the following devices may be returned within this header:
- NJE job transmitters
- OFFLOAD job transmitters

The total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For job transmitter devices the data (section) type is JDTYJBXM. The following sections could be returned within this header:
- Prefix section (always first) "Common prefix section (JDCXPREF)" on page 363
- Job transmitter common section "Job transmitter common data (JDXCJXMT)" on page 410
- Job transmitter JES2 section "Job transmitter JES2 section (JDX2JXMT)" on page 415
- Job transmitter NJE section "Job transmitter NJE section (JDXNJXMT)" on page 411
- Job transmitter OFFLOAD section "Job transmitter OFFLOAD section (JDXOJXMT)" on page 411
- Job information section "Common sections" on page 425

Field Name
Description

JDXHEYE
Eye catcher

JDXHOHDR
Offset to first (prefix) section
SSI Function Code 83

JDXHJPL8
Address of IAZJPLXI for this device

JDXHJPLX
31-bit part of the pointer

JDXHNEX8
Address of header of the next device

JDXHNEXT
31-bit part of the pointer

JDXHPAR8
Address of parent device (offload, line or NJE connection)

JDXHPARN
31-bit part of the pointer

Job transmitter common data (JDXCJXMT): Job transmitter common section. Contains common attributes of job transmitter device. Job transmitter devices are uniquely identified by the combination of device type (JDXCDEVT), device class (JDXCDEVNC), device name (JDXCNAME) and name of the owning system (JDXCSYSN).

JDXCLNG
Length of this section

JDXCTYPE
Section type

JDXCMOD
Section type modifier

JDXCDEVT
Device type

JDXCDEVNC
Device class: JDDCNJE for NJE devices and JDDCOFLD for OFFLOAD devices

JDXCNAME
Device name

JDXCSTAT
Device status (see common device status flags):

JDXCSTA1
First status byte

JDXCSTA2
Second status byte

JDXCSYSN
Owning MVS system name

JDXCMBRN
JESPLEX member name

JDXCSECL
Security label

JDXCCSTA
Status, character value
**Job transmitter NJE section (JDXNJXMT):** Job transmitter NJE section. Contains attributes specific for JES2 NJE job transmitters:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXNLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDXNTYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDXNMOD</td>
<td>Section type modifier</td>
</tr>
</tbody>
</table>

Values for attributes used for work selection. Job size range for work selection (records):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXNWJSL</td>
<td>Job size low limit</td>
</tr>
<tr>
<td>JDXNWJSH</td>
<td>Job size high limit</td>
</tr>
</tbody>
</table>

Work selection criteria in printable form. The work selection criteria string is represented in the format which would be used by appropriate JES configuration command.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXNWSCO</td>
<td>Offset from the beginning of DSECT to the work selection string</td>
</tr>
<tr>
<td>JDXNWSCL</td>
<td>Length of the work selection string</td>
</tr>
</tbody>
</table>

Work selection criteria in encoded form. The work selection criteria is encoded as an array of bytes, where the value of each byte represents an attribute used for work selection. (See symbol definitions for work selection attributes - JDWSxxx.)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXNWSEO</td>
<td>Offset from the beginning of DSECT to the work selection array</td>
</tr>
<tr>
<td>JDXNWSEL</td>
<td>Length of the work selection array</td>
</tr>
</tbody>
</table>

**Job transmitter OFFLOAD section (JDXOJXMT):** Job transmitter OFFLOAD section. Contains attributes specific for OFFLOAD job transmitters.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXOLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDXOTYPE</td>
<td>Section type</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDXOMOD
Section type modifier

JDXOFLG1
Processing flags:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXO1NFY</td>
<td>Send notification message to TSO userid as requested</td>
<td></td>
</tr>
<tr>
<td>JDXO1STR</td>
<td>Start this receiver when OFFLOAD device is started</td>
<td></td>
</tr>
</tbody>
</table>

JDXODISP
Post-offload job disposition:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXODDEL</td>
<td>DELETE</td>
</tr>
<tr>
<td>JDXODHLD</td>
<td>HOLD</td>
</tr>
<tr>
<td>JDXODKP</td>
<td>KEEP</td>
</tr>
</tbody>
</table>

Values for attributes used for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXOWOWN</td>
<td>Name of job owner</td>
</tr>
<tr>
<td>JDXOWJBN</td>
<td>Job name</td>
</tr>
<tr>
<td>JDXOWSVCN</td>
<td>Service class name</td>
</tr>
<tr>
<td>JDXOWSVC</td>
<td>Scheduling environment</td>
</tr>
</tbody>
</table>

Job ID range for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXOWJIL</td>
<td>Job ID low limit</td>
</tr>
<tr>
<td>JDXOWJIH</td>
<td>Job ID high limit</td>
</tr>
</tbody>
</table>

Job size range for work selection (records):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXOWJSL</td>
<td>Job size low limit</td>
</tr>
<tr>
<td>JDXOWJSH</td>
<td>Job size high limit</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDXOWFLG
Work selection flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDXOWHLD</td>
<td>Select held (HOLD=YES)</td>
</tr>
<tr>
<td>JDXOWRLS</td>
<td>Select non held (HOLD=NO) If neither bit set, select none (HOLD=NONE)</td>
</tr>
<tr>
<td>JDXOFANY</td>
<td>Default member affinity is ANY. (also see section starting with field JDXOWMBO).</td>
</tr>
<tr>
<td>JDXOWJOB</td>
<td>Job ID range is for JOB</td>
</tr>
<tr>
<td>JDXOWSTC</td>
<td>Job ID range is for STC</td>
</tr>
<tr>
<td>JDXOWTSU</td>
<td>Job ID range is for TSU</td>
</tr>
</tbody>
</table>

Array of job classes This is a variable size array of fixed-size character strings which represent names of job classes. The class array can also be viewed as a single string, which starts JDXOWCLO bytes from JDXOJXMT and which length is JDXOWCL#.

Field Name | Description
---|---
JDXOWCLO | Offset from the beginning of DSECT to the first job class or job class group name
JDXOWCL# | Number of elements in array
JDXOWCLL | Length of each element

Array of routing codes/destination ids This is variable size array of fixed-size character strings which represent routing codes or destination ids.

Field Name | Description
---|---
JDXOWDSO | Offset from the beginning of DSECT to the first route code/dest ID
JDXOWDS# | Number of elements in array
JDXOWDSL | Length of each element

Array of binary route codes for work selection This is variable size array of fixed-size structures, mapped by the JDD2DEST structure, of binary route codes used for work selection.
SSI Function Code 83

Field Name
Description

JDXOWRCO
Offset from the beginning of DSECT to the first route code

JDXOWRC#
Number of elements in array

JDXOWRCL
Length of each element

Array of MAS member names for work selection (member affinity) This is variable size array of fixed-size character strings which represent MAS member names used for work selection. NOTE: Check the JDXOFANY bit first before using these fields.

Field Name
Description

JDXOWMBO
Offset from the beginning of DSECT to the first MAS member name

JDXOWMB#
Number of elements in array

JDXOWMBL
Length of each element

Array of spool volume names for work selection This is variable size array of fixed-size character strings which represent spool volume names used for work selection.

Field Name
Description

JDXOWVLO
Offset from the beginning of DSECT to the first volume name

JDXOWVL#
Number of elements in array

JDXOWVLL
Length of each element

Work selection criteria in printable form. The work selection criteria string is represented in the format which would be used by appropriate JES configuration command.

Field Name
Description

JDXOWSCO
Offset from the beginning of DSECT to the work selection string

JDXOWSCL
Length of the work selection string

Work selection criteria in encoded form. The work selection criteria is encoded as an array of bytes, where the value of each byte represents an attribute used for work selection. (See symbol definitions for work selection attributes - JDWSxxx.)
SSI Function Code 83

**JDXOWSEO**
Offset from the beginning of DSECT to the work selection array

**JDXOWSEL**
Length of the work selection array

*Job transmitter JES2 section (JDX2JXMT):* Job transmitter device JES2 section
Contains attributes of job transmitter, which are specific to JES2.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDX2LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDX2TYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDX2MOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDX2DVID</td>
<td>Binary device ID</td>
</tr>
</tbody>
</table>

**SYSOUT transmitter data header (JDYHSXMT):** Header for SYSOUT transmitter data. Data for the following devices may be returned within this header: - NJE SYSOUT transmitters - OFFLOAD SYSOUT transmitters Total length of the data for a device is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For SYSOUT transmitter devices the data (section) type is JDTYSYXM. The following sections could be returned within this header:

- Prefix section (always first) "Common prefix section (JDCXPREF)" on page 363
- SYSOUT transmitter common section "SYSOUT transmitter common section (JDYCSXMT)" on page 416
- SYSOUT transmitter JES2 section "SYSOUT transmitter device JES2 section (JDY2SXMT)" on page 422
- SYSOUT transmitter NJE section "SYSOUT transmitter NJE section (JDYNSXMT)" on page 416
- SYSOUT transmitter OFFLOAD section "SYSOUT transmitter OFFLOAD section (JDYOSXMT)" on page 418
- Job information section "Common sections" on page 425

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYHEYE</td>
<td>Eye catcher</td>
</tr>
<tr>
<td>JDYHOHDR</td>
<td>Offset to first (prefix) section</td>
</tr>
<tr>
<td>JDYHJPL8</td>
<td>Address of IAZJPLXI for this device</td>
</tr>
<tr>
<td>JDYHJPLX</td>
<td>31-bit part of a pointer</td>
</tr>
<tr>
<td>JDYHNEX8</td>
<td>Address of header of the next device</td>
</tr>
</tbody>
</table>
### SSI Function Code 83

**JDYNEXT**
- 31-bit part of the pointer

**JDYHPAR8**
- Address of parent device (offload, line or NJE connection)

**JDYHPARN**
- 31-bit part of the pointer

*SYSOUT transmitter common section (JDYCSXMT):* SYSOUT transmitter common section. Contains common attributes of SYSOUT transmitter devices. SYSOUT transmitter devices are uniquely identified by the combination of device type (JDYCDEVT), device class (JDYCDEVCL), device name (JDYCNAME) and name of the owning system (JDYCSYSN).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYCLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDYCTYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDYCMOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDYCDEVCL</td>
<td>Device type</td>
</tr>
<tr>
<td>JDYCDEVCL</td>
<td>Device class: JDDCNJE for NJE devices JDDCOFLD for OFFLOAD devices</td>
</tr>
<tr>
<td>JDYCNAME</td>
<td>Device name</td>
</tr>
<tr>
<td>JDYCSTAT</td>
<td>Device status (see common device status flags):</td>
</tr>
<tr>
<td></td>
<td>JDYCSTA1 First status byte</td>
</tr>
<tr>
<td></td>
<td>JDYCSTA2 Second status byte</td>
</tr>
<tr>
<td>JDYCSYSN</td>
<td>Owning MVS system name</td>
</tr>
<tr>
<td>JDYCMBRN</td>
<td>JESPLEX member name</td>
</tr>
<tr>
<td>JDYCECL</td>
<td>Security label</td>
</tr>
<tr>
<td>JDYCCSTA</td>
<td>Status, character value</td>
</tr>
</tbody>
</table>

*SYSOUT transmitter NJE section (JDYNSXMT):* SYSOUT transmitter NJE section. Contains attributes specific for JES2 NJE SYSOUT transmitters.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYNLNG</td>
<td>Length of this section</td>
</tr>
</tbody>
</table>
Values for attributes used for work selection. Dataset size range for work selection (records):

Field Name
Description

JDYNWDSL
Dataset size low limit

JDYNWDSH
Dataset size high limit

SYSOUT size range for work selection (pages):

JDYNWPLL
Page limit - low limit

JDYNWPLH
Page limit - high limit

JDYNFLAG
Work selection flags:

Bit Value
Description

JDYNFODH
Select output with OUTDISP=HOLD

JDYNFODK
Select output with OUTDISP=KEEP

JDYNFODL
Select output with OUTDISP=LEAVE

JDYNFODW
Select output with OUTDISP=WRITE

Work selection criteria in printable form. The work selection criteria string is represented in the format which would be used by appropriate JES configuration command.

Field Name
Description

JDYNWSCO
Offset from the beginning of DSECT to the work selection string

JDYNWSCL
Length of the work selection string

Work selection criteria in encoded form. The work selection criteria is encoded as an array of bytes, where the value of each byte represents an attribute used for work selection. (See symbol definitions for work selection attributes - JDWSxxx.)

Field Name
Description
SSI Function Code 83

JDYNWSEO
Offset from the beginning of DSECT to the work selection array

JDYNWSEL
Length of the work selection array

SYSOUT transmitter OFFLOAD section (JDYOSXMT): SYSOUT transmitter OFFLOAD section. Contains attributes specific for JES2 OFFLOAD SYSOUT transmitters.

Field Name
Description

JDYOLNG
Length of this section

JDYOTYPE
Section type

JDYOMOD
Section type modifier

JDYOFGLG1
Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYO1NFY</td>
<td>Send notification message to TSO userid as requested</td>
</tr>
<tr>
<td>JDYO1STR</td>
<td>Start this transmitter when OFFLOAD device is started</td>
</tr>
</tbody>
</table>

JDYODISP
Post-offload SYSOUT disposition:

JDYODDEL
DELETE

JDYODHLD
HOLD

JDYODKP
KEEP

Values for attributes used for work selection:

Field Name
Description

JDYOWFCB
FCB name

JDYOWFLH
Flash ID

JDYOWOWN
Dataset owner/creator

JDYOWJBN
Job name

Dataset size for work selection (records):
### SSI Function Code 83

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWDLL</td>
<td>Dataset size low limit</td>
</tr>
<tr>
<td>JDYOWDHL</td>
<td>Dataset size high limit</td>
</tr>
</tbody>
</table>

SYSOUT size for work selection (pages):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWPLL</td>
<td>Page limit - low limit</td>
</tr>
<tr>
<td>JDYOWPLH</td>
<td>Page limit - high limit</td>
</tr>
</tbody>
</table>

Job ID range for work selection:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWJIL</td>
<td>Job ID low limit</td>
</tr>
<tr>
<td>JDYOWJIH</td>
<td>Job ID high limit</td>
</tr>
<tr>
<td>JDYOWUCS</td>
<td>UCS name</td>
</tr>
<tr>
<td>JDYOWWTR</td>
<td>Writer name</td>
</tr>
<tr>
<td>JDYOWPTY</td>
<td>Output priority</td>
</tr>
</tbody>
</table>

**Work selection flags:**

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWBRS</td>
<td>Select SYSOUT with BURST=YES (if not set - select BURST=NO)</td>
</tr>
<tr>
<td>JDYOWHLD</td>
<td>Select output which is held (if not set - select output which is not held)</td>
</tr>
<tr>
<td>JDYOWODH</td>
<td>Select output with OUTDISP=HOLD</td>
</tr>
<tr>
<td>JDYOWODK</td>
<td>Select output with OUTDISP=KEEP</td>
</tr>
<tr>
<td>JDYOWODL</td>
<td>Select output with OUTDISP=LEAVE</td>
</tr>
<tr>
<td>JDYOWODW</td>
<td>Select output with OUTDISP=WRITE</td>
</tr>
</tbody>
</table>
### Array of output classes
This is a variable size array of fixed-size character strings which represent names of output classes.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWCLO</td>
<td>Offset from the beginning of DSECT to the first output class</td>
</tr>
<tr>
<td>JDYOWCL#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDYOWCLL</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

### Array of form names
This is a variable size array of fixed-size character strings which represent form names.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWFMO</td>
<td>Offset from the beginning of DSECT to the first form name</td>
</tr>
<tr>
<td>JDYOWFM#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDYOWFML</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

### Array of processing modes
This is a variable size array of fixed-size character strings which represent names of processing modes.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWPMO</td>
<td>Offset from the beginning of DSECT to the first processing mode name</td>
</tr>
<tr>
<td>JDYOWPM#</td>
<td>Number of elements in array</td>
</tr>
<tr>
<td>JDYOWPML</td>
<td>Length of each element</td>
</tr>
</tbody>
</table>

### Array of routing codes/destination IDs
This is variable size array of fixed-size character strings which represent routing codes or destination IDs.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDYOWDSO</td>
<td>Offset from the beginning of DSECT to the first route code/dest ID</td>
</tr>
<tr>
<td>JDYOWDS#</td>
<td>Number of elements in array</td>
</tr>
</tbody>
</table>
SSI Function Code 83

JDYOWDSL
   Length of each element

Array of binary route codes for work selection. This is a variable-size array of
fixed-size structures, mapped by the JDD2DEST structure, of binary route codes
used for work selection.

Field Name  Description
JDYOWRCO  Offset from the beginning of DSECT to the first route code
JDYOWRC#  Number of elements in array
JDYOWRCL  Length of each element

Array of spool volume names for work selection. This is a variable-size array of
fixed-size character strings which represent spool volume names used for work
selection.

Field Name  Description
JDYOWVLO  Offset from the beginning of DSECT to the first volume name
JDYOWVL#  Number of elements in array
JDYOWVLL  Length of each element

Work selection criteria in printable form. The work selection criteria string is
represented in the format which would be used by appropriate JES configuration
command.

Field Name  Description
JDYOWSCO  Offset from the beginning of DSECT to the work selection string
JDYOWSCL  Length of the work selection string

Work selection criteria in encoded form. The work selection criteria is encoded as
an array of bytes, where the value of each byte represents an attribute used for
work selection. (See symbol definitions for work selection attributes - JDWSxxxx.)

Field Name  Description
JDYOWSEO  Offset from the beginning of DSECT to the work selection array
JDYOWSEL  Length of the work selection array
JDYOWFL2  Work selection flags:
SSI Function Code 83

Bit Value
Description
JDYO2JOB  
Job ID range is for JOB
JDYO2STC  
Job ID range is for STC
JDYO2TSU  
Job ID range is for TSU

SYSOUT transmitter device JES2 section (JDY2SXMT):  
SYSOUT transmitter device JES2 section Contains attributes of SYSOUT transmitter, which are specific to JES2.

Field Name
Description
JDY2LNG  
Length of this section
JDY2TYPE  
Section type
JDY2MOD  
Section type modifier
JDY2VID  
Binary device ID

NJE connection data header (JDJHNJEC):  
Header for NJE connection data. Data for NJE connection is returned within this header. Total length of this data is accounted for by the prefix section which follows this header structure. The prefix section will also identify the type of data returned in this header. For NJE connection the data (section) type is JDTYNJEC. The following sections could be returned within this header:

- Prefix section (always first) "Common prefix section (JDCXPREF)" on page 363
- NJE connection common section "NJE connection common section (JDJCNJEC)" on page 423
- SNA application section (NJE over SNA) "SNA application section (JDAPPLIC)" on page 425
- SNA application JES2 section "SNA application JES2 section (JDA2APPL)" on page 426
- TCP socket section (NJE over TCP/IP) "TCP socket section (JDSKSOCK)" on page 426
- TCP socket JES2 section "TCP socket JES2 section (JDK2SOCK)" on page 427

Data for devices related to that NJE connection is chained to this header using JDJHDEVC pointer. Number of devices in the chain is in JDJHDEV#. Data returned for each related device will have its own unique header structure. Depending on the type of NJE connection, devices related to NJE connection include:

- NJE job transmitters (see "Job transmitter data header (JDXHJXMT)" on page 409)
- NJE job receivers (see "Job receiver data header (JDBHJRCV)" on page 398)
- NJE SYSOUT transmitters (see "SYSOUT transmitter data header (JDYHSXMT)" on page 415)
- NJE SYSOUT receivers (see "SYSOUT receiver device data header (JDSHSRCV)" on page 403)
SSI Function Code 83

- Line device (see "Line device data header (JDLHLINE)" on page 371)
- NETSRV device (see "NETSRV device header (JDNHNSRV)" on page 368)
- Logon device (see "Logon device data header (JDGHLOGN)" on page 366)

Note: This data is not returned if dominance check of SSI caller security label vs security label of the adjacent node fails (see JDJCDNSL).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDJHEYE</td>
<td>Eye catcher</td>
</tr>
<tr>
<td>JDJHOHDR</td>
<td>Offset to first (prefix) section</td>
</tr>
<tr>
<td>JDJHDEV#</td>
<td>Number of related devices in the chain (see JDJHDEV8)</td>
</tr>
<tr>
<td>JDJHJPL8</td>
<td>Address of IAZJPLXI for this device</td>
</tr>
<tr>
<td>JDJHJPLX</td>
<td>31-bit part of a pointer</td>
</tr>
<tr>
<td>JDJHNEX8</td>
<td>Address of header of the next NJE connection</td>
</tr>
<tr>
<td>JDJHNEXT</td>
<td>31-bit part of the pointer</td>
</tr>
<tr>
<td>JDJHDEV8</td>
<td>Address of header of the first related device</td>
</tr>
<tr>
<td>JDJHDEVVC</td>
<td>31-bit part of the pointer</td>
</tr>
</tbody>
</table>

NJE connection common section (JDJCNJEC): NJE Connection common section. Contains common attributes of NJE connection. NJE connection is uniquely identified by the NJE connection name (JDJCNAME), communication protocol type (JDJCPROT) and name of the owning system (JDJCSYSN).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDJCLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDJCTYPE</td>
<td>Section type</td>
</tr>
<tr>
<td>JDJCMOD</td>
<td>Section type modifier</td>
</tr>
<tr>
<td>JDJCNAME</td>
<td>NJE connection name</td>
</tr>
<tr>
<td>JDJCSYSN</td>
<td>Owning MVS system name</td>
</tr>
<tr>
<td>JDJCMBRN</td>
<td>JESPLEX member name</td>
</tr>
<tr>
<td>JDJCADJN</td>
<td>Adjacent node name</td>
</tr>
</tbody>
</table>
SSI Function Code 83

**JDJCNDSL**
Adjacent node security label

**JDJCSTAT**
NJE connection status (see common device status flags):

- **JDJCSTA1**
  First status byte

- **JDJCSTA2**
  Second status byte

**JDJCPROT**
Communication protocol type:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDJCPBSC</td>
<td>BSC</td>
</tr>
<tr>
<td>JDJCPSNA</td>
<td>SNA</td>
</tr>
<tr>
<td>JDJCPTCP</td>
<td>TCP/IP</td>
</tr>
</tbody>
</table>

**JDJCFLAG**
Processing flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDJCFAUT</td>
<td>Auto restart</td>
</tr>
<tr>
<td>JDJCFTRB</td>
<td>Basic trace requested</td>
</tr>
<tr>
<td>JDJCFTCM</td>
<td>Common code trace requested</td>
</tr>
<tr>
<td>JDJCFTEX</td>
<td>Extended trace requested</td>
</tr>
<tr>
<td>JDJFCNND</td>
<td>Auto connect required (CONNECT=YES)</td>
</tr>
<tr>
<td>JDJFCNDA</td>
<td>Auto connect not required (CONNECT=NO) if both JDJFCNDA and JDJFCNN are off, CONNECT=DEFAULT</td>
</tr>
</tbody>
</table>

**JDJCNAM2**
Associated device name:
- Line device name for BSC
- Logon device name for SNA
- NETSRV name for TCP/IP

**JDJCRINT**
Auto restart interval (minutes)

**JDJCRETR**
Maximum number of restart retries (0 - indefinite retry)
SSI Function Code 83

JDJCSTR#
Number of SYSOUT transmitters

JDJCSRC#
Number of SYSOUT receivers

JDJCJTR#
Number of job transmitters

JDJCJRC#
Number of job receivers

JDJCSKID
TCP/IP socket ID assigned by NETSRV (NJE over TCP/IP)

JDJCCSTA
Status, character value

Storage management control block DSECT (JDGSTRG): Storage management fields are for use by the subsystem only. The SSJDSTRP field in SSJD points to a chain of these control blocks. This storage is accessible to a caller but protected from modification.

Field Name
Description

JDSGEYE
Eye-catcher

JDSGSTHL
Length of header area

JDSGSTSP
Subpool of this block

JDSGSTPL
Recommended subpool to use

JDSGSTTL
Total length of this block (including this header)

JDSGNEXT
Pointer to next block

JDSGAVL
Pointer to first available byte

JDSGDATA
Start of data in the block

Common sections: The following structures represent common data sections. These sections can be returned for devices of different types.

SNA application section (JDAPPLIC): The SNA application section contains attributes of SNA applications. This section can be returned with the following information:
• Data for NJE connection over SNA (reports attributes of a peer application)
• Data for SNA line used for NJE (reports attributes of a peer application)
• Logon device used for NJE connection (reports attributes of a local application)

Field Name
Description
SNA application JES2 section (JDA2APPL): The SNA application JES2 section contains JES2-only attributes of SNA application. This section can be returned with the following information:

- Data for NJE connection over SNA (reports attributes of a peer application)
- Data for SNA line used for NJE (reports attributes of a peer application)
- Logon device used for NJE connection (reports attributes of a local application)

Field Name
Description

JDA2LNG
Length of this section

JDA2TYPE
Section type

JDA2MOD
Section type modifier

JDA2LNAM
Associated line name

JDA2LGNM
Associated logon name

JDA2LNDV
Associated line device ID

TCP socket section (JDSKSOCK): The TCP socket data section contains attributes of a TCP socket. This section can be returned with the following information:

- Data for NJE connection over TCP/IP (reports attributes of a peer socket)
- Data for TCP/IP line used for NJE (reports attributes of a peer socket)
- NETSRV device used for NJE connection (reports attributes of a local socket)

Field Name
Description

JDSKLNG
Length of this section
JDSKTYPE  
Section type

JDSKMOD  
Section type modifier

JDSKNAME  
Socket name

JDSKIHNO  
Offset to the IP host name from the section start

JDSKIHNL  
Length of the IP host name

JDSKIADR  
IP address

JDSKTPNM  
TCP port name

JDSKTPNR  
TCP port number

JDSKNSRV  
NETSRV name

JDSKFLAG  
Socket flags:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDSKFTLS</td>
<td>Secure socket (TLS)</td>
</tr>
<tr>
<td>JDSKFSRV</td>
<td>Server-type socket - dynamically created for inbound (passive) TCP connections</td>
</tr>
</tbody>
</table>

TCP socket JES2 section (JDK2SOCK): The TCP socket data JES2 section contains attributes of a TCP socket. This section can be returned with the following information:

- Data for NJE connection over TCP/IP (reports attributes of a peer socket)
- Data for TCP/IP line used for NJE (reports attributes of a peer socket)
- NETSRV device used for NJE connection (reports attributes of a local socket)

Field Name  
Description

JDK2LNG  
Length of this section

JDK2TYPE  
Section type

JDK2MOD  
Section type modifier

JDK2LNAM  
Associated line name

JDK2LNDV  
Associated line device ID
**Current job information section (JDJINFO):** This section reports the job-level information on the currently active job on the device. This section is optional and is only returned if the device is currently processing a job. Depending on the version of JES and the type of device, some fields might not be reported.

Note that the job information is only available if the security label of the caller dominates the security label of the job owner.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDJBLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDJBTYP</td>
<td>Section type. Set to JDTYJOB by the subsystem.</td>
</tr>
<tr>
<td>JDJBMOD</td>
<td>Section type modifier. Set to JDMDJBCM by the subsystem.</td>
</tr>
<tr>
<td>JDJBJOBN</td>
<td>Name of the job being processed</td>
</tr>
<tr>
<td>JDJBJOBI</td>
<td>Job ID of the job being processed</td>
</tr>
<tr>
<td>JDJBJNUM</td>
<td>Job number of the job being processed</td>
</tr>
<tr>
<td>JDJBOOWN</td>
<td>Name of the owner of the job, or name of the creator of the SYSOUT dataset being processed by the device</td>
</tr>
<tr>
<td>JDJBOWSL</td>
<td>Security label of the owner of the job, or of the creator of the SYSOUT dataset being processed by the device</td>
</tr>
<tr>
<td>JDJBJOBC</td>
<td>Job class of the job being processed</td>
</tr>
<tr>
<td>JDJBPRIO</td>
<td>Job priority</td>
</tr>
<tr>
<td>JDJBHTYP</td>
<td>Job type</td>
</tr>
<tr>
<td>JDJBSTC</td>
<td>Started Task (STC)</td>
</tr>
<tr>
<td>JDJBTSU</td>
<td>Time Sharing User (TSU)</td>
</tr>
<tr>
<td>JDJBJOB</td>
<td>Batch job (JOB)</td>
</tr>
</tbody>
</table>

Job-level progress counters (for devices that process jobs rather than SYSOUT data sets):
### Field Name
#### Description

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDJBTRC#</td>
<td>Total records in the job</td>
</tr>
<tr>
<td>JDJBPRC#</td>
<td>Number of job records processed</td>
</tr>
</tbody>
</table>

**Current output information section (JDUTINFO):** This section contains the information on the SYSOUT data set currently processed by the device. This section is optional and is only returned if the device is processing a SYSOUT dataset.

Note that the SYSOUT data set information is only available if the security label of the caller dominates the security label of the owner or creator of the SYSOUT data set.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDUTLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDUTTYPE</td>
<td>Section type. Set to JDYOUTI by the subsystem.</td>
</tr>
<tr>
<td>JDUTMOD</td>
<td>Section type modifier. Set to JDMDOTCM by the subsystem.</td>
</tr>
<tr>
<td>JDOUTUTC</td>
<td>Output class of the SYSOUT data set being processed</td>
</tr>
<tr>
<td>JDUTFORM</td>
<td>Current form name</td>
</tr>
<tr>
<td>JDUTPRMD</td>
<td>Current processing mode (PRMODE)</td>
</tr>
<tr>
<td>JDUTWRTN</td>
<td>Current writer name</td>
</tr>
<tr>
<td>JDUTJBN</td>
<td>Transaction job name (for transactional output)</td>
</tr>
<tr>
<td>JDUTTWKI</td>
<td>Transaction work ID (for transactional output)</td>
</tr>
<tr>
<td>JDUTFLSH</td>
<td>Current flash ID (FLASH)</td>
</tr>
<tr>
<td>JDUTFCB</td>
<td>Current Forms Control Buffer (FCB) name</td>
</tr>
<tr>
<td>JDUTUCS</td>
<td>Current Universal Character Set (UCS) name</td>
</tr>
<tr>
<td>JDUTDEST</td>
<td>Current destination ID</td>
</tr>
<tr>
<td>JDUTPRIO</td>
<td>Output priority</td>
</tr>
<tr>
<td>JDUTFLG1</td>
<td>Flags:</td>
</tr>
</tbody>
</table>

**Chapter 3. SSI Function Codes Your Program Can Request** 429
**SSI Function Code 83**

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDUT1BRS</td>
<td>Burst setting (ON=YES, OFF=NO)</td>
</tr>
</tbody>
</table>

Progress counters of active SYSOUT data set:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDUTTPG#</td>
<td>Total pages in the SYSOUT data set</td>
</tr>
<tr>
<td>JDUTPPG#</td>
<td>Number of pages processed</td>
</tr>
<tr>
<td>JDUTTRC#</td>
<td>Total records in the SYSOUT dataset</td>
</tr>
<tr>
<td>JDUTPRC#</td>
<td>Number of records processed</td>
</tr>
</tbody>
</table>

*Current output information JES2 section (JDU2INFO):* This section contains the SYSOUT data set information specific to JES2. This section is an expansion to the current output information section ("Current output information section (JDU2INFO)" on page 429).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDU2LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDU2TYPE</td>
<td>Section type. Set to JDYOUTI by the subsystem.</td>
</tr>
<tr>
<td>JDU2MOD</td>
<td>Section type modifier. Set to JDMDOTJ2 by the subsystem.</td>
</tr>
<tr>
<td>JDU2JOID</td>
<td>Job Output Element (JOE) identifier of the SYSOUT data set currently processed by the device</td>
</tr>
<tr>
<td></td>
<td>This identifier consists of the following elements:</td>
</tr>
<tr>
<td>JDU2JOEN</td>
<td>Name of the JOE</td>
</tr>
<tr>
<td>JDU2JOE1</td>
<td>JOE ID field 1</td>
</tr>
<tr>
<td>JDU2JOE2</td>
<td>JOE ID field 2</td>
</tr>
<tr>
<td>JDU2IMQT</td>
<td>JES2 spool record address (MQTR) of a spin IOT (same MQTR format as used for spool read SSI)</td>
</tr>
<tr>
<td>JDU2DEST</td>
<td>Binary route code/destination ID of the SYSOUT data set</td>
</tr>
<tr>
<td></td>
<td>The value of this field is mapped by the binary route code structure (&quot;Binary route code structure (JDD2DEST)&quot; on page 431)</td>
</tr>
</tbody>
</table>
|            | This field consists of the following elements:
Current output information JES3 section (JDU3INFO): This section contains the SYSOUT data set information specific to JES3. This section is an expansion to the current output information section (“Current output information section (JDU1INFO)” on page 429).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDU3LNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JDU3TYPE</td>
<td>Section type. Set to JDTYOUTI by the subsystem.</td>
</tr>
<tr>
<td>JDU3MOD</td>
<td>Section type modifier. Set to JDMDOTJ3 by the subsystem.</td>
</tr>
<tr>
<td>JDU3COPY</td>
<td>Copy count</td>
</tr>
</tbody>
</table>

Binary route code structure (JDD2DEST): The structure definition maps a JES2 binary route code.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDD2NODE</td>
<td>NJE node number (nodal part)</td>
</tr>
<tr>
<td>JDD2RTE</td>
<td>Remote workstation number (remote part)</td>
</tr>
<tr>
<td>JDD2USER</td>
<td>User name (user ID part)</td>
</tr>
</tbody>
</table>

Modify Job Function Call — SSI Function Code 85

The modify job function call (SSI function code 85) allows a user-supplied program to modify job properties and to manage memory associated with the request. For z/OS 2.1, only JES2 subsystems support the job modification SSI.

Modifying a job requires a specific level of security access, depending on the requested action. This access is defined using JESJOBS profiles, which are described in the *Controlling who can modify job attributes using the Job Modify SSI 85* section in *z/OS Security Server RACF Security Administrator’s Guide*.

Additional security checks are made in the requestor’s address space for access to alter the job that is being processed. These security checks use the JESJOBS class and require additional resource names for the major actions that can be performed. *Table 12 on page 432* describes the JESJOBS entity names that are used.
Table 12. SSI 85 actions, JESJOBS class entities and Required access

<table>
<thead>
<tr>
<th>SSI 85 action</th>
<th>JESJOBS class entity</th>
<th>Required access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify</td>
<td>MODIFY.nodename.userid.jobname</td>
<td>Update</td>
</tr>
<tr>
<td>Hold</td>
<td>HOLD.nodename.userid.jobname</td>
<td>Update</td>
</tr>
<tr>
<td>Release</td>
<td>RELEASE.nodename.userid.jobname</td>
<td>Update</td>
</tr>
<tr>
<td>Purge</td>
<td>PURGE.nodename.userid.jobname</td>
<td>Alter</td>
</tr>
<tr>
<td>Cancel</td>
<td>CANCEL.nodename.userid.jobname</td>
<td>Alter</td>
</tr>
<tr>
<td>Restart</td>
<td>RESTART.nodename.userid.jobname</td>
<td>Control</td>
</tr>
<tr>
<td>Spin</td>
<td>SPIN.nodename.userid.jobname</td>
<td>Control</td>
</tr>
<tr>
<td>Reroute execution</td>
<td>REROUTE.nodename.userid.jobname</td>
<td>Update</td>
</tr>
<tr>
<td>Start</td>
<td>START.nodename.userid.jobname</td>
<td>Control</td>
</tr>
</tbody>
</table>

Modify job request types

The IAZSSJM modify job SSI does not require callers to be PSW key/state authorized. The IAZSSJM SSI does not support being broadcast, but it can be directed to any subsystem, including subsystems that run independently of any JES environment.

In z/OS 2.1, SSI85 supports two functions: modifying a job and any memory management that is required by the request. IAZSSJM supports the same filtering capabilities as Extended Status SSI 80 (macro IAZSSST).

IAZSSJM supports all of the following actions against jobs (JQEs):

- Modify job characteristics ($T) – specifically job class, priority (absolute or relative), SYSAFF (replacement list of JES members or MVS systems), service class, WLM scheduling environment or offload status (OFFS=).
- Hold a job ($H)
- Release a job ($A)
- Purge a job ($P)
- Cancel a job ($C) – with options to purge, dump, force or ARM restart.
- Restart a job ($E) – with the cancel or step and hold options
- SPIN a job ($T,SPIN) – with the optional DDNAME option.
- Change execution node ($R XEQ) – only before the execution phase.
- Start a job ($S)

Each of these items is an ACTION on the modify job function. Only one action is permitted per SSI call, but the action can operate on an arbitrary number of jobs.

Additional action-dependent data is included with a mask to indicate which fields are set. For example, to modify the job class for a job, a bit indicates that a new class was specified, with a new field for the updated class. Flags are provided to specify options, such as the purge option on the cancel action.

Type of Request

Directed SSI call only.
Use information

To use the modify job SSI, a caller must select one action to take. Only one action can be specified per call, but the action can run on an unlimited number of selected jobs.

Next, the caller must decide which filters to use. A filter is an attribute that a job or SYSOUT must possess to be processed by the interface. Filters exist at the job or SYSOUT level, and are independent of the type of action being taken.

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 filters for jobs that are held. If no filters are applied, the modify job function call is returned with an error. Because the number of jobs and SYSOUT in the system can be large, it is recommended that a filter be specified to limit the jobs that the action runs on.

When an application calls SSI 85, the SSI clears any output areas that are passed in to the SSI. SSI 85 considers it residual data and clears it to avoid any confusion with the jobs that are selected for the current request. Each SSI 85 call results in one set of output areas that must be processed before making another SSI 85 request.

For JES2 subsystems, jobs processed through this SSI can be obtained from a local copy of JES2’s work queues. SSI 85 processes job or SYSOUT data to determine whether the job should be selected for the modify request. When JES2 gets the request to modify the job, the job might no longer be in a state where the modify request is allowed, in which case the request is rejected.

The modify job SSI can operate synchronously or asynchronously, as specified by the input processing option flag byte SSJMOPT1. Turn the SSJMPSYN bit ON to indicate that the SSI is to perform the requested job modify action synchronously. For a synchronous request, the job modify SSI does not return control to the caller until the job modify action has been processed for every selected job and results of the request are available to be returned to the caller. A synchronous request provides results of the job modify action for each job so the user can analyze which jobs were modified successfully and which ones were unable to be modified, with the reason why any job could not be modified.

If you do not require feedback on the results of the job modify action, you can use an asynchronous request. For an asynchronous request, the job modify SSI returns control to the caller once all selected jobs are queued to JES2. The caller does not get feedback on the results of the job modify action, but gets control back more quickly using an asynchronous request.

The modify job request does not provide a method to freeze the job status in the system. Other SAPI applications, JES writers, networking writers and operators can change the state of any job that has been selected or updated by the job modify request. The job status might allow the job to be selectable at the time of a job modify request, but by the time the action is performed, the job status might no longer allow the job to be modified. Likewise, after job properties are modified by a job modify request, they might be changed by some other process before the SSI caller processes the job feedback information from SSI 85.

The response to a modify job request includes an output feedback element (SSJF) for each job that is selected to be modified. The elements are chained together and are anchored by the SSJ[MS]F8/SSJ[MS]FP field in IAZSSJM(SSJM).
SSI Function Code 85

Issued to
- A JES2 subsystem (either primary or secondary) as a directed request.

Related SSI Codes
None.

Related Concepts
None.

Environment
The caller (issuer of the IAZSSJM 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
- IAZSSJM

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Caller variable</th>
<th>Caller value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Problem state, any PSW key.</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 SSOB, SSIB, and IAZSSJM control blocks can reside above or below 16 megabytes in virtual storage.</td>
</tr>
<tr>
<td>Recovery</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 29 on page 435 shows the environment at the time of the call for SSI function code 85.
### Input Register Information

Before issuing the IAZSSJM 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
- IAZSSJM

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

---

**Figure 29. Environment at Time of Call for SSI Function Code 85**
SSI Function Code 85

Field Name     Description
SSOBID          Identifier ‘SSOB’
SSOBLEN         Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC        SSI function code 85 (SSOBSSJM)
SSOBSSIB        Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See [Subsystem Identification Block (SSIB)] on page 8 for more information on the life-of-job SSIB.
SSOBINDV        Address of the function-dependent area (IAZSSST control block).

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

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

Field Name     Description
SSIBID          Identifier ‘SSIB’
SSIBLEN         Length of the SSIB (SSIBSIZE) control block
SSIBSSNM        Subsystem name — name of the subsystem to which this extended status function call is directed (or MSTR if it is to be broadcast).

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

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

Specify SSJM1CHR and SSJMZOMO to identify for the SSI service which characters in selection EBCDIC strings are wildcard characters. If SSJM1CHR and SSJMZOMO are not specified, the default wildcard characters are “?” for SSJM1CHR and “*” for SSJMZOMO. If either value is not X’00’ (if either is specified), then both provided values are used even if one value is X’00’.

It is an error to specify equal values for SSJM1CHR and SSJMZOMO, unless the equal values are X’00’. If both values are X’00’, the default values are used.

Job Modification Action Input Parameters: The following fields specify input parameters used to define the job modification action to be taken on selected jobs. These fields are only relevant when their corresponding job modify action is requested. Fields SSJMOPT1 and SSJMTYPE are supplied for any job modify action.
SSJMOPT1

Input processing option:
Flag        Description

SSJMPD64
ON - Return output in 64-bit virtual storage.

SSJMPSYN
SYNC request (ON) or ASYNC request (OFF).

SSJMTYPE

The input request type:

Request type        Description

SSJMHOlD (4)
Hold selected jobs.

SSJMRLS (8)
Release selected jobs.

SSJMPRG (12)
Purge selected jobs.

SSJMCANC (16)
Cancel selected jobs.

SSJMSTRT (20)
Start selected jobs.

SSJMRST (24)
Restart selected jobs.

SSJMSPIN (28)
SPIN selected jobs.

SSJMCHR (32)
Change characteristics of selected jobs.

SSJMNODE (36)
Change execution node of selected jobs.

SSJMRSTG (128)
Release storage.

SSJMHOlD, SSJMRLS and SSJMSTRT are job modify actions that do not require additional input parameters. The remaining action types require additional input parameters, and are described below.

SSJMPF1GL

Job purge (SSJMPRG) option flags:

Flag        Description

SSJMPPRT
Perform a protected cancel/purge of the jobs.

SSJMCFL1G

Job cancel (SSJMCANC) option flags:

Flag        Description
SSIJMCARM
Request that automatic restart management automatically restart
the selected jobs after they are cancelled.

SSJMCPRG
Purge output of the cancelled jobs.

SSJMEFLG
Job restart (SSJMRST) option flags:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJMECAN</td>
<td>Cancel and hold the restarted jobs prior to execution.</td>
</tr>
<tr>
<td>SSJMERES</td>
<td>Restart the selected jobs at the next processing step once the current step completes.</td>
</tr>
<tr>
<td>SSJMESTH</td>
<td>Hold and re-queue the restarted jobs for execution once the current step completes.</td>
</tr>
</tbody>
</table>

SSJMTSFL
Job SPIN (SSJMSPIN) option flags:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJMTSDD</td>
<td>SPIN only the data set specified in SSJMTSDN. Otherwise, SPIN all eligible data sets.</td>
</tr>
</tbody>
</table>

SSJMTSDN
If SSJMTSDD is ON, specifies the data set name to SPIN.

SSJMRNOD
Change Execution Node (SSJMNODE) request: the name of the node to route jobs to.

SSJMCOP1
Change characteristics (SSJMJCHR) request options:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJMCJBC</td>
<td>Change the job class of the selected jobs to the class specified in SSJMJBCL.</td>
</tr>
<tr>
<td>SSJMSCVC</td>
<td>Change the service class of the selected jobs to the class specified in SSJMSVCL.</td>
</tr>
<tr>
<td>SSJMCSCCH</td>
<td>Change WLM scheduling environment of the selected jobs to the WLM scheduling environment specified in SSJMSCEV.</td>
</tr>
</tbody>
</table>
**SSI Function Code 85**

**SSJMCPRA**
Change the priority of the selected jobs to the priority specified in SSJMJPRI (absolute priority). Mutually exclusive with SSJMCPRR.

**SSJMCPRR**
Change the priority of the selected jobs to the current priority PLUS the value in SSJMJPRI (relative priority - SSJMJPRI can be negative). Mutually exclusive with SSJMCPRA.

**SSJMCOFL**
Mark that selected jobs are offloaded on offload devices specified (by number) in the SSJMOLST offload device list. Mutually exclusive with SSJMCNOF.

**SSJMCNOF**
Mark that selected jobs are NOT offloaded on offload devices specified (by number) in the SSJMOLST offload device list. Mutually exclusive with SSJMCOFL.

**SSJMCAFF**
Change characteristics (SSJMJCHR) affinity options:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJMCANY</td>
<td>Selected jobs are eligible to run on ANY member (SYAFF=ANY).</td>
</tr>
<tr>
<td>SSJMCRPL</td>
<td>REPLACE the affinity list of selected jobs using the members listed in the SSJMCMBP affinity member list.</td>
</tr>
<tr>
<td>SSJMCADD</td>
<td>ADD TO the current affinity list of selected jobs using the members listed in the SSJMCMBP affinity member list.</td>
</tr>
<tr>
<td>SSJMCDEL</td>
<td>DELETE FROM the current affinity list of selected jobs using the members listed in the SSJMCMBP affinity member list.</td>
</tr>
</tbody>
</table>

**SSJMJBCL**
Change job class of selected jobs to this value. This option is only relevant when the SSJMCJBC option bit is set.

**SSJMSVCL**
Change service class of selected jobs to this value. This option is only relevant when the SSJMCSCVC option bit is set.

**SSJMSCEV**
Change WLM scheduling environment of selected jobs to this value. This option is only relevant when the SSJMCSCCH option bit is set.

**SSJMOLST**
List of offload device numbers. Each value is 1 byte and can have a value from 1-8. The list terminates with a 0. This option is only relevant when the SSJMCOFL or SSJMCNOF option bit is set.

**SSJMJPRI**
Change priority of the selected jobs using this value. This option is only relevant when the SSJMCPRA (absolute) or SSJMCPRR (relative) option bit is set. This value can be negative for processing a relative request.
SSI Function Code 85

SSJMCMBN
Member name count. This option is only relevant when one of the
SSJMCRL, SSJMCADD, or SSJMCDEL options are specified.

SSJMCMBP
Pointer to 4-byte member name list. This option is only relevant when one
of the SSJMCRL, SSJMCADD, or SSJMCDEL options are specified. This
list is a count followed by a pointer to a list of 4 byte member names. For
example, to set 'MB1','MB2','MB3', set:
- SSJMMLN = 'F3' 3 members in the list
- SSJMMP = A(MBRLIST) Pointer to member list
- MBRLIST = CL4'MB1 ,CL4'MB2 ,CL4'MB3 '

'* ' can be specified to denote the issuing member instead of explicitly
specifying a member name. Optionally, Independent mode (SYSAFF=IND)
can also be specified in the list using 'IND '.

Job Filter Indicator Input Parameters: The following fields specify indicators of
which job filters to use to select jobs to be processed.

Field Name
Description

SSJM1CHR
One byte value that indicates a one character wild card.

SSJMZOMO
One byte value that indicates a zero or more characters wild card

SSJMSEL1
Flag byte which describes the filters to use to select jobs. Each bit
corresponds to a filter field which must match any job returned.

Bit Name
Description

SSJMSCCLS
Apply job class filter in SSJMCLSL or SSJMCLSP. Only one class
needs to match. If only specifying one class, it must be specified in
SSJMCLSL.

SSJMMDST
Use SSJMDEST and SSJMDSTP as filters (Match any one dest).
Apply default destination filter in SSJMDEST or SSJMDSTP. Only
one destination needs to match. If only specifying one destination
it must be specified in SSJMDEST.

SSJMJOBN
Apply job name filter in SSJMJOBN or SSJMJBNP. Only one job
name needs to match. If only specifying one job name it must be
specified in SSJMJOBN. SSJMJOBN or SSJMSJBI cannot be specified
with SSJMSSJIL.

SSJMJSBI
Apply job ID filters in SSJMJBIL and SSJMJBIIH. SSJMJSBI cannot
be specified with SSJMSCLK, SSJMSSJIL, SSJMJSJBI or SSJMSCOR.

SSJMOJII
Apply original job ID filter in SSJMOJII.

SSJMSOWN
Apply current owner filter in SSJMSOWN.
### SSJMSSEC
Apply current SECLABEL filter in SSJMSECL.

### SSJMSEL2
Flag byte which describes the type of jobs for which data is requested.

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJMSSTC</td>
<td>Started tasks are selected.</td>
</tr>
<tr>
<td>SSJMSTSU</td>
<td>Time sharing users are selected.</td>
</tr>
<tr>
<td>SSJMSJOB</td>
<td>Batch jobs are selected.</td>
</tr>
<tr>
<td>SSJMSAPC</td>
<td>APPC initiators are selected. Because APPC initiators are also started tasks they are also returned if SSJMSSTC is specified. Use only SSJMSAPC to select only APPC initiators.</td>
</tr>
</tbody>
</table>

### SSJMSEL3
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.

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJMSPRI</td>
<td>Apply JES job priority filter in SSJMPRIO.</td>
</tr>
<tr>
<td>SSJMSVOL</td>
<td>Apply SPOOL volume filters in SSJMVOL.</td>
</tr>
<tr>
<td>SSJMSPHZ</td>
<td>Use SSJMPHAZ and SSJMPHZP as filters (match any one phase).</td>
</tr>
<tr>
<td>SSJMSHLD</td>
<td>Select jobs that are currently held. Setting both SSJMSHLD and SSJMSNHL on is the same as setting both bits off.</td>
</tr>
<tr>
<td>SSJMSNHL</td>
<td>Select jobs that are not currently held. Setting both SSJMSNHL and SSJMSHLD on is the same as setting both bits off.</td>
</tr>
<tr>
<td>SSJMSSYS</td>
<td>Only jobs active on the system listed in SSJMSYS are returned.</td>
</tr>
<tr>
<td>SSJMMSMEM</td>
<td>Only jobs active on the JES member listed in SSJMMEMB are returned.</td>
</tr>
</tbody>
</table>

### SSJMSEL4
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>SSJMSORG</td>
<td>Apply origin node filter in SSJMORGN.</td>
</tr>
</tbody>
</table>
SSJMSXEQ
Apply execution node filter in SSJMXEQN.

SSJMSSRV
Apply WLM service class filter in SSJMSRVC. When filtering by service class and not filtering by job number (SSJMSJBI) nor job phase (SSJMSPHZ), only jobs on the service class queue specified in SSJMSRVC are returned. When filtering on job number or job phase, any job assigned the service class specified in SSJMSRVC 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.

SSJMSESEN
Apply scheduling environment filter in SSJMSENV.

SSJMCLXL
Apply job class filter in SSJMCLSL and SSJMCLSP only to jobs in SSJM-SELECT or SSJM-ONMAIN phase.

SSJMSOJD
Do not apply job name filter in SSJMJOBN and SSJMJOBNP and job id filters in SSJMJOBIL and SSJMJOBIH to jobs that created OUTPUT with STST1APC on. SSJMSOJD cannot be specified with SSJMSJIL.

SSJMSJIL
Use SSJMJOBNP as a list of 8 character JES JOBIDs for which information is to be returned. The complete list is specified using SSJMJOBNP. JOBIDs cannot be specified using SSJMJOBIL and SSJMJOBIH when SSJMSJIL is specified. SSJMSJIL cannot be specified with SSJMSBN, SSJMSBI, SSJMSCTK, SSJMSTPI, SSJMSTPN, SSJMSOJD or SSJMSCOR.

SSJMSEL5
Flag byte which describes the filters to use to select jobs. Each bit corresponds to a filter field which must match any job returned.

Bit Name  Description

SSJMSCOR
Use SSJMJCRP as a pointer to a job correlator filter. SSJMSCOR cannot be specified with SSJMSJBI, SSJMSCTK or SSJMSJIL.

SYSOUT Filter Indicator Input Parameters: The following fields specify additional SYSOUT filters used to select jobs to be processed. All output owned by the job will be visited and compared to the filtering criteria. If ANY output matches the filter criteria, the job will be selected.

SSJMSSSL1
Flag byte which describes the SYSOUT filters to use to select jobs to modify. Each bit corresponds to a filter field which must match for the job to be selected for modification. Only jobs with SYSOUT that match the specified filters are selected.

Bit Name  Description

SSJMSCCTK
Use the SYSOUT token in SSJMCTKN as a filter. SYSOUT tokens
can be obtained from dynamic allocation or field SSJMCTKN from a previous extended status request. SSJMSCTK cannot be specified with SSJMSJBI or SSJMSCOR.

**SSJMSSOW**

Apply the SYSOUT owner filter in SSJMSCRE.

**SSJMSSDS**

Apply the SYSOUT destination filter in SSJMSDES. SSJMSDSP also contains additional SYSOUT destination filters. Mutually exclusive with SSJMSLCL or SSJMSNT.

**SSJMSSCL**

Apply the SYSOUT class filter in SSJMSCLA. SSJMSCLP also contains additional SYSOUT class filters.

**SSJMSSWR**

Apply the SYSOUT external writer filter in SSJMSWTR.

**SSJMSSHL**

Select held SYSOUT.

**SSJMSSNH**

Select non-held SYSOUT.

**Note:** Setting SSJMSSHL and SSJMSSNH both ON has the same effect as setting them both off.

**SSJMSSL2**

Flag byte that describes the SYSOUT filters to use to select jobs to modify. Only jobs with SYSOUT that match the specified filters are selected for modification.

**SSJMSSF**

Apply the SYSOUT forms name filter in SSJMSFOR.

**SSJMSSSF**

Apply the SYSOUT PRMODE filter in SSJMSPRM.

**SSJMSSSP**

Apply the Select SPIN output only filter in SSJMSSP. SSJMSSSP and SSJMSNS are mutually exclusive. If SSJMSSSP and SSJMSNS are both ON or both OFF, then the spin state of the output will not be considered.

**SSJMSSNS**

Apply the non-SPIN output only filter in SSJMSNS. SSJMSSSP and SSJMSNS are mutually exclusive. If SSJMSSSP and SSJMSNS are both ON or both OFF, then the spin state of the output will not be considered.

**SSJMSSIP**

Select SYSOUT elements that are routed to an IP address.

**SSJMSSNI**

Select SYSOUT elements that are not routed to an IP address.

**SSJMSSOD**

When on with SSJMSSOW, it indicates to match if SYSOUT is destined to SSJMSCRE on the local node.

**SSJMSSJD**
SSI Function Code 85

- If JES2 is running with checkpoint mode z2 in R11 and
  SSJMSJBN is on, it indicates to match if SYSOUT is destined to
  SSJMJOBN or SSJMJBNP on the local node (ignored if SSJMSJBN
  is off).
- If JES2 is running with checkpoint mode z11 in R11 and
  SSJMSJBN and SSJMSTPN are on, it indicates to match if
  SYSOUT is destined to SSJMJOBN, SSJMJBNP or transaction job
  name on the local node (ignored if SSJMSJBN is off).

SSJMSSL3

More SYSOUT selection criteria.

SSJMSSLC
Select SYSOUT that is destined to the local node. If SSJMSSLC and
SSJMSSNT 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 SSJMSSDS being set.

SSJMSSNT
Select SYSOUT that is not destined to the local node. If SSJMSSLC
and SSJMSSNT 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 SSJMSSDS being set.

SSJMSSNJ
For selection purposes, treat SYSOUT destined to an NJE node as
OUTDISP of WRITE regardless of the actual OUTDISP. This has no
effect if SSJMMSWRT, SSJMESHOL, SSJMSKEP and SSJMSLVE are all
on or all off.

SSJMMSWRT
Select output that has an OUTDISP of WRITE.

SSJMESHOL
Select output that has an OUTDISP of HOLD.

SSJMSKEP
Select output that has an OUTDISP of KEEP.

SSJMSLVE
Select output that has an OUTDISP of LEAVE.

Note: Setting SSJMMSWRT, SSJMESHOL, SSJMSKEP and SSJMSLVE all on has
the same effect as setting them all off.

SSJMSSL4

SSJMSSL4 is used to support filtering of jobs based on transaction name,
transaction job id, or transaction owner.

SSJMSTPN
Transaction job name filtering. SSJMSTPN cannot be specified with
SSJMSJIL. If this bit is on, jobs with SYSOUT associated with a
transaction job name that matches SSJMJOBN or SSJMJBNP are
selected for modification. The SSJMSTPN bit is ignored if one of
the following situations occurs:
  • SSJMSJBN is not set.
  • SSJMSOJD is not set.
  • JES2 is not running with checkpoint mode z11.
SSJMSTPI
Transaction job ID filtering. SSJMSTPI cannot be specified with SSJMSJIL.
• If SSJMSTPI is not set, only jobs with SYSOUT that has a job ID in the range specified by SSJMJBIL and SSJMJBIH are selected.
• If SSJMSTPI is set, jobs with SYSOUT associated with a SYSOUT data set with a transaction job ID are also selected. The job ID is in the range specified by SSJMJBIL and SSJMJBIH.

The SSJMSTPI bit is ignored if one of the following situations occurs:
• SSJMSJBI is not set.
• SSJMSOJD is not set.
• JES2 is not running with checkpoint mode z11.

SSJMSTPU
SYSOUT owner filtering. If this bit is on, jobs with SYSOUT that are associated with a SYSOUT data set whose transaction owner matches SSJMOWNR are returned. The SSJMSTPU bit is ignored if one of the following situations occurs:
• SSJMSOWN is not set.
• JES2 is not running with checkpoint mode z11.

SSJMSSJ1
If SYSOUT is destined to SSJMJOBN or SSJMJBNP on the local node, SSJMSSJ1 indicates to match using the first jobname supplied in SSJMJOBN in the following situations:
• If SSJMSSJ1 is on with SSJMSJBN
• If SSJMSTPN is on with SSJMSJBN
SSJMSSJ1 is ignored if SSJMSJBN is off.

Job Filter Value Input Parameters
The following fields specify job filter values used to select jobs to be processed. These fields are only relevant when the corresponding job filter indicator request bit is set.

Field Description
SSJMJOBN
Job name filter (used if SSJMSJBN is set). The name is 1-8 characters, left justified, and padded on the right with blanks. The generic characters ‘*’ and ‘?’ are allowed.

SSJMJBIL
Low job ID value (used if SSJMSJBI is set). The job ID is left justified and padded on the right with blanks. When SSJMJBIL is 2-8 characters and starts with one of the prefixes ‘I’, ‘IN’, ‘INT’, ‘J’, ‘JO’, ‘JOB’, ‘T’, ‘TS’, ‘TSU’, ‘S’, ‘ST’, ‘STC’ or ‘*’, then the suffix is converted to a binary value. Job IDs with a suffix matching the SSJMJBIL suffix are returned. The prefix character ‘*’ is not allowed for verbose requests.

When SSJMJBIL 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 SSJMJBIL are returned. A single character SSJMJBIL with ‘*’ or ‘?’ is not allowed. SSJMJBIL must be blank. Generics characters ‘*’ or ‘?’ are not allowed for verbose requests.
When SSJMSTPI is set and SSJMJBIL 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 SSJMJBIL suffix are also returned.

When SSJMSTPI is set and SSJMJBIL 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 SSJMJBIL are also returned. A single character SSJMJBIL with '*' or '?' is not allowed. When generic characters are used, SSJMJBIH must be blank.

**SSJMJBIH**

High job ID value (used if SSJMSJBI is set). If this field is not specified, then information is only returned using the filter specified in SSJMJBIL.

When SSJMJBIH is 2-8 characters and starts with one of the prefixes 'I', 'IN', 'INT', 'J', 'JO', 'JOB', 'T', 'TS', 'TSU', 'S', 'ST', 'STC', then the suffix is converted to a binary value. Job IDs with a suffix within the range from the SSJMJBIL suffix through the SSJMJBIH suffix are returned. Generics characters '*' or '?' are not allowed.

When SSJMSTPI 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 SSJMJBIL through SSJMJBIH, are returned. Generics characters '*' or '?' are not allowed.

See [Table 13](#) Numeric matches are in normal font, EBCDIC matches are in italicized font.

<table>
<thead>
<tr>
<th>SSJMJBIL</th>
<th>Examples of standard job ID matches</th>
<th>Examples of transaction job ID matches if SSJMSTPI 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 SSJMSTPI is not on.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es):A100</td>
</tr>
<tr>
<td>*000100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): no additional matches.</td>
<td>Numeric match(es): JOB00100, A000100, Z100, ZZZZZ100, and so on. EBCDIC match(es): no additional matches.</td>
</tr>
<tr>
<td>*100</td>
<td>Numeric match(es): JOB00100 or INT0100, and so on. EBCDIC match(es): JOB0100, T9999100, and so on.</td>
<td>Numeric match(es): JOB00100, A000100, Z100, and so on. EBCDIC match(es): JOB0100, 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): 5555555</td>
</tr>
<tr>
<td>*55555</td>
<td>Numeric match(es): JO5555555 or ST5555555 and so on. EBCDIC match(es), T9555555, J8555555, and so on.</td>
<td>Numeric match(es):JO5555555, ZZ5555555, and so on. EBCDIC match(es): Z5555555, 55555555, and so on.</td>
</tr>
</tbody>
</table>

See [Table 13](#) Examples of jobs returned for SSJMJBIL when SSJMJBIH is blank.
### Table 13. Examples of jobs returned for SSJMJBIL when SSJMJBIH is blank. (continued)

<table>
<thead>
<tr>
<th>SSJMJBIL</th>
<th>Examples of standard job ID matches</th>
<th>Examples of transaction job ID matches if SSJMSTPI is on</th>
</tr>
</thead>
<tbody>
<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): A000100, Z0001A0, K0001J0, KT001P0, and so on.</td>
</tr>
<tr>
<td>ZZ#00100</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): ZZ#00100</td>
<td>Error if SSJMSTPI is not on.</td>
</tr>
</tbody>
</table>

**SSJMOJBI**
Job ID value originally assigned to the job (used if SSJMSOJI 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.

**SSJMOWNR**
Current user ID that the security product has assigned as owner of the job (used if SSJMSOWN is set). The owner is 1-8 character, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

**SSJMSECL**
Current SECLABEL that the security product has assigned to the job (used if SSJMSSEC is set). The SECLABEL is 1-8 character, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

**SSJMORGN**
NJE node where the job originated (used if SSJMSORG is set). The origin node is 1-8 character, left justified, and padded on the right with blanks.
SSI Function Code 85

SSJMXEQN
NJE node where the job is to, or was, executed (used if SSJMSORG is set).
The execution node is 1-8 character, left justified, and padded on the right
with blanks.

SSJMCLSL
The job class associated with the job (used if SSJMSCLS is set). The job
class is 1-8 character, left justified, and padded on the right with blanks.

SSJMSYS
The name of the MVS system on which the job must be active (used if
SSJMSSYS 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.

SSJMMEMB
The name of the JES member on which the job must be active (used if
SSJMSMEM 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.

SSJMSRVC
The name of the WLM service class assigned to the job (used if SSJMSSRV
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.

SSJMSENV
The name of scheduling environment (SCHENV= from the JOB statement)
required by a job. (used if SSJMSSEN 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.

SSJMDEST
Default print or punch destination assigned to the job (used if SSJMSDST
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.

The user ID portion of the destination can contain the generic characters '*'
and '?'. This can match jobs with a default print route code that contains a
matching user ID 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.

SSJMVOL
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 SSJMSVOL is set). The SPOOL VOLSERS are each 1-6
caracter, left justified, and padded on the right with blanks. Unused
entries can be set to blanks or zero.

SSJMSPRI
The 1-byte binary priority associated with the job (used if SSJMSPRI is set).
The job’s priority must match exactly to be selected.

Valid priorities are 0-15.
SSI Function Code 85

<table>
<thead>
<tr>
<th>SSJMPHAZ</th>
<th>The current job processing phase (used if SSJMSPHZ is set).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The following values are valid for SSJMPHAZ:</td>
</tr>
<tr>
<td></td>
<td>Phase Value Description</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_INPUT</strong> Job is active in input processing.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_WTCNV</strong> Job is queued for conversion.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_CONV</strong> Job is actively converting.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_VOLWT</strong> Job is queued for SETUP (not currently used by JES2 code).</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_SETUP</strong> Job is active in SETUP (not currently used by JES2 code).</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_SELECT</strong> Job is queued for execution.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_ONMAIN</strong> Job is actively executing.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_SPIN</strong> JES2 is processing SPIN data sets for the JOB.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_WTBKDN</strong> Job is queued for output processing.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_BRKDWN</strong> Job is active in output processing.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_OUTPT</strong> Job is on the hard copy queue.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_WTPURG</strong> Job is queued for purge.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_PURG</strong> Job is currently being purged.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_RECV</strong> Job is active on an NJE SYSOUT receiver.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_WTXMIT</strong> Job is queued for execution on another NJE node.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_XMIT</strong> Job is active on an NJE JOB transmitter.</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_EXEC</strong> Job has not completed execution (combines multiple states in one phase request).</td>
</tr>
<tr>
<td></td>
<td><strong>SSJM_POSTEX</strong> Job has completed execution (combines multiple states in one phase request).</td>
</tr>
</tbody>
</table>
SYSOUT Filter Value Input Parameters: The following fields specify SYSOUT filter values used to select jobs to be processed. These fields are only relevant when the corresponding SYSOUT filter indicator request bit is set.

SSJMCTKN
Address of client token for selection (if SSJMSCTK is on).

SSJMSCRE
SYSOUT owner (creator) for selection (if SSJMSSOW is on).

SSJMSCLA
SYSOUT class for selection (if SSJMSSCL is on). The SYSOUT class is 1-8 characters in length. Additional classes pointed to by SSJMSCLP.

SSJMSWTR
SYSOUT writer name for selection (if SSJMSSWR is on).

SSJMSFOR
SYSOUT forms name for selection (if SSJMSSFR is on).

SSJMSPRM
SYSOUT PRMODE for selection (if SSJMSSPR is on).

SSJMSDES
SYSOUT destination for selection (if SSJMSSDS is on). Additional SYSOUT destinations pointed to by SSJMSDSP.

Filter Value List Input Parameters: The following fields specify lists of job and SYSOUT filter values to apply when selecting jobs for modification. These fields are only relevant when their corresponding filter indicator request bit is set.

SSJMCLSN
Count of job classes in the job class list SSJMCLSP. Used for selecting jobs if SSJMSCLS is ON.

SSJMCLSP
Pointer to a list of job classes. Used for selecting jobs if SSJMSCLS is ON.

SSJMJBNN
Count of job names in the job name list SSJMJBNP. Used for selecting jobs if SSJMSJBN is ON.

SSJMJBNP
Pointer to a list of job names. Used for selecting jobs if SSJMSJBN is ON.

SSJMDSTN
Count of job destination names in the job destination list SSJMDSTP. Used for selecting jobs if SSJMSDST is ON.

SSJMDSTP
Pointer to a list of job destination names. Used for selecting jobs if SSJMSDST is ON.

SSJMPHZN
Count of job phases in the job phase list SSJMPHZP. Used for selecting jobs if SSJMSPHZ is ON.

SSJMPHZP
Pointer to a list of job phases. Used for selecting jobs if SSJMSPHZ is ON.

SSJMSCLN
Count of SYSOUT classes in the SYSYOUT class list SSJMSCLP. Used for selecting jobs if SSJMSSCL is ON.
SSI Function Code 85

**SSJMSCLP**
- Pointer to a list of SYSOUT classes. Used for selecting jobs if SSJMSSCL is ON.

**SSJMSDSN**
- Count of SYSOUT destinations in the SYSYOUT destination list SSJMSDSP. Used for selecting jobs if SSJMSSDS is ON.

**SSJMSDSP**
- Pointer to a list of SYSOUT destinations. Used for selecting jobs if SSJMSSDS is ON.

**SSMJCRP**
- Pointer to a job correlator used to select jobs if SSJMSCOR is ON.

**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>SSJMRTOK (0)</td>
<td>The modify job function call has completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSJMINVA (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the modify job function call.</td>
</tr>
<tr>
<td>SSJMLERR (8)</td>
<td>A logic error occurred. See the reason codes that are defined for SSJMRETN.</td>
</tr>
<tr>
<td>SSJMINVT (12)</td>
<td>The call is not a supported call type (SSJMTYPE).</td>
</tr>
</tbody>
</table>

**Output Parameters**
Output parameters for the function routine are:
- SSOBRETN
- SSJMRETN
- IAZSSJM

**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:
### SSI Function Code 85

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSJMRTOK (0)</strong></td>
<td>Input parameters were valid, check SSJMJOBF for output.</td>
</tr>
<tr>
<td><strong>SSJMINVA (4)</strong></td>
<td>The search arguments, though syntactically valid, cannot be used (for example, specifying a volume serial in SSJMVOL that is not being used as a SPOOL volume).</td>
</tr>
<tr>
<td><strong>SSJMLERR (8)</strong></td>
<td>Logic error in one of the search arguments. See output parameter SSJMRETN (below) for details as to the exact error.</td>
</tr>
<tr>
<td><strong>SSJMINVT (12)</strong></td>
<td>The request type in SSJMTYPE is not valid.</td>
</tr>
</tbody>
</table>

**SSJMRETN Contents:** When SSOBRETN contains an 8 (SSJMLERR) indicating a logic error, the field SSJMRETN indicates the specific error detected. SSJMRETN 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><strong>SSJMRDST (4)</strong></td>
<td>Destination in SSJMDEST is not valid.</td>
</tr>
<tr>
<td><strong>SSJMRJBL (8)</strong></td>
<td>Low job ID in SSJMJBIL is not valid.</td>
</tr>
<tr>
<td><strong>SSJMRJBH (12)</strong></td>
<td>High job ID in SSMJBIH is not valid.</td>
</tr>
<tr>
<td><strong>SSJMRJLM (16)</strong></td>
<td>The high job ID in SSMJBIH is less than the low job ID in SSJMJBIL.</td>
</tr>
<tr>
<td><strong>SSJMRCLS (20)</strong></td>
<td>Job class in SSJMCLSL or SSJMCLSP is not valid.</td>
</tr>
<tr>
<td><strong>SSJMRVOL (24)</strong></td>
<td>The volume list in SSJMVOL is null or has characters that are not that are not allowed.</td>
</tr>
<tr>
<td><strong>SSJMRJBH (28)</strong></td>
<td>The phase specified in SSJMPHAZ or SSJMPHZP is either not valid or not supported by this subsystem.</td>
</tr>
<tr>
<td><strong>SSJMRQUE (32)</strong></td>
<td>Unable to access job queue.</td>
</tr>
<tr>
<td><strong>SSJMREYE (36)</strong></td>
<td>The eyecatcher in SSJMEYE is not C’S$JMPL’</td>
</tr>
<tr>
<td><strong>SSJMRLEN (40)</strong></td>
<td>The length of the IAZSSJM specified in SSJMLEN is too short.</td>
</tr>
<tr>
<td><strong>SSJMRJBN (44)</strong></td>
<td>The job name in SSJMJOBN or SSJMJBNP is not valid.</td>
</tr>
<tr>
<td><strong>SSJMROWN (48)</strong></td>
<td>The owning user ID in SSJMOWNR is not valid.</td>
</tr>
<tr>
<td><strong>SSJMRSYS (52)</strong></td>
<td>The system name in SSJMSYS is not a valid system name.</td>
</tr>
</tbody>
</table>
SSJMRMEM (56)
The member name in SSJMMEMB is not valid.

SSJMRRCST (60)
SSJMSEL2 specifies to select only non-batch jobs and batch job class
selection was specified in SSJMSCLS.

SSJMROJB (64)
Original job ID in SSJMOJBI is not valid.

SSJMRSEC (68)
The SECLABEL in SSJMSECL is not valid.

SSJMRORG (72)
The origin node in SSJMORGN is not defined.

SSJMRXEQ (76)
The execution node in SSJMXEQN is not defined.

SSJMRPRI (80)
The priority in SSJMPRIO is not valid for this JES.

SSJMRSCVC (84)
The service class in SSJMSRVC is not valid.

SSJMSSEN (88)
The scheduling environment in SSJMSSEN is not valid.

SSJMRSCT (92)
The SYSOUT token pointed to by SSJMCTKN is not valid.

SSJMRSCR (96)
The SYSOUT owner in SSJMSCRE is not valid.

SSJRSSSD (100)
The SYSOUT destination in SSJMSDES or SSJMSDSP is not valid.

SSJMRSSC (104)
The SYSOUT class in SSJMSCLA or SSJMSCLP is not valid.

SSJMRSXW (108)
The SYSOUT external writer in SSJMSWTR is not valid.

SSJMREJC (112)
SSJ[M]SBI and SSJMSCTK are mutually exclusive.

SSJMRSF (124)
SSJ[MS]FOR is not valid.

SSJMSPR (128)
SSJ[MS]PRM is not valid.

SSJMRSP (132)
Function or filter not supported.

SSJMRIDS (144)
SSJMRIDS indicates SSJ[M]SSDS is set with either SSJ[M]SSLC or SSJ[M]SSNT.

SSJMRWIL (152)
The same non-zero value specified for both SSJ[M]CHR and SSJ[M]ZOMO.

SSJMRCJ (156)
SSI Function Code 85

SSJMRJIP (160)
At least one of the JOBIDs in the list pointed to by SSJMJBNNP is not valid.

SSJMRJIZ (164)
SSJMSJIL is set and either SSJMJBNN or SSJMJBNP is zero.

SSJMRJCR (168)
The job correlator pointed to by SSJMCRP is not valid.

SSJMRJCO (172)
SSJMSCOR is set with SSJMBSI, SSJMSCTK or SSJMSJIL.

SSJMSBEX (184)
SJOB extension address is missing or zero.

SSJMERBP (188)
Storage pointer is not valid.

SSJMERCA (192)
Invalid parameters on a CANCEL (SSJMCANC) request.

SSJMERRE (196)
Invalid parameters on a RESTART (SSJMRST) request.

SSJMERMD (200)
Invalid parameters on a MODIFY (SSJMCHR) request.

SSJMSJF8
Pointer to a list of job feedback (SSJF) elements.

   SSJMSJFP
      31-bit part of SSJMSJF8

SSJMNSJF
Count of the number of job feedback elements (SSJF) returned in the output area.

SSJMOFG1
Output flags associated with the output area returned.

   Bit Name
   Description
   SSJMO1CP
      Job selection indicator:
         ON     Indicates that job selection was performed using a checkpoint version.
         OFF    Indicates that job selection was performed using the live checkpoint data.

Job feedback elements (SSJF)
For each job that matches specified filter requirements, a feedback element is added to the chain pointed to by SSJMSJF8/SSJMSJFP. Each element is composed of a fixed size job feedback element mapped by the SSJF DSECT.

Feedback Element Prefix: The fields in SSJF describe the selected job and indicate the results of the modify request:

   Field Name
   Description
   SSJFEYE
      Eye catcher, set to CSSJF'.
SSI Function Code 85

SSI Function Codes

Address of the next job feedback (SSJF) element (64 bit address).

SSJFNEXT
31-bit part of SSJFNXT8.

SSJFNAME
Job Name.

SSJFJID
Job Identifier.

SSJFJID
Original Job Identifier.

SSJFOUID
Owner user ID.

SSJFMEM
Member name.

SSJFSYS
System name.

SSJFJCOR
Job correlator.

SSJFSTAT
Job processing status indicator.

SSJ_MOK (0)
SYNC request processed successfully or ASYNC request successfully queued for processing.

SSJ_MCL (128)
Class authorization failed.

SSJ_MRD (132)
Reroute destination authorization failed.

SSJ_JLCK (136)
Request ignored, job locked

SSJ_JNTF (140)
Request ignored, job not found.

SSJ_FBID (144)
Bad job ID.

SSJ_FQLO (148)
Failed to locate the job.

SSJ_BJCR (152)
Job correlator mismatch.

SSJ_JNCN (156)
Job not cancellable or purgeable.

SSJ_FPUR (160)
Job not cancellable or purgeable due to being protected.

SSJ_FPCON (164)
Job not cancellable due to conflicting parameters.

SSJ_FSEQ (168)
Job not routable for execution.
SSI Function Code 85

- **SSJFBADJ (172)**
  - Job class not valid.

- **SSJFNSVC (176)**
  - Job service class cannot be changed - not set yet.

- **SSJFBADS EQU 180**
  - Service class not valid.

- **SSJFPOEX (184)**
  - Request failed - job is post-execution.

- **SSJFJOBC (188)**
  - Job change request failed.

- **SSJFISCH (192)**
  - WLM scheduling environment not valid.

- **SSJFIOFF (196)**
  - Request failed - offload device number non-numeric.

- **SSJFBAFF (200)**
  - Request failed - invalid job phase to modify SYSAFF.

- **SSJFBMBR (204)**
  - Request failed - invalid member name to modify SYSAFF.

- **SSJFO0MBR (208)**
  - Request failed - no member affinity left after SYSAFF modification.

- **SSJFNOBR (212)**
  - Request failed - no BERTs.

- **SSJFNOEX (216)**
  - Request failed - job not in execution.

- **SSJFDDIN (220)**
  - Request failed - DDNAME not valid for downlevel member.

- **SSJFINTE (224)**
  - Request failed - internal error, member number in JQE not valid.

- **SSJFIPRI (228)**
  - Request failed - invalid priority value.

- **SSJNOTE (232)**
  - Job not eligible for start job request.

- **SSJSFDRN (236)**
  - Job not eligible for start. System draining.

- **SSJFDUPJ (240)**
  - Job not eligible for start. Duplicate job running.

- **SSJFSENA (244)**
  - Job not eligible for start. Scheduling environment not available.

- **SSJFINDE (248)**
  - Job not eligible for start. Independent mode mismatch.

- **SSJFSPOL (252)**
  - Job not eligible for start. Spool(s) unavailable.

- **SSJFINTP (256)**
  - Request failed - internal processing error. Field SSJFINTC contains the return code from the service routine.
SSI Function Code 85

SSJFEX49 (260)
Job not eligible for start. Exit 49 rejected the job.

SSJFSECL (264)
Job not eligible for start. SECLABEL not available.

SSJFNAFF (268)
Job not eligible for start. No affinity to any active system.

SSJFARMR (272)
Job not eligible for start. ARM restart pending.

SSJFBUSY (276)
Job not eligible for start. Job already executing.

SSJFNBAT (280)
Job not eligible for start. Not a batch job.

SSJFNEXQ (284)
Job not eligible for start. Job not on execution queue.

SSJFNOJ2 (288)
Job not eligible for start. No JES2 matches the job.

SSJFNJ2S (292)
Job not eligible for start. No JES2 can select the job.

SSJFMINL (296)
Job not eligible for start. Minimum z/OS level not available.

SSJFNBBM (300)
Job not executing on broadcast member.

SSJFNBTC (304)
Request failed. Not a batch job.

SSJFNSJB (308)
Request failed. No SJB control block, no journal, or not a batch job.

SSJFINTC
If SSJFSTAT = SSJFINTP, then this is the service routine return code.

SSJFEMSG
Text description of the job processing indicator SSJFSTAT.

SSJFSIZE
Current size of job feedback (SSJF) element.

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

JPSYSNPRF
System Information Prefix Section

JPSYSNINF
JES System Information Section. This section contains one or more of the following entries:
SSI Function Code 85

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:
Field Name  
Description
JPSYXLNG   
Length of all the sections.
JPSYXTYP   
Type of this section.
JPSYXMOD   
Modifier for this section.

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:
Field Name  
Description
JPSYLNG   
Length of the section, including all system information entries.
JPSYTYPE   
Type of this section.
JPSYMOD   
Modifier for this section.
JPSYOENT   
Offset to the first system information entry.
JPSYNENT   
Number of system information entries returned.
JPSYSENT   
Size of a system information entry.

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:
Field Name  
Description
JPSYSYSN   
MVS system name.
JPSYMBRN   
JES2 MAS member name.
SSI Function Code 85

**JPSYSUBS**
- JES subsystem name.

**JPSYCMCL**
- JES command prefix length.

**JPSYCMCH**
- JES command prefix.

**JPSYVERN**
- Version of JES.

**JPSYFLAG**
- Processing flags.

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSYFPRC</td>
<td>Data processed for this system.</td>
</tr>
<tr>
<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>JPSYFSUP</td>
<td>No data returned for this system - not supported.</td>
</tr>
<tr>
<td>JPSYFINA</td>
<td>No data returned for this system because system is not active or cannot be reached.</td>
</tr>
<tr>
<td>JPSYFGLB</td>
<td>Global system in a complex (JES3).</td>
</tr>
<tr>
<td>JPSYPRI</td>
<td>Primary subsystem.</td>
</tr>
<tr>
<td>JPSYFPXQ</td>
<td>PXEQ issued on this member (JES2).</td>
</tr>
<tr>
<td>JPSYFERR</td>
<td>Error accessing data from the system.</td>
</tr>
</tbody>
</table>

**JPSYVERD**
- Version of the data returned from this system.

**JPSYMBNR**
- JES2 MAS member number.

**JPSYJ2PL**
- JES product level.

**JPSYJ2SL**
- JES service level.
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 487. If your subsystem handles installation-defined directed requests, you must identify each function using a function code from 236 to 255. These codes are not broadcast functions. You can also subdivide installation-defined function codes by using subtypes you identify by passing parameters in your SSOB function dependent area.

If you plan to have your subsystem support the MVS-defined function codes, see the specific function code descriptions for requirements on your function routine. The sections that follow describe general considerations for all function routines you write.
Environment

On entry to a function routine, the function routine must save registers using
standard save area conventions.

The register contents on entry to a function routine are:

Register | Contents
---------|---------
Reg 0    | Address of the SSCVT (mapped by the IEFJSCVT macro)
Reg 1    | Address of the SSOB control block passed by the requestor. This is
         | explained in “Subsystem Options Block (SSOB)” on page 7.
Reg 13   | Standard 18-word save area
Reg 14   | Return address
Reg 15   | Entry point address

On exit from a function routine, a function routine must restore registers 0-14 to
the contents on entry using standard exit linkage.

As you write your function routines, be aware of what state and key the function
routine must be in to do its work. Your function routine gets control in the key and
state of the requestor. If your routine requires that it be in a different key or state,
your routine must handle mode and state switching. However, you must reverse
the mode switch before returning control to the SSI because the SSI gets control
back in your routine’s key and state.

Address mode (AMODE) considerations are handled by the SSI system routines.
Other addressability considerations must be handled by the function routine. Any
addresses passed to an AMODE 24 function routine (including the save area) must
be below 16 megabytes. If the subsystem runs in a separate address space, the
function routine must establish cross memory space communication either by SRB
scheduling or cross memory instructions. For an explanation of using multiple
address spaces, see 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.

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 477 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 IEAAPFx, 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 475 or “Enabling Your Subsystem for New Functions” on page 476 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 477 for information on IEFSSVT and see “Deactivating Your Subsystem” on page 479 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

Use a meaningful name for your subsystem to facilitate debugging. 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 472 for more information.

### The primary subsystem

MVS requires that at least one subsystem be defined as a job entry subsystem (JES) to bring jobs into the system. The JES that you select, which can be either JES2 or JES3, is called the primary subsystem. 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.

Note: Your JES primary subsystem name must match the procname that is used to start JES.

If you attempt to IPL without specifying an IEFSSNxx member and IEFSSN00 is not present or does not identify the primary subsystem, the system issues message IEFJ005I (see "Handling Initialization Errors" on page 555) 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 475 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.

Prior to z/OS V1R12, the subsystem initialization routines specified in parmlib member IEFSSNxx were invoked in the sequence they appeared and under a task that never terminated. From z/OS V1R12, the initialization routines are invoked in parallel after the BEGINPARALLEL keyword in parmlib member IEFSSNxx is processed, and no longer run under a permanent task when they are run in parallel. Because of this, you should examine your subsystem initialization routines to see if they allocate resources that will be freed at task termination when previously the resources would have remain held. Resources to consider:
- Data spaces created, but not deleted (particularly CADS)
- Task-related storage obtained, but not released
- ENQ obtained, but not DEQ'd
- ALESERV ADD without DELETE
- ESTAE CREATE without DELETE
- Joining XCF groups without leaving
- Connections to Coupling Facility structures obtained, but not released
- Task-level Name/Tokens created without deleting

For more information, see "Initializing Your Subsystem" on page 472.
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 464 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.

To take advantage of running subsystem initializations in parallel, the IEFSSNxx parmlib member needs to be updated to include the BEGINPARALLEL keyword.

Note: All initialization routines specified before the BEGINPARALLEL keyword are invoked serially. All initialization routines specified after the BEGINPARALLEL keyword are invoked in parallel.

For IBM products or a vendor-supplied subsystem, check the product’s installation or configuration documentation to determine the placement of the BEGINPARALLEL keyword.

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 30 on page 468 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 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) 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.
Setting Up

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 535 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.
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.
Note: The ability to run in parallel is only in the keyword format of the IEFSSNxx parmlib member.

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 z/OS MVS Data Areas in the z/OS Internet Library (<a href="http://www.ibm.com/systems/z/os/zos/bkserv/">http://www.ibm.com/systems/z/os/zos/bkserv/</a>) for the format of JSIPL.</td>
</tr>
</tbody>
</table>
Figure 31 shows the input to the initialization routine when your initialization routine gets control from the system.

![Diagram](image)

**Figure 31. Input to the Initialization Routine**

**Coding the Initialization Routine**

Before coding your initialization routine, consider:

- You can set up a control block structure for your subsystem by building a control block to hold any necessary information and anchoring that control block with the put/get function of the IEFSSI macro. See "Storing Subsystem-specific Information" on page 480 and "Retrieving Subsystem-specific Information" on page 481 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 481 for more information on the options function of the IEFSSI macro.

If you have chosen to have your subsystem initializations run in parallel, update the IEFSSNxx parmlib member to include the BEGINPARALLEL keyword. Remember that all initialization routines specified before the BEGINPARALLEL keyword are invoked serially and all routines specified after the BEGINPARALLEL keyword are invoked in parallel. The BEGINPARALLEL keyword must be specified after the SMS entry.

- Your initialization routine determines whether the subsystem can respond to the SETSSI command by using the options function of the IEFSSI macro. See "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 if you are doing serial processing.
- 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.

Prior to z/OS V1R12, the subsystem initialization routines specified in parmlib member IEFSSNxx were invoked in the sequence they appeared and under a task that never terminated. From z/OS V1R12, the initialization routines are invoked in parallel after the BEGINPARALLEL keyword in parmlib member IEFSSNxx is processed, and no longer run under a permanent task when they are run in parallel. Because of this, you should examine your subsystem initialization routines to see if they allocate resources that will be freed at task termination when previously the resources would have remain held. Resources to consider:

- Data spaces created, but not deleted (particularly CADS)
- Task-related storage obtained, but not released
- ENQ obtained, but not DEQ’d
- ALESERV ADD without DELETE
- ESTAE CREATE without DELETE
- Joining XCF groups without leaving
- Connections to Coupling Facility structures obtained, but not released
- Task-level Name/Tokens created without deleting

“Example 1 — Subsystem Initialization Routine (TSYSINIT)” on page 561 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 575 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 466.
- 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 463 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 480 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 461 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 575 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 475 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 481 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 475.
- 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.

**Reactiving a Subsystem after Deactivation**

Use the activate request or the SETSSI ACTIVATE command to reactivate a deactivated subsystem. A subsystem can be activated, deactivated and reactivated as many times as is necessary.

**Deactivating Your Subsystem**

To deactivate your subsystem, you can use either:

- The IEFSSI macro
- The SETSSI DEACTIVATE command.

Use the deactivate request of the IEFSSI macro or the SETSSI DEACTIVATE command to deactivate your subsystem so that your subsystem can suspend operations or stop responding to SSI function requests. The SSI stops routing requests, including broadcast requests, to the subsystem when it receives the deactivation request or command. However, there may be outstanding function requests that have not completed. Since it is not possible to determine when the outstanding requests complete, subsystems must not attempt to delete function routines or other resources that might still be in use after either the deactivate request or SETSSI DEACTIVATE command has been issued.

**Note:** If a job requires the use of paired subsystem function requests, such as, allocate/unallocate or open/close, the job may not end as expected if the subsystem processing these requests is deactivated when the first request of the pair has been processed but the second has not. The SSI cannot determine if this situation exists. It is both the installation's and the subsystem's responsibility to control the job sequence and subsystem deactivation requests to avoid potential problems.

Outputs: The deactivate request returns a vector table token to its caller in the location identified by the optional OUTTOKEN parameter. The token represents the SSVT that has been deactivated. You can use the token in subsequent activate requests, if the same set of functions is supported when it is reactivated. The vector table token is output only. A deactivate request always applies to the active subsystem vector table.

A deactivate request or command is processed only if the target subsystem is dynamic, even if the active vector table is not managed by the SSI. In this case, the output token contains a zero and the request receives the IEFSSI_WARNING (4) return code.

**Note:** If the subsystem does not have vector tables managed by the SSI, the subsystem cannot be reactivated dynamically.
Swapping Subsystem Functions

A subsystem can maintain two subsystem vector tables. The two tables can describe different sets of functions to which the subsystem responds or identify different function routines to be invoked for the same function codes.

A subsystem would find it useful to maintain two subsystem vector tables if, for example, a subsystem must quiesce operations. This way, a subsystem can keep one full-function vector table and a second limited-function vector table, and swap so that it can continue to support some minimum set of function while shutting down.

The swap request of the IEFSSI macro allows the subsystem to deactivate the active vector table and activate the inactive table in a single operation. The swap request eliminates the need for separate deactivate and activate requests, which would result in a period of time when the subsystem cannot respond to requests.

Inputs

The swap request allows the user to specify a subsystem vector table token on input. The input token, which is named with the INTOKEN parameter, identifies the vector table that is to be activated (with the activate request or command). If INTOKEN is not specified, the inactive (previously created) vector table is activated.

Outputs:

The swap request allows the user to specify a subsystem vector table token on output. On completion of the swap, the output token, which is named with the OUTTOKEN parameter, identifies the outgoing (previously active) vector table.

If the subsystem is initially inactive, the swap request receives the IEFSSI_WARNING (4) return code and is treated as an activate request. The output token identified with the OUTTOKEN parameter contains a zero. If the outgoing (initially active) vector table is not managed by the SSI, the output token contains a zero and the request receives the IEFSSI_WARNING return code.

Storing and Retrieving Subsystem-specific Information

To store and retrieve subsystem-specific information, you can use the IEFSSI macro. A subsystem or a subsystem initialization routine needs to be able to pass information to the subsystem's function routines. If the subsystem code and its function routines are in separate load modules or run in separate address spaces, there may be no direct way for the subsystem to communicate with its function routines. The store and retrieve services provide a way for subsystems to store and retrieve subsystem-specific information and pass that information between subsystem components.

Storing Subsystem-specific Information

Use the put request of the IEFSSI macro to store subsystem-specific information. The put service allows a subsystem to store a total of 8-bytes of subsystem-specific information in two non-contiguous 4-byte fields, which are identified by the SUBDATA1 and SUBDATA2 parameters. The user can store the data in either or both of the two fields on a single invocation of the put service.
A typical use of the put service is to store a pointer to a subsystem-specific control block, which the subsystem initialization routine created and made available for use by the subsystem function routines.

IBM recommends that your subsystem create and anchor control blocks to store subsystem data, even if the stored data is small enough to fit within the two fields provided. This lets your subsystem store more information at a later time. In addition, the information stored using this service does not reside in fetch-protected storage. However, the subsystem can create its control block in a fetch-protected subpool.

**Retrieving Subsystem-specific Information**

Use the get request of the IEFSSI macro to retrieve subsystem-specific information. The get service allows a subsystem to retrieve subsystem-specific information that was stored using the put request. The retrieved information, which is identified by the SUBDATA1 and SUBDATA2 parameters, is the information that was originally identified by the corresponding put service parameter.

**Defining Subsystem Options**

To define subsystem options, you can use the IEFSSI macro. The options request allows a subsystem to specify:

- Whether it responds to the SETSSI command
- The subsystem (MSTR or primary) under which the subsystem is to be started.

Use

You can invoke the options request more than once for a single subsystem. The most recent invocation of the service determines the characteristics of the subsystem. The first time the service is invoked, the defaults described in the IEFSSI macro are effective for parameters that are not specified. See [z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG](https://publib.boulder.ibm.com/infocenter/zos/v1r13/topic/com.ibm.zos.mus.fskm001.doc/25fasug/25fasug.htm) for more information on the IEFSSI macro. For subsequent invocations, characteristics corresponding to omitted parameters retain their most recent value. For example, if the first invocation does not specify the COMMAND parameter, the default of COMMAND=NO is used. However, if the first invocation specifies COMMAND=YES and a second invocation does not specify the COMMAND parameter, the subsystem continues to respond to the SETSSI command as specified by the first invocation.

**Responding to the SETSSI Command**

The system does not process the SETSSI command directed to subsystems that have not explicitly authorized the commands, because existing subsystem were not designed for the possibility of dynamic manipulation by commands. The system may be disrupted if these subsystems are manipulated unexpectedly by commands.

**Starting Your Subsystem Under the Primary Subsystem**

A subsystem may require the services of the primary subsystem when being started. For example, it may require the primary subsystem to provide the use of subsystem data sets or an internal reader. The options service specifies whether the subsystem being added requires the primary subsystem, and is intended for use in a subsystem initialization routine.
If the START command does not specify the subsystem under which the target subsystem should start, the system uses the information specified with the REQDSUB parameter of the options request.

## Querying Subsystem Information

To query subsystem information, an application can use the IEFSSI macro or an operator can use the DISPLAY SSI command. The query request allows either an application or the operator to query the following information for all subsystems defined to the SSI:

- The subsystem name
- If the subsystem is dynamic or not dynamic
- If the subsystem is the primary subsystem
- If the subsystem is active or inactive
- If the subsystem is dynamic, whether it accepts or rejects dynamic SSI commands
- If the subsystem is active, which function codes it supports.

An application can also query the following additional information:

- The number of vector tables associated with the subsystem, with a maximum of two vector tables.
- The following information for each associated vector table:
  - If the vector table is managed by the SSI. A vector table managed by the SSI is a vector table created with the IEFSSVT REQUEST=CREATE macro.
  - A locator. This locator is a token if the vector table is managed by the SSI and is an address if the vector table is not managed by the SSI.
  - If the vector table is active.
  - The function codes supported by the vector table.

This information represents a snapshot of the subsystems defined to the SSI when you process the query request.

To obtain information about the primary subsystem without knowing its name, use the query request and specify a subsystem name of !PRI.

## Using the Subsystem Query Request of the IEFSSI Macro

The query request of the IEFSSI macro is the only service provided by this macro that does not require the caller to be authorized.

### Inputs

The SSI obtains the storage necessary to return the query request information, because the issuer of the query request cannot determine in advance how much information will be returned. The issuer of the query request can use the WORKASP parameter to specify the subpool in which the SSI can obtain the storage. The query request fails if the SSI is unable to obtain enough storage. Unauthorized callers are limited to unauthorized subpools.

The query request returns information either for a single subsystem or for all subsystems matching the pattern specified with the SUBNAME parameter. The pattern can contain the following wildcard characters:

- An asterisk (*) — matches zero or more characters
A question mark (?) — matches one character.

Outputs: The mapping macro IEFJSQRY maps the output returned by the query request.

If the SSI obtains the storage it needs to use the query request, the SSI returns the address of the output work area in the variable that the WORKAREA parameter identifies. The JQRYLEN field mapped by the IEFJSQRY macro contains the length of the returned storage. Upon completion, the issuer of the IEFJSQRY macro must free the returned storage. You should have established a recovery routine to free the returned storage in case your program ends abnormally. IBM recommends you use task-oriented or job-oriented storage to ensure that the storage is released upon task or job completion.

If you request information about multiple subsystems, the output lists the information in broadcast order. That is, the subsystems are listed in the same order in which SSI broadcast processing invokes them. For each subsystem, the IEFSSI query request returns information about all associated vector tables managed by the SSI, active or not. For vector tables that are not managed by the SSI, the system locates only the active vector table and returns information about that vector table only.

A query request may fail to return information about some subsystems. If a subsystem is defined after IPL by directly manipulating the SSI control blocks and the definition either occurs during the processing of the query request or is not correctly completed, some subsystems may not be represented in the response to the query request.

Using the Display SSI Command

The DISPLAY SSI command displays status information about all subsystems defined to the SSI. You can request information for all subsystems at once or for those subsystems which meet the criteria specified by the filters used when issuing the DISPLAY SSI command. You can use filters to limit the information displayed to:

• One particular subsystem or those subsystems whose names match a specified pattern
• Subsystems that are either dynamic or not dynamic
• Subsystems that are either active or not active
• Subsystems that respond to a given list of function codes.

In addition, the issuer of the command can use the LIST or ALL keywords to specify whether to display subsystem function codes. Subsystem information is displayed in broadcast order.

Maintaining Information About the Callers of Your Subsystem

A common requirement for a subsystem is to maintain information specific to each of its callers. To accomplish this, a subsystem needs both:

• A method of uniquely identifying each caller.
• A work area to store information about each caller (or a place to store the address of a work area).
The subsystem affinity service solves both of these requirements. It allows a subsystem to store and retrieve data at the task control block (TCB) level, thus removing its dependence on information passed by callers.

Consider the following example: A subsystem provides service to many callers, and must also maintain use counts by caller. Each caller can be identified by the TCB that is associated with it.

The subsystem uses the subsystem affinity service to maintain a separate use count for each of its callers. For each caller, the subsystem affinity service provides the subsystem with a unique fullword entry, called a subsystem affinity entry. Figure 32 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 32. Subsystem Affinity Service**

**Accessing the Subsystem Affinity Entry:** To access the subsystem affinity entry for each of its callers, a subsystem needs to:

- Invoke the verify subsystem function (SSI function code 15) through the IEFSSREQ macro to acquire its subsystem affinity index. See “Verify Subsystem Function Call — SSI Function Code 15” on page 44 for information on SSI function code 15.
- Issue the SSAFF SET request to store data in the entry.

On subsequent invocations, the subsystem can issue the SSAFF OBTAIN request to retrieve the address of a work area from the entry.

**SSAFF: Set/Obtain Subsystem Affinity**

Use the SSAFF macro to SET or OBTAIN a subsystem affinity entry.

An SSAFF SET request places one fullword of subsystem passed data in the subsystem affinity entry, which is identified by the TCB parameter and the subsystem affinity index. This allows the subsystem to put its entry in the subsystem affinity entry of the current, active TCB.

An SSAFF 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](https://www.ibm.com/support/docview.wss?uid=dfs00028).

Before you issue the SSAFF macro, register 13 must point to an 18-word save area.

The syntax of the SSAFF macro is:

```
[symbol] SSAFF {SET [,TCB=tcb-address]} {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.

OBTAIN requests have the following requirements:
- The caller must be in task mode. If this condition is not met, the calling routine abends.
- The caller must have current addressability to the home address space.

The SSAFF macro parameters have the following meanings:

- **symbol** any valid assembler language symbol.
- **SET** indicates that MVS is to place the value specified by the DATA parameter into the subsystem’s associated subsystem affinity entry. The SET request destroys the contents of registers 14, 15, 0, 1, and 2.
- **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:
- 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,” on page 561 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.
SSI Function Code 4

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: Authorized Assembler Services Guide.

Issued to

- All active subsystems that indicate they support the End-of-Task function when the system (MVS) issues the End-of-Task call.

Related SSI Codes

SSI function code 4 is similar to SSI function code 50 (Early End-of-Task call). The only difference is that, for SSI function code 4, your routine is given control after most resource managers are given control. For SSI function code 50, your routine is given control before most resource managers are given control. If you want to obtain control before most resource managers have been invoked, see SSI function code 50 (Early End-of Task).

Related Concepts

None.

Environment

Review “Function Routines/Function Codes” on page 461, 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 575 and “Enabling Your Subsystem for New Functions” on page 580 for more information.

The subsystem function routine runs in the address space of the ending task. Because each subsystem function routine is called for every ending task, the subsystem function routine should not be a long running program. That is, the function routine should quickly determine if the subsystem was ever associated with the ending task and, if not, return to the system. Also, do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT. Entering a WAIT can cause degraded system performance.

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

- IEFSSOBH
- IEFJSSIB
- IEFSSET

The function routine receives control in the following environment:
SSI Function Code 4

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 33 shows the environment on entry to the function routine for SSI function code 4.

**Figure 33. 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>
SSI Function Code 4

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 (SSOBSIZE) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 4 (SSOBFUNC)</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. 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 &quot;Output Register Information&quot; on page 491 for a list of possible SSOBRETN return codes.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the 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 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>
</tbody>
</table>
SSI Function Code 4

- SSTYPE 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 (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>

**End-of-Address Space (End-of-Memory) Call — SSI Function Code 8**
The End-of-Address Space Function (End of Memory) call (SSI function code 8) provides the ability to free up any system-level resources, such as CSA, obtained by a subsystem on behalf of an address space. Whenever an address space ends, all active subsystems that are enabled to receive SSI function code 8 are given control from the SSI. The function routine gets control for every address space that ends.

**Type of Request**
Broadcast SSI call.

**Use Information**
Your subsystem can use SSI function code 8 to clean up any system-level resources which your subsystem obtained for one or more address spaces. Because private storage for the address space has already been deleted, your function routine must not reference any storage in the ending address space.

Because your function routine gets control for every address space that ends, using your own subsystem may not be the most efficient way to do your own clean up for ending address spaces. IBM recommends that you define your own resource manager through the use of the RESMGR macro. You can use RESMGR to receive control for specific ending address spaces, rather than having to check each ending task or address space to see if it used the subsystem. For a general description of
resource managers and how they can be defined at both IPL time and dynamically, see [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com/systems/ibm/z/mvs/services_guide).

**Issued to**

- All active subsystems that indicate they support the End-of-Address space function when the system (MVS) issues the End-of-Address space call.

**Related SSI Codes**

None.

**Related Concepts**

None.

**Environment**

Review [“Function Routines/Function Codes” on page 461](https://www.ibm.com/systems/ibm/z/mvs/services_guide), 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 575](https://www.ibm.com/systems/ibm/z/mvs/services_guide) and [“Enabling Your Subsystem for New Functions” on page 580](https://www.ibm.com/systems/ibm/z/mvs/services_guide) 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
- IEFSSIB
- IEFSSSEN

The subsystem function routine receives control in the following environment:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 SSOB, SSIB, and SSEN control blocks reside in storage below 16 megabytes.</td>
</tr>
</tbody>
</table>
The function routine 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 34 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:

**Register** | **Contents**
--- | ---
0 | Address of the subsystem’s SSCVT
1 | Address of the SSOB control block
13 | Address of a standard 18-word save area
14 | Return address
15 | Entry point address

### Input Parameters
Input parameters for the function routine are:
- SSOB
SSI Function Code 8

- 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>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLLEN</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 (SENSIZE) 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>- <strong>SSENTYPE ON</strong> — indicates an abnormal ending address space</td>
</tr>
<tr>
<td></td>
<td>- <strong>SSENTYPE OFF</strong> — 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 (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>

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:

```assembly
:  
  * Register declarations
  R1 EQU 1 Declaration for register 1
  R13 EQU 13 Declaration for register 13
  R15 EQU 15 Declaration for register 15
  
  DOBRDCST EQU * Request broadcast of WTO/WTORs
  LA R1,SCSRPLST Get addressability to SCSR
  ST R1,SCSRPTR Save pointer for standard linkage
  XC SCSRPLST(SCSPLEN),SCSRPLST Zero out parameter list
  MVC SCSACRO,SCSRACRN Set acronym value
  MVI SCSVER,SCSVERSN Set version level
  OI SCSFUNC1,SCSBRDON Indicate to broadcast WTO/WTORs
```
SSI Function Code 9

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 \textit{z/OS MVS Planning: Operations}
- Installation-written exit routines — see \textit{z/OS MVS Installation Exits}
- Automation — see \textit{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 \textit{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 461](#), 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 575](#) and [“Enabling Your Subsystem for New Functions” on page 580](#) for more information.

WTOs occur frequently on MVS. Function routines should therefore be as efficient as possible. Function routines should never enter a WAIT and should never use system services that have implied WAITs (such as I/O). Entering a WAIT can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:
- IEFSSOBH
- IEFJSSIB
- IEFSSWT
- IHAWQE
- IHAORE
SSI Function Code 9

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:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 SSOB, SSIB, SSWT, and WQE control blocks reside in storage below 16 megabytes. The ORE control block resides above 16 megabytes.</td>
</tr>
</tbody>
</table>

Recovery

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 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 SSORETN 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>
SSI Function Code 9

**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>
</tbody>
</table>
SSW Function Code 9

SSWTSNMOD
Value of SUBSMOD keyword on the WTO macro

SSWTPRSP
Indicates whether the SSWTPRTY field is valid

Note: Because messages are no longer assigned priorities, this field is ignored. It remains present for compatibility purposes only.

SSWTPRTY
Value of the PRTY keyword on the WTO macro

Note: Because messages are no longer assigned priorities, this field is ignored. It remains present for compatibility purposes only.

SSWTSNSP
Indicates whether the JOBNAME keyword was specified on the WTO

SSWTSISP
Indicates whether the JOBID keyword was specified on the WTO

Figure 35 shows the 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>WQEWTOR</td>
<td>— indicates the message is a WTOR</td>
</tr>
</tbody>
</table>

Figure 35. Environment for a Single-line WTO in the SSWT Control Block
SSI Function Code 9

- **WQEAUTH** — indicates the message is issued by an authorized program.

**WQEASID**
ASID of the message issuer

**WQETCB**
TCB address of the message issuer

**WQESEQ#**
Message DOM id

**WQEMCSF1**
Indicators
- **WQEMCSA** — indicates the WQEROUT and WQEDESCD fields are valid
- **WQEMCSB** — indicates the WQECNID and WQECNNME fields are valid
- **WQEMCSC** — indicates the message is a command response
- **WQEMCSD** — indicates the WQEMSGTP field is valid
- **WQEMCSE** — indicates the message is reply to WTOR
- **WQEMCSF** — indicates BRDCAST was specified on the WTO
- **WQEMCSG** — indicates HCONLY was specified on the WTO.

**WQEMCSF2**
Indicators
- **WQEMCSM** — indicates the message is a hardcopy image of the operator command
- **WQEMCSN** — indicates that NOCPY was specified on the WTO.

**WQEMSGTP**
Message type

**WQEROUT**
Routing codes

**WQEFLG1**
Indicators
- **WQERETAN** — indicates that the message is retained by AMRF
- **WQENMOD** — indicates the subsystem cannot modify the message
- **WQEPNPA** — non-action message issued by a non-authorized program
- **WQERISS** — indicates the message is an SVC reissue of a message that has already been processed by SVC. WTO MPF and the SSI have already processed the message. Note that MPF processing occurs only during the original SVC WTO.

**WQEDESCD**
Descriptor codes

**WQEJSTCB**
Address of the job step TCB

**WQEVRSN**
Version level — contains the WQEVRID

**WQESYSNM**
System name
SSI Function Code 9

WQEXMOD
Copy of the MPF/IEAVMIXIT user exit request flags. Indicator:
- **WQERDSTM** — 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.

WQEMLVL
Message level

WQEERC
Extended routing codes

WQELENG
Length of WQE — contains the WQESIZE

WQEKEY
Value of the KEY keyword on the WTO

WQETOKN
Value of the TOKEN keyword on the WTO

WQECNID
Console ID

WQEOBJID
Originating job ID

WQEOJBNM
Originating job name

WQEAUTOT
Value of the automation token from MPF

WQEERFS
Extended request flags from the MPF/IEAVMIXIT user exit

WQECNNME
Console name

WQECART
Value specified on the CART keyword on the WTO

WQEBENIP
Indicators
- **WQEDOMD** — indicates the message has been deleted by the DOM macro.
- **WQENBEW** — indicates the message created by 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.
- **WQENHABD** — indicates the message has been displayed on the IPL or system console. This is a result of issuing a WTO with SYNCH=YES specified.

WQECASEL
Message color

WQEHASEL
Message highlighting

WQEIASEL
Message intensity
SSI Function Code 9

WQEMISC
Indicator
  • WQEAUTO — indicates AUTO(Y) specified in the MPF for this message.

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” 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 36 shows the environment for the first line of a multi-line WTO in the SSWT control block.

![Figure 36. Environment for the First Line of a Multi-line WTO in the SSWT Control Block](image)

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></td>
<td>• WMJMMLWB — 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>
</tbody>
</table>
SSI Function Code 9

WMJMDSP
Indicators
- WMJMDSPH — indicates the message issued by an authorized program

WMJMASID
ASID of the message issuer

WMJMTCB
TCB address of the message issuer

WMJMSEQ#
Message DOM id

WMJMMCS1
Indicators
- WMJMC1A — indicates that the WMJMRTC and WMJMDEC fields are valid
- WMJMC1B — indicates that the WMJMCNID and WMJMCNME fields are valid
- WMJMC1C — indicates the message is a command response
- WMJMC1D — indicates the WMJMMT field is valid
- WMJMC1E — indicates the message is a reply to the WTOR
- WMJMC1F — indicates the BRDCAST keyword is specified on the WTO
- WMJMC1G — indicates the HCONLY is specified on the WTO

WMJMMCS2
Indicators
- WMJMC2E — indicates the message is the hardcopy image of the operator command
- WMJMC2F — indicates the NOCPY is specified on the WTO

WMJMLTY1
Line type indicator
- WMJMLTYA — control line
- WMJMLTYB — label line
- WMJMLTYC — data line
- WMJMLTYD — end line

Note: The WMJMLTYC and WMJMLTYD fields can be on at same time.

WMJMRTC
Routing codes

WMJMFGL1
Indicator
- WMJMRETN — indicates the message will be retained by AMRF
- WMJMNNMOD — indicates the subsystem cannot modify the message
- WMJMPPNA — indicates the message is issued by the problem program
- WMJMRISS — indicates the message is an SVC reissue of a message that has already been processed by SVC WTO. MPF and the SSI have already processed the message. Note that MPF processing occurs only during the original SVC WTO. Examples of using this indicator include
messages that originate on one system (MVS sysplex), but are transported for display to another system (JES3 complex).

**WMJMDDEC**  
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.

**WMJMLMLVL**  
Message level

**WMJMMERC**  
Extended routing codes

**WMJMLENG**  
Length of WQE — contains the WMJMSIZE

**WMJMKVEY**  
Value of the KEY keyword on the WTO

**WMJMTOKN**  
Value TOKEN keyword on the WTO

**WMJMCNID**  
Console ID

**WMJMOJBI**  
Originating job ID

**WMJMOJBN**  
Originating job name

**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

**WMJBENIP**  
Indicators
- **WMJMDOMD** — indicates the message has been deleted by the DOM macro.
- **WMJMNBEW** — indicates the message created by the branch-entered WTO. Branch-entered WTOs are WTOs that MVS has called for
SSI Function Code 9

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>WMNMLT1B</td>
<td>— label line</td>
</tr>
<tr>
<td>WMNMLT1C</td>
<td>— data line</td>
</tr>
<tr>
<td>WMNMLT1D</td>
<td>— end line</td>
</tr>
</tbody>
</table>

**Note:** The WMNMLT1C and WMNMLT1D fields can be on at same time.

**WMNMLTH1**
Length of the minor WQE

**WMNMTL1**
Length of the minor text

**WMNMTXT1**
Minor line text

**WMN1XMOD**
Copy of the request flags from the MPF/IEAVMXIT user exit

**Figure 37 on page 507** shows the environment for minor lines of a multi-line WTO in the SSWT control block.
Multiline use information: The following example illustrates the method in which the system presents a multiline message to the SSI for updating. Consider the six-line message, IEE889I:

```
IEE889I 09.38.52 CONSOLE DISPLAY 191
/SF5800002/SF590000
MSG: CURR=0 LIM=1500 RPLY:CURR=0 LIM=10 SYS=SCOTT01 PFK=01
/SF5800003/SF590000
CONSOLE ID --------------- SPECIFICATIONS ---------------
/SF5800004/SF590000
SYSLOG COND=H AUTH=CMDS NBUF=N/A
/SF5800005/SF590000
ROUTCDE=ALL
/SF5800006/SF590000
LOG BUFFERS IN USE: 0 LOG BUFFER LIMIT: 1000
```

A multiline 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 14.

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 14. 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>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 37. Environment for Minor Lines of a Multi-line WTO in the SSWT Control Block

IEFSSWT

<table>
<thead>
<tr>
<th></th>
<th>MAJOR WQE</th>
<th>MINOR WQE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTWQE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSWTMIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSWTORE = 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chapter 6. SSI Function Codes Your Subsystem Can Support 507
SSI Function Code 9

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, WMNMXN1 (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" 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>

**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" 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 38 on page 509 shows the 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 (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTRTOK (0)</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
<tr>
<td>SSWTNDSP (4)</td>
<td>Do not display message; hardcopy message</td>
</tr>
<tr>
<td>SSWTOKNH (8)</td>
<td>Display message; do not hardcopy message</td>
</tr>
<tr>
<td>SSWTNDNH (12)</td>
<td>Do not display message; do not hardcopy message</td>
</tr>
</tbody>
</table>

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:
SSI Function Code 9

Field Name  
Description  

**WQETXTLN**  
New length of the message text (if text was altered)

**WQETXT**  
New/changed message text

**WQEMSGTP**  
New/changed message type. The WQEMCSD field must be set appropriately.

**WQEROUT**  
New/changed routing codes. The WQEMCSA field must be set appropriately.

**WQEDESCD**  
New/changed descriptor codes. The WQEMCSA field must be set appropriately. The WQEROUT field must be non-zero.

**WQEERC**  
New/changed extended routing codes

**WQECASEL**  
New/changed message color

**WQEHASEL**  
New/changed message highlighting

**WQEIASEL**  
New/changed message intensity

**WQEFLG1**  
Indicator
- **WQERETAN** — indicates that the message is retained by AMRF

**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.

WQE Contents for a Multi-line WTO (major line): The contents of the following fields in the WQE control block for a multi-line WTO (major line) on output are:

Field Name  
Description  

**WMJMTXTL**  
New length of the message text (if text was altered)

**WMJMTXT**  
New/changed message text

**WMJMXMOD**  
Copy of the MPF/IEAVMXIT user exit request flags

**WMJMMT**  
New/changed message type. The WMJMCSD1D field must be set.

**WMJMRTC**  
New/changed routing codes. The WMJMCSD1A field must be set.
SSI Function Code 9

WMJMDEC
New/changed descriptor codes. The WMJMCS1A field must be set. The
WMJMRTC field must be non-zero.

WMJMERC
New/changed extended routing codes

WMJCASEL
New/changed message color

WMJHASEL
New/changed message highlighting

WMJIASEL
New/changed message intensity

WMJMRTDM
Indicates whether the message is to be deleted

WMJMRETN
Indicates that the message is retained by AMRF

ORE Contents for a WTOR: 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:

Field Name Description

WMNMTL1
New length of the minor text (if the WMNMTXT1 field is modified)

WMNMTXT1
New/changed minor line text

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.
SSI Function Code 10

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 461, 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 575 and “Enabling Your Subsystem for New Functions” on page 580 for more information.

Do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT. Entering a wait can cause degraded system performance.

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

- IEFSSOBH
- IEFJSSIB
- IEFSSCM
- IEZMGCR

The function routine receives control in the following environment:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 SSOB, SSIB, and SSCM control blocks reside in storage below 16 megabytes.</td>
</tr>
</tbody>
</table>
SSI Function Code 10

Environment variable Recovery

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.

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 516 for instructions on what your function routine should specify to the system.

*Figure 39* shows the environment on entry to the function routine for SSI function code 10.

*Figure 39. Environment on Entry to the Function Routine for SSI Function Code 10*
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
- SSIB
- SSCM
- Command Sensitive Area
- MGCRPL

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>SSOBFSRC</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 (SCCM 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 SCCM control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
SSI Function Code 10

SSCMLEN
Length of the SSCM (SSCMSIZE) control block

SSCMVRSN
Version level of the SSCM (SSCMVRID) control block

SSCMBUFF
Address of the command buffer in the MGCRPL control block

SSCMACRN
Identifier 'SSCM'

SSCMAUTH
Command authority of command issuer — see the SSCM control block information for the definition of flags within this byte.

SSCMDISP
Disposition flags — see the SSCM control block information for the definition of flags within this byte.

SSCMBLEN
Length of the command buffer pointed to by the SSCMBUFF field.

SSCMOLIB
If the command text was changed by symbolic substitution (indicated by an ON value in the SSCMSYMS field), this field contains a DSECT that maps the original command text (the text that existed before symbolic substitution occurred).

SSCMOLIP
If the command text was changed by symbolic substitution (indicated by an ON value in the SSCMSYMS field), this field contains the address of the SSCMOLIB structure.

SSCMSYMS
The command text was changed by symbolic substitution.

SSCMUTOK
Address of the UTOKEN
The UTOKEN identifies the issuer of the command. The RACROUTE macro accepts the UTOKEN to perform command authorization checking using a security product (RACF).

SSCMULTH
Length of the UTOKEN

SSCMCNID
4-byte console ID — identifies the console that the command was issued from.

SSCMSCNM
Console name of the console whose ID is in the SSCMCNID

SSCMCTX
Address of a 126-byte buffer containing the command text

SSCMCLEN
Length of command text

SSCMCART
Command and response token
SSI Function Code 10

To identify the source of the command, all command responses issued by a function routine through a WTO or WTOR should specify either SSCMCNID or SSSCNM and SSCMCART.

SSCMCXPT
Address of command sensitive area or zero

Command Sensitive Area Contents: MVS sets the following fields in the command sensitive area for a REPLY command on input. The address of this area (when present) is available in the SSCMCXPT field. If not present, SSCMCXPT=0.

<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 SSCMCTXT 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>MGCRLGTH</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:

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSCMSCMD (0)</td>
<td>The command does not belong to the function routine.</td>
</tr>
<tr>
<td>SSCMSUBC (4)</td>
<td>The command belongs to the function routine and was processed.</td>
</tr>
<tr>
<td>SSCMIMSG (8)</td>
<td>The command belongs to the function routine, but could not be processed. Message IEE707I is issued.</td>
</tr>
</tbody>
</table>

**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>MGCRNLGTH</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, MGCRNLGTH (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.

**Note:**

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/support/docview.wss?rs=131&id=redbooks:112157) 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) 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.

• Using the SSCMAUTH field. Use the flag settings of the SSCMAUTH field to test the command authority of the caller. This field is mapped by the UCMAUTH field in the UCME (IEECUCM).

• Using the SSCMCART field. All command response messages issued through a WTO should use the values passed in the SSCMCNID field (above) and in the SSCMCART field. The use of these values ensures proper delivery of the message to the command issuer.

Considerations for Command Processing Calls in a Sysplex:

In a sysplex, command processing SSI calls are made to subsystems:

• On the originating console's system only, when the command is not routed to any other system in the sysplex.

• On the originating console's system only, when the command is routed to another system in the sysplex as the result of the location (L=) operand on the command or the specification of a console by name.

• On the receiving system only, when it is a prefix command that is routed through the MCS command prefix facility.

• On both the originating system and the receiving system, when the ROUTE command is issued, as follows:
  – On the originating system for the ROUTE command.
  – On the receiving system for the command that is routed.

Considerations for Commands That Specify System Symbols:

When a command contains system symbols, MVS provides the command text to the SSI after it substitutes text for the system symbols. For example, if the following command is entered to display a console group on system SYS1:

```
DISPLAY CNGRP,G=(CN1GRP&SYSCLONE.)
```

The SSI receives the following text (assuming that the default for &SYSCLONE, the last two characters of the system name, is taken):

```
DISPLAY CNGRP,G=(CN1GRPS1)
```

If the function routine requires the original command text (the one that existed before symbolic substitution), it can access the SSCMOLIB field in the SSCM (see z/OS MVS Data Areas in the z/OS Internet library (http://www.ibm.com/systems/z/os/zos/bkserv/) 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 Z/OS MVS Programming: Authorized Assembler Services Guide.

Environment
Review “Function Routines/Function Codes” on page 461, 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 575 and “Enabling Your Subsystem for New Functions” on page 580 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
- IEFSSIB
- IEFSSOB
- IEFSSDM
- IHADOMC

The delete operator message mapped by IHADOMC represents a DOM.
SSI Function Code 14

The function routine receives control in the following environment:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 SSOB, SSIB, and SSDM control blocks reside in storage below 16 megabytes.</td>
</tr>
</tbody>
</table>

Recovery

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 DOM if an abend occurs.

The function routine's recovery should specify a retry point (address) and return 4 on the SETRP macro before returning to the system. Use the retry point to complete a normal return to the function routine's caller. When the function routine returns to its caller under these circumstances, it should indicate to the system, by setting both register 15 and the SSOBRETN to zero, to take no action against the message. See the next topic, Input Register Information, for how to specify to the system the action you want your function routine to take.

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>
</tbody>
</table>
SSI Function Code 14

**SSOBLEN**
Length of the SSOB (SSIBHSIZ) control block

**SSOBFUNC**
SSI function code 14 (SSOBDOM)

**SSOBSSIB**
Address of the SSIB control block

**SSOBRETN**
Return code from the previous function routine (when SSI function code 14 is operating in broadcast mode)

**SSOBINDV**
Address of the function-dependent area (SSDM control block)

**SSDM Contents:** MVS sets these fields in the SSDM control block on input:

<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 in the z/OS Internet library (http://www.ibm.com/systems/z/os/zos/bkserv/) 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 461, 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:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 ABDPlcontrol blocks reside in storage below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The function routine should provide an ESTAE-type recovery environment. For more information on how to set up an ESTAE-type recovery environment, see the topic on writing recovery routines in z/OS MVS Programming: Authorized Assembler Services Guide</td>
</tr>
</tbody>
</table>

Figure 40 on page 523 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>
</tbody>
</table>
SSI Function Code 48

**SSOBRETN**
Return code from previous subsystem function routine or zero.

Because broadcast requests are routed to all active subsystems, the SSOBRETN field contains the return code value set by some previously invoked subsystem or zero. See "Output Register Information" on page 491 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 461](#), 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 IEFISVEC service; see [“Building the SSVT” on page 573](#) and [“Enabling Your Subsystem for New Functions” on page 580](#) 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
SSI Function Code 50

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:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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 an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

Figure 41 shows the environment on entry to the function routine for SSI function code 50.

Figure 41. 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>SSOBLEN</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>
</tbody>
</table>

SSOBRETN
Return code from previous subsystem function routine or zero.
Since broadcast requests are routed to all active subsystems, upon entry to the function routine SSOBRETN contains the return code value set by the previously invoked subsystem function code(s) or zero. See “Output Register Information” on page 528 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>
</tbody>
</table>

Chapter 6. SSI Function Codes Your Subsystem Can Support 527
SSI Function Code 50

Subsystem name — name of subsystem which is enabled to receive this function code.

**SSET Contents:** The following fields in the SSET control block are set on input:

<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>
<tr>
<td>SSETFLAG</td>
<td>Flag indicators</td>
</tr>
<tr>
<td>SSETYPE ON</td>
<td>indicates an abnormal ending task</td>
</tr>
<tr>
<td>SSETYPE OFF</td>
<td>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>

**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.
SSI Function Code 54

The information that follows describes what a user-supplied subsystem needs to provide so that it can process incoming SSI function code 54 requests from callers that request information like the information provided by the two IBM-supplied subsystems. The user-supplied subsystem must then provide both the function routine to handle this request, as well as the information concerning the specific returned information. The user-supplied subsystem must provide information to the callers, because all version information returned to the caller is defined by, and has meaning only to, the user-supplied subsystem.

**Type of Request**
Directed SSI call.

**Use Information**
A subsystem may want to allow users to obtain the following information about itself:
- Product function modification identifier (FMID)
- Product version number
- Subsystem common name (such as 'XYZ1')
- Any other information that the subsystem wishes to present to the caller.

**Issued to**
- A user-supplied subsystem

**Related SSI Codes**
None.

**Related Concepts**
You need to understand:
- What the caller of the SSI function code 54 must code and what the caller expects to receive. See “Request Subsystem Version Information Call — SSI Function Code 54” on page 60 for a description of this Request Subsystem Version Information call from a calling program’s point of view.
- What the format of the IEFSSVI functional extension is as defined in “Request Subsystem Version Information Call — SSI Function Code 54” on page 60.

**Environment**
Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in you function routine:
- IEFSSOBH
- IEFJSSIB
- IEFSSVI

The function routine receives control in the following environment:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum authorization</td>
<td>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.</td>
</tr>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>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.</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
</tbody>
</table>
## SSI Function Code 54

### Environment variable
- **Value**

### 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](https://www.ibm.com/docs/en/zos) [Authorized Assembler Services Guide](https://www.ibm.com/docs/en/zos) for more information on how to set up an ESTAE-type recovery environment.

Figure 42 shows the environment on entry to the function routine for SSI function code 54.

**Figure 42. Environment on Entry to the Function Routine for SSI Function Code 54**

### Input Register Information

On entry to the function routine the general purpose registers contain:

<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 of the requestor of the service</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>
SSI Function Code 54

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>SSISSNM</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:

Field Name  
Description

SSOBTREN
The function routine sets this field to SSVIOK (decimal 0).

SSVIRLEN
The function routine sets this field to the number of bytes that is used to return the requested information.

This value includes the fixed section as well as the system variable section.

SSVIRVER
The function routine sets this field to the version of the SSVI macro used (SSVICVER).

SSVIFLEN
The function routine sets this field to the length of the fixed header output section (SSVIFSIZ).

SSVIASID
A 2-byte field that contains the ASID of the subsystem if the subsystem has an address space and supports the use of this field.

SSVIFMID
The function routine sets this field (left-justified, and padded to the right with blank (X'40') characters) to the function routine's FMID, if available.

SSVIVERS
The function routine sets this field (left-justified, and padded to the right with blank (X'40') characters) to the version of the subsystem installed. The function routine defines and uses the version naming conventions and meanings.

SSVICNAM
The function routine sets this field (left-justified, and padded to the right with blank (X'40') characters) to the processing subsystem's common name. For example, a subsystem defined as 'JOHN' might choose to return the SSVICNAM value of 'JOHNNY'. The subsystem defines the common name.

SSVIPLVL
This 1-byte field contains the product level of the subsystem. The content of the field is defined by the subsystem.

This field should only be used if the caller-supplied version in field SSVIVER is greater than or equal to 2.
SSI Function Code 54

SSVISLVL
This 1-byte field contains the service level of the subsystem. The content of
the field is defined by the subsystem.

This field should only be used if the caller-supplied version in field
SSVIVER is greater than or equal to 2.

SSVIUOOF
The function routine sets this field to zero (there is no installation variable
output section).

SSVISDOF
The function routine sets this field to the offset of the start of the system
variable section (same value as SSVIFLEN), if the function routine wants to
supply system variable information.

The DSECT SSVIVDAT mapping begins at this offset, within the SSVI
control block that the caller provided to the function routine. The caller
must provide an SSVI control block large enough to contain the fixed
section and system variable section beginning at this offset (SSVISDOF)
past the start of the fixed section (SSVIHEAD).

The function routine may provide a system variable output section that
contains additional information returned to the caller and mapped using
SSVIVDAT. If it doesn't provide this, the SSVISDOF field must be set to
zero.

The function routine sets the first halfword of this system variable
information section to the length of the system variable section (not
including itself) in the SSVILENGTH field, so that the first byte of the
character string starts past the SSVILENGTH 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 SSVILENGTH field, followed by the character string,
beginning at the SSVIFIELD field. The first byte at the SSVIFIELD 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 528 for more
information on the syntax of the returned system variable sections. IBM
recommends that your function routine also use the same syntax
conventions.

If the function routine returned unsuccessfully to the caller, the system function
may provide any of the following processing depending on the reasons for the
unsuccessful return:

  - Insufficient Storage

    The function routine has determined that the requestor has not supplied a
storage area large enough to contain the requested information. That is, the
caller has not provided a value in the SSVILEN field that is large enough to
contain both the fixed section, as well as any possible system variable section
(length plus actual data). The function routine therefore sets the following fields:

Field Name
  Description
SSOBRRETN
The function routine sets SSOBRRETN to the value of SSVINSTR (decimal 8).

SSVIRLEN
The function routine sets SSVIRLEN to the amount of storage needed to satisfy the request.

The function routine determines the SSVIRLEN value by adding the length of the fixed header section (SSVIFSIZ) to the length of the system variable output section, plus two bytes (for the length value) of the returned string.

Suppose the caller in the example above only provided 30 decimal bytes for the returned information. Our function routine would return decimal 70 in the SSVIRLEN field as follows:
1. 48 — decimal value of the defined symbol SSVIFSIZ
2. 2 — length of the SSVI control block (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 SSOBRRETN field as follows:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBRRETN</td>
<td>The function routine sets SSOBRRETN to the value of SSVIPARM (decimal 16). All other fields in the SSVI control block are not set by the function routine.</td>
</tr>
</tbody>
</table>

• 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>SSOBRRETN</td>
<td>The function routine sets SSOBRRETN to the value of SSVIABLG (decimal 24).</td>
</tr>
</tbody>
</table>
SSI Function Code 54

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 469 for a description of this relationship and an example of the associated processing.

Environment
Review “Function Routines/Function Codes” on page 461, 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 575 and “Enabling Your Subsystem for New Functions” on page 580 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:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</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>
</tbody>
</table>
**SSI Function Code 58**

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<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 SSSM control blocks reside in storage below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>None</td>
</tr>
</tbody>
</table>

Figure 43 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:

- **Register**
  - **Contents**
    - 0 : Address of the SSCVT
    - 1 : Address of the SSOB control block
    - 13 : Address of a standard 18-word save area
    - 14 : Return address
    - 15 : Entry point address

On entry to the function routine the access registers are unused.

**Input Parameters**

Input parameters for the function routine are:

- SSOB
- SSIB
SSI Function Code 58

- SSSM

**SSOB Contents:** SMF 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 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>SSIIBLEN</td>
<td>Length of the SSIB control block</td>
</tr>
<tr>
<td>SSIIBSSNM</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>
<tr>
<td></td>
<td>• SSSMSMFA — SMF is active</td>
</tr>
<tr>
<td></td>
<td>The following flags identify the source of the SUBPARM parameter value for the subsystem:</td>
</tr>
<tr>
<td></td>
<td>• SSSMMEMB — Value from the parmlib member</td>
</tr>
<tr>
<td></td>
<td>• SSSMRPLY — Value from the operator reply</td>
</tr>
<tr>
<td></td>
<td>• SSSMDFLT — Value from the default table</td>
</tr>
<tr>
<td></td>
<td>• SSSMCONF — Value changed due to conflicts</td>
</tr>
<tr>
<td></td>
<td>• SSSMCHNG — Value changed by IPL or SET processing</td>
</tr>
</tbody>
</table>

You can use the following fields to communicate with the console that issued the SET SMF=xx or SETSMF command being processed:

**SSSMCNID**

Command console ID
SSI Function Code 58

SSSTOKN
Command CART

Output Register Information
Upon exit from the function routine, the general purpose registers must contain:

Register
Contents
0-12  Restored to contents on entry
14    Return address
15    Return code

Return Code Information
Upon return to the caller of SSI function code 58 (MVS or SMF), register 15 contains the smallest return code from the SSI and SSOBRETN contains the largest return code associated with the smallest return code from the SSI.

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

  Return Code (Decimal)
  Meaning
  0  The function routine recognized the request but did not process it.
  4  The function routine recognized the request and processed it.

Restrictions
The SMF SUBPARM Option Change call cannot be made to subsystems with the following names:
- SYS
- JES2
- JES3
- STC
- TSO
- ASCH.

Example
See “Passing Accounting Parameters to Your Subsystem” on page 469 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 469 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
SSI Function Code 78

- A volume is mounted on a drive because it is in use by another job.

Within an eligible device array entry, the order of the characteristics reflects their relative importance. For example, whether a specific device is mounted is more important than whether the device is automatically switchable.

The system then builds a list of eligible tape devices and associated eligibility values generated from bits in the SSTAIBMM field in the eligible device array entry.

At this point, the system issues SSI function code 78, passes the SSTA (including the eligible device array), and gives your Tape Device Selection function routine a chance to affect the selection. When the function routine gets control, it can set bits in the STAUTSRM field. If STAUTSRM bits are set, the system generates eligibility values that combine STAUTSRM settings and SSTAIBMM settings.

2. Based on the list of eligible devices and associated eligibility values built in step 1, the system selects the optimal device to allocate for the request.

When the UNIT parameter of the request specifies an esoteric that consists of devices from multiple generics (for example, 3490 and 3590-1), the device preference order is used in conjunction with other IBM and user settings to choose the optimal device. Devices belonging to the generic that is higher in preference order naturally have preference over those belonging to a generic lower in the preference order. See z/OS HCD Planning for an explanation and directions for specifying the device preference order.

You cannot specify the generic device type and it cannot be overridden in either IBM or user settings of SSI78. You can apply user settings to influence choice of an optimal device within a generic of a multi generic esoteric. To influence which generic contains the optimal device using HCD or use the STAINEL setting for all devices in the generic which are not desirable. You can determine device type using EDTINFO (for details, see the STADNUM input parameter on page 547).

Table 15 shows the logical relationship between the system settings and the user settings in the eligible device array entry. The first column shows the 1-bit fields the system sets in SSTAIBMM; the second column shows the 1-bit fields the function routine can set in STAUTSRM. The criteria are listed in order of importance, from top to bottom. For example, the most important criteria are:

- STAINEL, a user field that can remove the device from consideration
- STADMND, 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 15. Relationship between System and User Criteria

<table>
<thead>
<tr>
<th>Importance</th>
<th>System criteria (SSTAIBMM)</th>
<th>User criteria (STAUTSRM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STAINEL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>STADMND</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SSTAUS01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SSTAUS02</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SSTAONUN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SSTAUS03</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SSTAUS04</td>
<td></td>
</tr>
</tbody>
</table>
### Table 15. 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>8</td>
<td>SSTANAFH</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>SSTAUS05</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>SSTAUS06</td>
</tr>
<tr>
<td>11</td>
<td>SSTASPCM</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>SSTAUS07</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>SSTAUS08</td>
</tr>
<tr>
<td>14</td>
<td>Generic device type not specified by a bit</td>
<td></td>
</tr>
<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>SSTAACL1</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>SSTAACL2</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>SSTAACL3</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>SSTAIVOL</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>SSTAANAS</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>SSTAAANAS</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 545; descriptions of SSTAUSRM fields are found in “Output Parameters” on page 549.

**Environment**

Review “Function Routines/Function Codes” on page 461, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.
SSI Function Code 78

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 575 and “Enabling Your Subsystem for New Functions” on page 580 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:

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum authorization</td>
<td>Supervisor state with PSW key 1</td>
</tr>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>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>Serialization for allocation resources is held by Allocation</td>
</tr>
<tr>
<td>Control parameters</td>
<td>The SSIB, SSOB, and SSTA control blocks can reside either above or below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The function routine must provide an ESTAE-type recovery environment. See z/OS MVS Programming: Authorized Assembler Services Guide</td>
</tr>
</tbody>
</table>

The following figures show the environment at the time of the call for SSI function code 78.
Figure 44. 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>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

---

*Figure 45. Continuation of Environment at Time of Call for SSI Function Code 78*
**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>SSOBLLEN</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>SSILOSSNM</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-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>SSTAID</td>
<td>Identifier ‘SSTA’</td>
</tr>
<tr>
<td>SSTAVERS</td>
<td>Current SSTA version number</td>
</tr>
<tr>
<td>SSTAFLGS</td>
<td>Type of call, such as:</td>
</tr>
<tr>
<td></td>
<td>• First call for this job step or dynamic allocation invocation</td>
</tr>
<tr>
<td></td>
<td>• Call from allocation recovery</td>
</tr>
<tr>
<td></td>
<td>• Call from tape allocation retry processing</td>
</tr>
</tbody>
</table>
SSI Function Code 78

SSTASNAM
System name

SSTAJNAM
Job name

SSTASTNM
Job step name or procedure name and job step name. If the job step is not a procedure step, SSTASTNM is an 8-byte job step name and an 8-byte reserved field. If the job step is a procedure step, SSTASTNM is an 8-byte procedure step name and an 8-byte job step name of the step that called the procedure.

SSTASTPN
Step number

SSTANDDS
Number of DDs

SSTADDAP
Pointer to the first DD array for this job step. Set to zero if no DD array entries exist.

SSTAHDRL
Length of the SSTA header

DD Array Entry: There is one DD array entry for each DD statement or dynamic allocation that requests a non-SMS managed, non-DUMMY, non-SUBSYStem tape device. IBM sets the following fields on input:

Field Name
Description

SSTADDN
DD name. Blank if other than the first DD in a concatenation

SSTAJFCP
Pointer to the JFCB for this DD section

SSTACPOS
Concatenation position. Set to 1 for the first DD in a concatenation, 2 for the second DD, etc. Set to 1 for a DD that is not part of a concatenation

SSTADDF1
DD level information, including DISP and GDG specifications

SSTADDF2
DD level information byte 2, including unit affinity indicator

SSTANDRA
Number of devices requested

SSTADRAP
Pointer to the first device request section for this DD

SSTADDAN
Pointer to the next device request section

SSTADDAI
Length of one DD array entry

Device Request Array Entry: There is one device request array entry for each device or unit requested on a non-SMS managed, non-DUMMY, non-SUBSYStem...
DD statement or dynamic allocation request. For example, UNIT=(TAPE,2) would generate two device request array entries. IBM sets the following fields on input:

### Field Name
### Description

<table>
<thead>
<tr>
<th>SSTAVOLI</th>
<th>Volume serial number. Relevant only for a specific request (indicated by bit SSTASPEC in field SSTAREQT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTADNDV</td>
<td>Half word count of the number of eligible devices. The maximum number of eligible devices that this field can contain is 65535.</td>
</tr>
<tr>
<td><strong>Note:</strong> This field is for compatibility only. Use the SSTANDVS field for a full word count.</td>
<td></td>
</tr>
<tr>
<td>SSTAREQT</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>SSTADEVP</td>
<td>Pointer to the eligible device array for this DD</td>
</tr>
<tr>
<td>SSTADRAN</td>
<td>Pointer to the next device request array entry</td>
</tr>
<tr>
<td>SSTANDVS</td>
<td>Number of eligible devices.</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 549.

### Eligible Device Array Entry: There is one eligible device array entry for each device eligible for a particular DD array entry. IBM sets the following fields on input:

### Field Name
### Description

<p>| SSTADNUM   | Device number, in EBCDIC. Example: The representation of device number 5B0 would be F0F5C2F0. You can use this number as input to EDTINFO to obtain further information about the device, such as its generic device type and any esoteric service groups of which this device is a part. (See z/OS MVS Programming: Assembler Services Reference ABE-HSP GC28-1910, for additional information about EDTINFO.) |
| SSTAIIBM    | Following are the eligibility bits that the system sets. (Unless otherwise specified, the IBM eligibility bits apply to both dedicated and automatically switchable devices.) |</p>
<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>SSTAAACL1</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>SSTAAACL2</td>
<td>The installed ACL is inactive</td>
</tr>
<tr>
<td>SSTAAACL3</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 needed for this specific request. However, the last volume dismounted from this device also matches.</td>
</tr>
</tbody>
</table>
| SSTAAVOL   | A volume is mounted on this device and one of the following conditions is true:
SSI Function Code 78

- 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 S斯塔PRE和S斯塔USRM fields in the eligible device array entry. See “Output Parameters.”

**Output Register Information**

Upon exit from the function routine, the general purpose registers must contain:

**Register**

**Contents**

0  Used as a work register by the system

1  Address of the SSOB

2-13  Restored to contents on entry

14  Return address

15  Return code

**Return Code Information**

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB to one of the following:

  **Return Code (Decimal)**
  **Meaning**

  0  The function routine recognized the request but did not process it.

  4  The function routine recognized the request and processed it.

**Output Parameters**

Output parameters for the function routine are:

**Field Name**

**Description**

**S斯塔UDFR**

The field in device request section that forces a request to have mounting deferred until the dataset is actually opened

**S斯塔UPRF**

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.
SSI Function Code 78

- 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.

SSTAPREF

The field in the eligible device array entry that contains the preference value for the system to use. This field allows the function routine to influence the allocation of devices when all other attributes are the same. Use this field only if you set SSTAUPRF.

SSTAUSRM

The field in the eligible device array entry that allows the function routine to add its own criteria to the eligibility mask that the system associates with each eligible device:

<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 15 on page 540 shows how each of these bits relates to the system mask SSTAIBM.</td>
</tr>
</tbody>
</table>

SSTAEDAL

Length of one device request array entry

Restrictions

SSI function code 78 is not available to change the selection of SMS-managed or JES3-managed tape devices

Note that while MVS allocation processes your function routine, it is not processing other allocation requests. This might degrade performance.

Example

An installation writes a Tape Device Selection function routine to ensure that tape devices 270 and 271 are available only for HSM tape requests. (This example is included in SYS1.SAMPLIB as member IEFTASSI.)

TAPESSI CSECT
TAPESSI AMODE 31
TAPESSI RMODE ANY

******************* START OF SPECIFICATIONS ****************************
* *
* +01* NAME= *
  *
* +01* TYPE= Sample Subsystem *
  *
* +01* FIRST ELIGIBLE PRODUCT= HBB5520 *
  *
* +01* FIRST INELIGIBLE PRODUCT= HBB5510 *
  *
* +01* OPERATION= *
  *
This is a sample tape allocation subsystem. It will
  *
  reserve devices 270 and 271 for only HSM jobs.
  *
SSI Function Code 78

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

SOFTWARE DEPENDENCIES:

REQUIRED PRODUCTS= HBB5520

OUTPUT:

MSGIDS= NONE

ABENDCODES= NONE

END OF SPECIFICATIONS

Base register: 12

Other register use:

10 SSCVT
2 SSOB
9 SSIB
8 SSTA

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.
Supervisor state, AMODE(31), RMODE(ANY)

Chain save areas

USING TAPESSI,12
SAVE (14,12) Save caller registers
LR 12,15 Establish module base register
LR 10,0 Establish addressability
USING SSCT,10 to the SSCVT
LR 2,1 Establish addressability
USING SSOB,2 to the SSOB
GETMAIN R,LV=84,SP=230 Get working storage
ST 13,4(1) Chain saveareas forward
ST 1,8(13) Chain saveareas backward
LR 13,1 Point to this module's savearea
LR 11,1 Point to dynamic storage
USING DYNAM,11 Base dynamic storage

Validate the request

L 9,SSOBSSIB Establish addressability
USING SSIB,9 to the SSIB
CLC SSIBSSNM,SSCTSNAM Verify the subsystem name
BNE ERROR This should never happen
L 8,SSOBINDV Pointer to function dependent area

Check for job name beginning with HSM
ssi function code 78

using stta,8 set basing
clc hsmname,stajnam 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,staddap get address of dd entries
using stadda,7 base dd entries
l 4,staddds get number of dd's
ltr 4,4 check for zero
bz nocheck if zero then no dd's to check
st 4,numdds else save in local storage

ddloops equ *

start looping through dd's
l 6,stadrap get address of first request
using stadra,6 base request entries
l 4,stadra get number of requests
ltr 4,4 check for zero
bz ddloope if zero then no requests
st 4,numreqs else save in local storage

reqloops equ *

start looping through requests
l 5,stadevp get address of first device
using staeda,5 base device entries
l 4,staendvs get number of devices eligible
ltr 4,4 check for zero
bz reqloope if zero then no devices
st 4, numdevs else save in local storage

**************************************************************************
* check each eligible device entry to make sure that devices *
* 0270 and 0271 are not eligible to this request. *
**************************************************************************

devloops equ *

start looping through devices
clc stadnum,hsmdev1 is device reserved for hsm?
be makeinel yes, go make ineligible
clc stadnum,hsmdev2 is device reserved for hsm?
bne devloope no, bypass making ineligible

makeinel equ *
oi stausel, b'10000000' mark device ineligible

devloope equ *
end of eligible device loop
la 3,12 get size of staeda 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 devloopes loop back if more to process

reqloope equ *
end of request loop
l 6,stadran get address of next request
l 4,numreqs get local counter
la 3,1 get amount to decrement count
slr 4,3 decrement count
st 4,numreqs save request count
ltr 4,4 check request count
bnz reqloops loop back if more to process

ddloope equ *
end of dd loop
l 7,staddan get address of next dd entry
l 4,numddds 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 ddloopes loop back if more to process

nocheck equ *
mvc ssobretn,=f'0' indicate function success
b return
SSI Function Code 78

```
ERROR EQU *
MVC SSORETN,=F'20' Indicate function failure
***********************************************************************
***********************************************************************
* Return to the SSI *
***********************************************************************
RETURN EQU *

L 8,4(13) Pointer to caller's savearea
FREEMAIN R,LV=84,A=(13),SP=230
LR 13,8
LM 14,12,12(13) Restore caller's registers
LA 15,0 RC=0
BSM 0,14 Return to the SSI

* HSMNAME DC CL3'HSM' HSM Jobname
HSMDEV1 DC CL4'H'270' Device reserved for HSM
HSMDEV2 DC CL4'H'271' Device reserved for HSM

* DYNAM DSECT Dynamic storage
SAVEAREA DS 18F Module save area
NUMDDS DS F Number of DDs
NUMREQS DS F Number of requests
NUMDEVS DS F Number of eligible devices

* IEFJSCVT
IEFSSOBH
IEFJSSIB
IEFSSTA
END
```
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:
Troubleshooting

IEFJ007I  A SYSTEM ERROR HAS OCCURRED DURING INITIALIZATION OF SUBSYSTEM subname

If you get this message, examine the DUMP data set to identify the problem.

If an incorrect keyword is found in IEFSSNxx, the following message is written:

IEFJ001I  memname LINE line-number:  ERROR IN SUBSYSTEM DEFINITION, REFER TO HARDCOPY LOG

If you get this message, the system continues processing the rest of IEFSSNxx, and you should correct the keyword indicated. The system does not process the subsystem definition containing the incorrect keyword.

Handling function request errors

When you are troubleshooting errors during SSI function request processing, do the following:

- Capture the system dump
- Identify the type of error
- Determine the cause of the error.

Capturing the System Dump

If an abend occurs while processing a subsystem function request, the SSI requests a dump (unless a subsystem function routine takes one first). The dump title is similar to the following:

```
TITLE=COMPON=SSI,COMPID=5752SC1B6,ISSUER=IEFJSaaa,
MODULE=IEFJbbbb,ABEND=xxx,REASON=yyyyyyyy DUMP
```

The issuer is one of the following:

- IEFJSARR, if the caller of the SSI is in task mode and holds no locks
- IEFJSFRR, if the caller of the SSI is in SRB mode or holds a lock
- IEFJSPCE, if the error is a recursive failure in the SSIs recovery.

For function request errors, the module is one of the following:

- IEFJRASP, for broadcast function requests
- IEFJSRE1, for directed function requests or for broadcast function requests that have not yet been passed to IEFJRASP.

Other module names may appear for errors in SSI services other than routing function requests.

Another variation of the dump title is the following:

```
DUMP TITLE=COMPON=SSI,COMPID=5752SC1B6,ISSUER=IEFJSaaa,
MODULE=IEFJbbbb,ABEND=xxx,REASON=yyyyyyyy,SNAME=zzzz
```

This variation will appear when SSI has determined that the error occurred in a subsystem function routine. The dump title identifies the name of the failing subsystem. SNAME refers to the subsystem, while zzzz is the name of the subsystem.

After creating a subsystem vector table, the SSI retains only the addresses of the function routines represented in the table, and therefore cannot identify the failing routine by name.
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
Troubleshooting

The error occurred when the SSI tried to reference the SSCVT describing the target subsystem. The target subsystem is either not dynamic, or is dynamic but is not using an SSI-managed SSVT control block to route function requests.

- **Error locating the subsystem**
  The error occurred when the SSI tried to locate system control blocks associated with the target subsystem.

- **Error validating the request**
  The error occurred when the SSI tried to validate the SSOB/SSIB control block chain describing the function request.

Contact the IBM Support Center for any other footprints that you may receive.

**Identifying Problem Type when the VRA is not Available**
You can identify the type of error when the VRA is not available by checking the PSW and the registers at the time of the error as follows:

- If the PSW equals register 15, it probably indicates that the subsystem function routine address in the SSVT is not valid.
- If the PSW contains a valid address in a module other than IEFJSRE1 or IEFJRASP, it is probably a subsystem function routine error. The error occurred in this routine.
- If the PSW contains a valid address in IEFJSRE1 or IEFJRASP, the error occurred while referencing subsystem related control blocks, the input parameter list, or in the SSI. Examine the SSCVT chain pointed to by the JESSSCT field for pointers that are not valid. The SSIDATA IPCS subcommand displays the subsystems defined to the SSI based on this chain, and may help identify a problem. See [z/OS MVS IPCS Commands](z/OS MVS IPCS Commands) or [z/OS MVS Diagnosis: Reference](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 SSI’s caller, and may contain information other than what was provided in the original control blocks. The VRA contains the address of the SSOB control block, and the SSOBSSIB field of the SSOB control block locates the SSIB control block. The SSOBINDV field, if non-zero, points to the SSOB extension originally provided by the caller.

You can also use the current SSCVT to identify the current subsystem. If the address of the SSCVT appears in the VRA, the SSCTSNAM field identifies the subsystem.
If the footprints indicate that the error occurred while locating the target subsystem, and the SSI was processing a broadcast request, the VRA identifies the last successfully processed subsystem. The VRA section with the header 'LAST PROCESSED SSCVT', lists the address of the last subsystem to which the current request was successfully routed. Subsystems receive broadcast requests in the order in which they appear in the SSCVT chain (anchored by the JESSCT 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 487, 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 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 566**
  This example documents General-Use Programming Interfaces and Associated Guidance Information.

- **“Example 3 — Subsystem Function Routine (DELETEIT)” on page 568**
  This example documents General-Use Programming Interfaces and Associated Guidance Information.

- **“Example 4 — Subsystem Function Routine (LISTEN)” on page 569**
  This example documents Product-Sensitive Programming Interfaces and Associated Guidance Information.

- **“Example 5 — Subsystem Requesting Routine (TSYSCALL)” on page 571**
  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 *
```
Appendix A — Examples

* Register 1 points to a two-word parameter list *
  * - Word 1 = address of the SSCVT for the TSYS subsystem *
  * - Word 2 = address of the JSIPL *
* REGISTER USE *
  * 1 - TSYS
  * 10 - SSCVT
  * 11 - JSIPL
  * 12 - Code register
  * 13 - Data register
* MACROS *
  * CVT
  * ESTAE
  * FREEMAIN
  * GETMAIN
  * IEFJESCT
  * IEFJSCVT
  * IEFSSI
  * IEFSSVT
  * IEFSSVTI
  * RETURN
  * SETRP
  * WTO
**************************************************************************
**
* Chain saveareas. *
*************************************************************************
USING TSYSINIT,12
SAVE (14,12) Save caller's registers
LR 12,15 Establish module base register
LR 10,1 Save pointer to parameter list
GETMAIN R,LV=WORKALEN Get working storage
ST 13,4(1) Chain saveareas backward
ST 1,8(13) Chain saveareas forward
LR 13,1 Point to this module's savearea

* USING WORKAREA,13 Addressability to work area
L 11,4(10) Establish addressability
USING JSIPL,11 to the JSIPL
L 10,0(10) Establish addressability
USING SSCT,10 to the SSCVT
*
*************************************************************************
**
* Establish ESTAE *
*************************************************************************
XC ESTAE,ESTAE Clear ESTAE parameter list
L 8,=A(TSYSERR) Address of ESTAE routine
ESTAE (8),CT,PARAM=ARETRY,MF=(E,ESTAE)
LTR 15,15 If ESTAE failed
BNZ ESTAERR report it and return
*
*************************************************************************
**
* Invoke the IEFSSVT REQUEST(CREATE) macro to build and initialize *
* the vector table, using the static function routine input table. *
* The function routines reside in LINKLIB and must be loaded to *
* global storage to make them available to all address spaces. *
* Register notation is used to identify the output token for *
* demonstration purposes. *
*************************************************************************
LA 2,TOKEN1
* IEFSSVT REQUEST=CREATE,SUBNAME=SSCTSNAME,SSVIDATA=ROUTINE1, *
  OUTTOKEN=(2),LOADTOGLOBAL=YES,MAXENTRIES=ENTRIES,
Appendix A — Examples

RETCODE=RC,RSNCODE=REASON, +
MF=(E,VTPARMS)

* B TESTVTCR(15) Check return code

TESTVTCR EQU *
B ANCHORCB 0 - Processing successful
B VTERR 4 - Warning
B VTERR 8 - Invalid parameters
B VTERR 12 - Request failure
B VTERR 16 - Error loading subsystem
B VTERR 20 - System error
B VTERR 24 - SSI service not available

* ANCHORCB EQU * Entry for vector table created
*************************************************************************
* Initialize and anchor the subsystem-specific control block used *
* by TSYS and its function routines.                          *
*************************************************************************
GETMAIN R,LV=CBLEN,SP=241 Get storage for TSYS control +
block
USING TSYSCB,1
XC TSYSCB,TSYSCB Clear control block
MVC TSYSSID(4),CBACRO Move in eye-catcher
LA 7,1 Version 1
STH 7,TSYSVER Put version number in control +
block
LA 7,CBLEN Get control block length
STH 7,TSYSLEN Put length in control block
ST 1,CBADDR Save control block address
DROP 1

* IEFSSI REQUEST=PUT,SUBNAME=SSCTSNAM,SUBDATA1=CBADDR, +
RETCODE=RC,RSNCODE=REASON, +
MF=(E,SSIPARMS)

* B TESTPUT(15) Check return code

TESTPUT EQU *
B OPTIONS 0 - Processing successful
B SSIERR 4 - Warning
B SSIERR 8 - Invalid parameters
B SSIERR 12 - Request failure
B SSIERR 16 - Not defined
B SSIERR 20 - System error
B SSIERR 24 - SSI service not available

*************************************************************************
* Inform the SSI that TSYS will respond to the SETSSI command. *
*************************************************************************
OPTIONS EQU * Entry for successful PUT
* IEFSSI REQUEST=OPTIONS,SUBNAME=SSCTSNAM,COMMAND=YES, +
RETCODE=RC,RSNCODE=REASON, +
MF=(E,SSIPARMS)

* B TESTOPT(15) Check return code

TESTOPT EQU *
B ACTIVATE 0 - Processing successful
B SSIERR 4 - Warning
B SSIERR 8 - Invalid parameters
B SSIERR 12 - Request failure
B SSIERR 16 - Not defined
B SSIERR 20 - System error
Appendix A — Examples

B SSIERR 24 - SSI service not available
* ACTIVATE EQU * Entry for successful OPTIONS
************************************************************************
* Activate the subsystem. *
************************************************************************
IEFSSI REQUEST=ACTIVATE,SUBNAME=SSCTSNAM,INTOKEN=TOKEN1, +
RETCODE=RC,RSNCODE=REASON, +
MF=(E,SSIPARMS)
************************************************************************
B TESTACT(15)
************************************************************************
ACTIVEOK 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 ERRMSG Issue error message
************************************************************************
SSIERR EQU *
MVC FAILSRV(L'SSISRV),SSISRV Get name of failing service
************************************************************************
ERRMSG EQU *
MVC SERVERRD(SERVMSGL),SERVERRS Copy static message
* L 7,RC Get return code
CVD 7,DOUBLE Convert to decimal
UNPK RCODE1,DOUBLE Make return code printable
MVZ RCODE1+3,RCODE1
MVC SERVERRD+43(2),RCODE1+2 Put return code in message
* L 7,REASON Get reason code
CVD 7,DOUBLE Convert to decimal
UNPK RCODE1,DOUBLE Make reason code printable
MVZ RCODE1+3,RCODE1
MVC SERVERRD+55(4),RCODE1 Put reason code in message
* MVC SERVERRD+18(L'FAILSRV),FAILSRV Put name of failing service in message
WTO MF=(E,SERVERRD),CONSNAME=JSICNAME Issue message
B DONE
************************************************************************
INITERR EQU *
MVC INITERRD(INITMSGL),INITERRS Copy static message
WTO MF=(E,INITERRD),CONSNAME=JSICNAME Issue message
B DONE
************************************************************************
ESTAERR EQU *
MVC ESTAERRD(ESTAMSGL),ESTAERRS Copy static message
WTO MF=(E,ESTAERRD),CONSNAME=JSICNAME Issue message
B RETURN
************************************************************************
**Appendix A — Examples**

* Cancel the ESTAE and return to caller. *

```assembly
DONE EQU *
ESTAE 0
RETURN EQU *
  L 8,4(13) Pointer to caller's savearea
FREEMAIN R, LV=WORKALEN, A=(13)
LR 13, 8
RETURN (14,12), RC=0
*
************************************************************************

* ESTAE routine. *
************************************************************************

TSYSERR EQU *
  DROP 12 Drop current addressability
  USING TSYSERR, 15 Set addressability to TSYSERR
  LR 12, 15 Copy address of TSYSERR
  S I2, =A(TSYSERR-TSYSINIT) Reestablish code register
  DROP 15 Drop addressability to TSYSERR
  USING TSYSINIT, 12 Reset addressability
  CL 0, =F'12' If no SDWA provided
  BE TSYSERRA Branch to percolate
  USING SDWA, 1
  L 4, SDWAPARM
  L 4, 0(4)
  DROP 1
  SETRP WKAREA=(1), RC=4, RETADDR=(4), FRESWA=YES, RETREGS=YES

TSYSERRA EQU *
  XR 15, 15 Indicate percolation
  BR 14
*
************************************************************************

* Define static function routine input table. *
************************************************************************

IEFSSVTI TYPE=INITIAL, SSVTDATA=ROUTINE1, TABLEN=STABLEN
IEFSSVTI TYPE=ENTRY, NUMFCODES=1, FCODES=254, FUNCNAME=WRITEIT
IEFSSVTI TYPE=ENTRY, NUMFCODES=1, FCODES=255, FUNCNAME=DELETEIT
IEFSSVTI TYPE=ENTRY, NUMFCODES=1, FCODES=9, FUNCNAME=LISTEN
IEFSSVTI TYPE=FINAL
*
************************************************************************

* Function routine data. *
************************************************************************

WRITEIT DC CL8'WRITEIT '
LISTEN DC CL8'LISTEN '
DELETEIT DC CL8'DELETEIT'
ENTRIES DC H'4'
SSVTSRV DC CL7'IEFSSVT'
SSISRV DC CL7'IEFSSI '
CBACRO DC CL4'TSCB '
*
ARETRY DC A(INITERR)
*
SERVERRS WTO 'TSYS ERROR IN xxxxxxx SERVICE, RETCODE xx, RSNCODE xxxx', +
  CONSNAME=, MF=L
SERVMSGL EQU *-SERVERRS
*
INITERRS WTO 'TSYS - SUBSYSTEM INITIALIZATION FAILED', +
  CONSNAME=, MF=L
INITMSGL EQU *-INITERRS
*
ESTAERRS WTO 'TSYS - SUBSYSTEM ESTAE FAILED', +
  CONSNAME=, MF=L
ESTAMSG EQU *-ESTAERRS
*
LTORG
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

---

Example 2 — Subsystem Function Routine (WRITEIT)
Appendix A — Examples

* Name of the module: WRITEIT
* System macros used:
  * FREEMAIN
  * GETMAIN
  * IEFJSCVT
  * IEFJSSIB
  * IEFJSSOB
  * 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

* 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,SSOBSIB 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',ROUTE=(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
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
*
************************************************************************
**
Appendix A — Examples

Example 4 — Subsystem Function Routine (LISTEN)

LISTEN CSECT
LISTEN AMODE ANY
LISTEN RMODE ANY

**************************************************************************
** Function:**
** This function routine of the TSYS subsystem is invoked by the SSI broadcast of WTO. When it detects the WTO message issued by the WRITEIT routine, it alters the attributes of the WTO to be a non-rollable message.
**
**************************************************************************
** Name of the module: LISTEN**
** System macros used:**
* FREEMAIN *
* GETMAIN *
* IEFJSSOB *
* IEFJSSIB *
* IEFJSCVT *
* IEFSSOBH *
* IEFJSSIB *
* END

**************************************************************************
** Validate the request and delete the critical eventual action message **
**************************************************************************

L 9,SSOBSSIB Establish addressability
USING SSIB,9 to the SSIB
CLC SSIBSSNM,SSCTSNAM Verify the subsystem name
BNE ERROR This should never happen
L 8,SSOBINDV Pointer to function dependent area

L 1,2(8) Get message identification returned by WTO

DOM MSG=(1)
MVC SSORETN,=F'0' Indicate function success
B RETURN

ERROR EQU *
MVC SSORETN,=F'4' Indicate function failure

**************************************************************************
** Return to the SSI **
**************************************************************************

RETURN EQU *

**************************************************************************
** Establish addressability to the SSIB **
**************************************************************************

RETURN EQU *

**************************************************************************
** Validate the request and delete the critical eventual action message **
**************************************************************************

Appendix A — Examples
Chapter 8. Examples — Subsystem Interface Routines 569
Appendix A — Examples

* Base register: 12
* Other register use:
  10 SSOB
  11 SSOBEXT
  9 WQE
* Attributes:
  This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.
************************************************************************
* Chain saveareas
************************************************************************
USING LISTEN,12
SAVE (14,12) Save caller registers
LR 12,15 Establish module base register
LR 10,1 Establish addressability
USING SSOB,10 to the SSOB
GETMAIN R,LV=72 Get working storage
ST 13,4(1) Chain saveareas forward
ST 1,8(13) Chain saveareas backward
LR 13,1 Point to this module's savearea
************************************************************************
* Alter message number TSYS001 to be a critical eventual action message (descriptor code of 11)
************************************************************************
L 11,SSOBINDV Chain through
USING SSOBEXT,11 the SSWT to
L 9,SSWTWQE establish addressability
USING WQE,9 to the WQE
CLC WQETXT+2(8),=C'TSYS001 ' Check for desired message
BNE MSGDONE
TM WQEDC2,WQEDCK Check for DESC(11) already set
BO MSGDONE
OI WQEDC2,WQEDCK Alter message to be DESC(11)
OI WQEMLI1,WQEMLCE and eventual critical
OI WQEMCSF1,WQEMCSA Indicate descriptor codes present
MVC SSOBRETN,=F'4' Indicate function recognized request, and processed it
B RETURN
MSGDONE EQU *
MVC SSOBRETN,=F'0' Indicate function recognized request, but did not care
*
************************************************************************
* Return to the SSI
************************************************************************
RETURN EQU *
L 8,4(13) Pointer to caller's savearea
FREEMAIN R,LV=72,A=(13)
LR 13,8
LM 14,12,12(13) Restore the caller's registers
LA 15,0 RC=0
BSM 0,14 Return
*
IEFJSSOB (WT),CONTIG=NO
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 forward **
** ST 1,SAVEAREA+4 Chain saveareas backward **
** LA 10,SSOBD Establish addressability **
** USING SSOBD,10 to the SSOB **
** LA 11,SSIBD Establish addressability **
** USING SSIBD,11 to the SSIB **
**
**************************************************************************
Appendix A — Examples

* Format the SSOB
* ********************************************************************************
MVC SSOBID,=C'SSOB' Set control block identifier
LA 8,SSOBHSIZ
STH 8,SSOBLEN Set control block size
ST 11,SSOBSSIB Set pointer to SSIB
MVC SSOBINDV,=A(MSGIDEXT) Set pointer to function dependent area
*
* ********************************************************************************

* Format the SSIB
* ********************************************************************************
MVC SSIBID,=C'SSIB' Set control block identifier
LA 8,SSIBSIZE
STH 8,SSIBLEN Set control block size
MVC SSIBSSNM,=C'TSYS' Set subsystem name
*
* ********************************************************************************

* Call the TSYS subsystem
* ********************************************************************************
MVC SSOBFUNC,WRITEIT Request the TSYS001 WTO message
OI PARMLST,X'80' Mark end of parameter list
LA 1,PARMLST Point to the parameter list
IEFSSREQ
LTR 15,15 Check return code from SSI
BNZ ERROR
CLC SSOBRETN,=F'0' Check return code from subsystem
BNZ ERROR
STIMER WAIT,BINTVL=TENSEC
*
MVC SSOBFUNC,DELETEIT Request DOM of the TSYS001 WTO message
LA 1,PARMLST Point to the parameter list
IEFSSREQ
LTR 15,15 Check return code from SSI
BNZ ERROR
CLC SSOBRETN,=F'0' Check return code from subsystem
BNZ ERROR
B RETURN
*
ERROR EQU *
ABEND 1001,,,USER Indicate function failure
*
* ********************************************************************************

* Restore registers and return
* ********************************************************************************
RETURN EQU *
L 13,SAVEAREA+4 Pointer to caller's savearea
RETURN (14,12),RC=0
*
* TENSEC DC F'1000' Ten seconds in 1/100ths
WRITEIT DC H'254'
DELETEIT DC H'255'
*
SAVEAREA DC 18F'0'
*
PARMLST DC A(SSOBD) IEFSSREQ parameter list
*
SSOBD DS 0F SSOB data
DC (SSOBSHSIZ)X'00'
*
MSGISIEXT DS 0F Function dependent area
MSGISLEN DC AL2(MSGIDSIZ)
MSGIDENT DC F'0' Message identifier from TFUNC1
MSGIDSIZ EQU *=MSGISIEXT
Appendix A — Examples

*  SSIBD   DS  OF  SSIB data
*  DC   (SSIBSIZE)X'00'
*  *  IEFSSOBH
*  *  IEFJSSIB
*  *  CVT DSECT=YES
*  *  IEFJESCT
*  *  END
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 471 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 466.
- 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 463 for more information.

Environment

The following mapping macros are supplied by IBM and may be included in your program when invoking IEFJSVEC:

- IEFVTSPL
Appendix B — Using IEFJSVEC

- IEFJSBVT

The requirements for the caller of IEFJSVEC are:

### Requirements for the caller of IEFJSVEC

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</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 VTSP 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="#">z/OS MVS Programming: Authorized Assembler Services Guide</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:

**Register**

- **Contents**
  - **Register 1**: Address of fullword that contains the address of the subsystem VTSP
  - **Register 13**: Address of a standard 18-word save area
  - **Register 14**: Return address

### Input Parameters

Input parameters for the IEFJSVEC service are:

- VTSPL
- JSBVT — both fixed and variable sections

**VTSP Contents**: Your program sets 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>VTSCVER</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>
</tbody>
</table>
Appendix B — Using IEFJSVEC

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 580 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 VTSPL control block to binary zeros.
Appendix B — Using IEFJSVEC

**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>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>
<tr>
<td>JSBSPL</td>
<td>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.</td>
</tr>
<tr>
<td>JSBMAXFR</td>
<td>Maximum number of function routines you expect the subsystem to need</td>
</tr>
</tbody>
</table>

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:
Appendix B — Using IEFJSVEC

Register
Contents
0-14 Same as on entry to call
15 Return code

Return Code Information
IEFJSVEC returns one of the following return codes in register 15:

Return Code (Decimal)
Meaning
VTSSUCES (0)
Successful completion. The request to build an SSVT was successfully processed.
VTSINVID (4)
An incorrect identifier was specified in VTSPL or JSBVT. Check the input parameter areas to make sure that you specified the proper identifiers, and that the pointers to the input parameter areas are properly defined.
VTSINVIN (8)
An incorrect subsystem name was specified. Check to make sure that you specified a valid subsystem name in the VTSNAME field. Consult with your system programmer to make sure that it matches the name of a valid subsystem defined in the IEFSSNxx parmlib member that is currently in use.
VTSGETFL (12)
Unable to obtain storage for the SSVT. Consult with your system programmer to verify that sufficient storage is available for the subpool specified in the JSBSPL field.
VTSLOGER (16)
Logic error. Contact your IBM service support center.
VTSLOADF (20)
An abend occurred when trying to load the function routine. The VTSFUNCT field contains the name of the function routine being loaded when the problem occurred.
VTSINVBI (24)
An incorrect bit was set in the request flags. Verify that you have set only the VTSCREAT indicator and that you have not set any other bits in the VTSREQ flag byte.
VTSINCR (28)
Unable to process the SSVT build request. The SSVT already exists. Verify that you have specified the correct subsystem name for which an SSVT is to be built. Also ensure that your subsystem initialization code is not accidentally attempting to build an SSVT twice for the same subsystem (specified in VTSNAME).

Output Parameters
Output parameters for the IEFJSVEC service are:
- VTSPL

VTSPL Contents: The VTSPL control block contains the following information upon return from your build SSVT request:
Appendix B — Using IEFJSVEC

Field Name  
Description

VTSSVTAD  
Address of the SSVT, if the SSVT build request was successful (Register 15=0)

VTSSSCVT  
Address of the SSCVT, if the SSVT build request was successful (Register 15=0)

VTSFUNCT  
Name of the function routine processed, if an error occurred when trying to load a function routine (Register 15=20)

Changing What Your Subsystem Can Do

To change what your subsystem can do, you can use IEFJSVEC to:

• Enable your subsystem for new functions
• Disable a previously supported function

Enabling Your Subsystem for New Functions

You can use the enable function of the IEFJSVEC service to:

• Dynamically add one or more function codes to an existing function routine. This function routine might have been specified on the original build SSVT request or might have been added by a previous enable request.

When preparing to enable additional function codes, consider:

– When you will invoke IEFJSVEC.
  If you are invoking IEFJSVEC while running under a command processor, for example, from a subsystem routine invoked during START command processing, provide a console ID and CART, as described in "Input Parameters" on page 581.

– 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 581.

– 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 575.

– What is the name of each additional function routine and the function codes it is to support.
Appendix B — Using IEFJSEVEC

- Where your subsystem function routines are to reside. See Chapter 4, “Setting Up Your Subsystem,” on page 461 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 IEFJSEVEC:

- IEFVTSP
- IEFJSBV

The requirements for the caller of IEFJSEVEC are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</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>Control Parameters</td>
<td>The VTSPL and JSBV control blocks must reside in storage below 16 megabytes.</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>Recovery</td>
<td>The caller of IEFJSEVEC should provide an ESTAE-type of recovery environment. See <a href="https://www.ibm.com/docs/en/zos">z/OS MVS Programming</a> and <a href="https://www.ibm.com/docs/en/zos">Authorized Assembler Services Guide</a> for more information on an ESTAE-type of recovery environment.</td>
</tr>
</tbody>
</table>

Restrictions
The number of function routines supported by a subsystem must not exceed 255.

Input Register Information
Before you invoke IEFJSEVEC, 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 IEFJSEVEC service are:

- VTSPL
- JSBV — 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>VTSLEN</td>
<td>Length of the VTSPL (VTSSIZE) control block</td>
</tr>
</tbody>
</table>
Appendix B — Using IEFJSVEC

VTSVER
Version number of the VTSPL (VTSCVER) control block

VTSCONID
The 1-byte console ID of the console that the subsystem initialization routine issues messages to. If your program sets this field to zero, the VTSCNSID field is used.

This field exists for versions of MVS previous to SP410. IBM recommends that you specify a 4-byte console ID as defined by the VTSCNSID field.

VTSFLAGS
Flags
- VTSGLOAD — load-to-global indicator.
  This indicator applies only when you are adding new function routines to your subsystem and does not apply when you are adding new function codes to an existing function routine. If the VTSGLOAD indicator is set, the SSI loads all of the function routines into pageable CSA. Each loaded routine is associated with the task under which the call to IEFJSVEC was made. The VTSGLOAD indicator applies to all function routines specified on a single invocation of IEFJSVEC.
  Only use the VTSGLOAD indicator when invoking the enable function under a system address space that does not end. If the subsystem invokes the enable function from its own address space or task, those routines are deleted from CSA when the task ends, causing invalid function routine addresses in the SSVT. IBM recommends that you use the VTSGLOAD indicator only when invoking IEFJSVEC from an initialization routine named in IEFSSNxx. Subsystems initialized through START commands should ensure that the function routines are in commonly addressable storage, that is, in the link pack area (LPA, MLPA, FLPA).
  If you want to have some function routines that are loaded into CSA and others that are not, issue separate invocations of IEFJSVEC, one with the VTSGLOAD indicator set and the other with the VTSGLOAD indicator not set. You may use the SSVT build function for one of the requests, if an SSVT does not already exist. However, for any subsequent calls you will need to use the enable function.
  See “Placement of Function Routines” on page 463 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.
Appendix B — Using IEFJSVEC

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.

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 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 VTSP extends must be set to binary zeros.

JSBVT Contents — Fixed Header Section: Your program must set the following fields in the JSBVT control block on input:

Field Name | Description
--- | ---
JSBID | Identifier 'JSBV'
JSBLEN | Length of the JSBVT (fixed header section) control block
JSBVERS | Version number of the JSBVT (JSBCVERS) control block
JSBFUN | Number of function routines specified in this table of data

All other fields in the fixed header section of the JSBVT control block must be set to binary zeros.

JSBVT Contents — Variable Length Section: The JSBVT fixed header is followed by a variable length function routine data area (one for each function routine). Your program must set the following fields on input:

Field Name | Description
--- | ---
JSBLGTH | Length of this function routine’s data area and its function code area (see the JSBFCOD field)
JSBNME | Name of a function routine. The function routine name specified should be either the name of a new function routine to be supported by the subsystem or the name of an existing function routine to which additional
function codes are to be added. The function routine name can be up to eight characters. It must be left-justified and padded to the right with blank (X’40’) characters.

**JSBNUM**
Number of function codes specified for this function routine.

If this enable request is being used to add a new function routine to a subsystem or is being used to add new function codes to an existing function routine, the JSBNUM field should be set to the number of new function codes to be supported by the function routine as specified in the JSBFCOD field on this invocation of IEFJSVEC.

**JSBVT Contents — Variable Length Section:** The function routine data area is 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:

**Return Code**
(Decimal)
*Meaning*

**VTSSUCES (0)**
Successful completion. The request to enable was successfully processed and the SSVT has been updated.

**VTSINVID (4)**
An incorrect identifier was specified in VTSPL or JSBVT. Check the input parameter areas to make sure that you specified the proper identifiers, and that the pointers to the input parameter areas are properly defined.

**VTSINVIN (8)**
An incorrect subsystem name was specified. Check to make sure that you specified a valid subsystem name in the VTSNAME field. Consult with your system programmer to make sure that it matches the name of a valid subsystem defined in the IEFSSNxx parmlib member that is currently in use.

**VTSLOGER (16)**
Logic error. Contact your IBM service support center.
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.

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.

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.

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.

Target vector table is SSI-managed and can only be updated through the IEFSSVT macro.

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:
Appendix B — Using IEFJSVEC

- 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:
- IEFVTSP
- IEFJSBVT

The requirements for the caller of IEFJSVEC are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</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>Control Parameters</td>
<td>The VTSPL and JSBVT control blocks must reside in storage below 16 megabytes</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>Recovery</td>
<td>The caller of IEFJSVEC should provide an ESTAE-type of recovery environment. See Authorized Assembler Services Guide for more information on an ESTAE-type of recovery environment.</td>
</tr>
</tbody>
</table>

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>
</tbody>
</table>
Appendix B — Using IEFJSVEC

VTSLEN
Length of the VTSPL (VTSSIZE) control block

VTSVER
Version number of the VTSPL (VTSCVER) control block

VTSCONID
The 1-byte console ID of the console that the subsystem initialization
routine issues messages to. If your program sets this field to zero, the
VTSCNSID field is used.

This field exists for versions of MVS previous to SP410. IBM recommends
that you specify a 4-byte console ID as defined by the VTSCNSID field.

VTSREQ
Request flags - defines the operation that this call performs

• VTSCDIS — Disable indicator

VTSNAME
Subsystem name. The name of the subsystem for which one or more
function codes are to be disabled. The subsystem name can be up to four
characters. It must be left-justified and padded to the right with blank
(X'40') characters.

VTSSVTD
Address of SSVT table data (see JSBVT contents)

VTSCNSID
4-byte console ID that the SSI uses for any messages issued on this
invocation of IEFJSVEC. If this field is set to zero, the messages go to the
master console.

Provide a CART and a console ID if IEFJSVEC is invoked while running
under a command processor. For example, if a subsystem is initialized
through START command processing. See 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.

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 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:
Appendix B — Using IEFJSVEC

Field Name | Description
-----------|-------------
JSBID      | Identifier 'JSBV'
JSBLEN     | Length of fixed header section
JSBVERS    | Version number (JSBCVERS)
JSBFUN     | Number of function routines specified in this table of data

All other fields in the fixed header section of the JSBVT control block must be set to binary zeros.

**JSBVT Contents — Variable Length Section:** The JSBVT fixed header is followed by a variable length function routine data area (one for each function routine). Your program must set the following fields on input:

Field Name | Description
-----------|-------------
JSBLGTH    | Length of this function routine's data area and it's function code area (see the JSBFCOD field)
JSBNME     | Name of a function routine. The function routine name can be up to eight characters. It must be left-justified and padded to the right with blank (X'40') characters.
JSBNUM     | Number of function codes

The JSBNUM field should be set to the number of function codes which are to be disabled for this function routine as specified in the JSBFCOD field on this invocation of IEFJSVEC.

**JSBVT Contents — Variable Length Section:** The function routine data area is followed by a variable length function code area (one for each function routine). Your program must set the following field on input:

Field Name | Description
-----------|-------------
JSBFCOD    | Function code (repeat if more than one function code is to be disabled). The value specified for each function code, must be in the range 1-255.

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

<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:

Return Code
(Decimal)
Meaning
VTSSUCES (0)
Successful completion. The request to disable was successfully processed
and the SSVT has been updated.
VTSINVID (4)
An incorrect identifier was specified in VTSPL or JSBVT. Check the input
parameter areas to make sure that you specified the proper identifiers, and
that the pointers to the input parameter areas are properly defined.
VTSINVIN (8)
An incorrect subsystem name was specified. Check to make sure that you
specified a valid subsystem name in the VTSNAME field. Consult with
your system programmer to make sure that it matches the name of a valid
subsystem defined in the IEFSSNxx parmlib member that is currently in
use.
VTSLOGER (16)
Logic error. Contact your IBM service support center.
VTSINVBI (24)
An incorrect bit was set in the request flags. Verify that you have set only
the VTSCDIS indicator and that you have not set any other bits in the
VTSREQ flag byte.
VTSINVLED (32)
Unable to process disable request; no SSVT found. Verify that you specified
a valid subsystem name in the VTSNAME field. If the subsystem name is
valid, make sure that the subsystem's SSVT has been built and is properly
pointed to from your subsystem's SSCVT prior to any IEFJSVEC disable
calls being made.

Output Parameters
Output parameters for the IEFJSVEC service are:
• VTSPL

VTSPL Contents: The VTSPL control block contains the following information
upon return from your disable request:

Field Name
Description
VTSSSCVT
Address of the SSCVT, if the disable request was successful (Register 15=0)
Appendix. Accessibility

Accessible publications for this product are offered through the z/OS Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrdfs@us.ibm.com or to the following mailing address:

IBM Corporation
Attention: MHVRDFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features

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.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually
exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1!
(KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

**Note:**
1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.
Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

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
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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.

### Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS](http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

### 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 at [Copyright and Trademark information](http://www.ibm.com/legal/copytrade.shtml).
Index

A
accessibility 591
contact IBM 591
features 591
additional recommendations for specifying keywords
related to function code 54 71
address space
subsystem 463
application thread
caller of the SSI function code 79 155
ASCRE macro
using to create a separate address space 464
assistive technologies 591
associate a new function routine with a supported function code
use of 477
with the IEFSSVT macro 477
automatic restart manager
considerations 55

B
batch jobs
initiating a job 8
broadcast request
description 2
build the SSVT
considerations 475, 575
with the IEFJSVEC service 575
with the REQUEST=CREATE parameter 475
BULK MODIFY SAPI Call
SSI Function Code 79 144

C
command processing call - SSI function code 10
considerations 517
considerations for system symbols 518
description 511
restrictions 517
sysplex considerations 518
command processing call - SSI function code 14
considerations 522
description 519
restrictions 521
command sensitive area
contents 516
for a REPLY command 516
considerations
automatic restart manager 55
COUNT SAPI Call
SSI Function Code 79 144
create an SSIB
steps 9

dalyrtkn text unit
use of 146
daldsnamin text unit
use of 29
use of for SSI Function Code 79 149
dalrtddn text unit
use of 29
dalsreq text unit
use of 29
use of for SSI Function Code 79 149
define your subsystem
description 464
directed request
description 2
disable previously supported functions
use of 477, 585
with the IEFJSVEC service 585
with the IEFSSVT macro 477
dundddnam text unit
use of 30, 146
dunovals text unit
use of 30
dunovalsp text unit
use of 30
dunovalsn text unit
use of 30
dynacic SSI
description 4

E
early notification of end-of-task call - SSI function code 50
description 524
enable your subsystem for new functions
use of 476, 580
with the IEFJSVEC service 580
with the IEFSSVT macro 476
end-of-address space (end-of-memory) call - SSI function code 8
description 491
end-of-task call - SSI function code 4
description 487
environment on entry to a function routine
description 462
register contents 462
example
for extended status function call - SSI function code 80 246
for request subsystem version information call - SSI function code 54 71
for the process SYSORT data sets call - SSI function code 1 33
passing accounting parameters 470
extended status function call - SSI function code 80
example 246
Extended Status function call - SSI function code 80
description 185
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 503
fixed header input section
contents 63
description 61
fixed header output section
contents 66
description 66
format of the variable output sections 68
function code descriptions
for SSI function codes your program can request 13
function codes (SSI)
description 7
list of 13, 487
requirements 461
function routines
placement 463
requirements 461

H
help call - SSI function code 48
description 522

I
IBM-defined keywords
related to function code 54 68
ieavg700 module
calling the module 495
iefjsipl mapping macro 472
iefssib mapping macro 8
iefjsvec service
disabling functions 585
enabling functions 580
using to build the SSVT 575
iefssnxx parmlib member
planning 464
iefssobh mapping macro 7
iefssreq macro
description 9
introduction 7
syntax 10
use of 9
with subsystem affinity service 484

© Copyright IBM Corp. 1988, 2013

599
IEFSSVT macro
  disabling functions 477
  enabling functions 476
  replacing the function routine 477
  use to build SSVT 473
  use to disable subsystem function codes 463
  use to enable functions 463
IEFSSVTI macro
  use of 475
initial program load 465
initialization routine 473
specifying 467
subsystem 472
initialize your subsystem by specifying an initialization routine 467
by using the START command 467
input to the SSVT 475
Input-only fields 158
installation variable output section contents 68
description 68
restrictions 70
integrity subsystem considerations 462
IPL (initial program load) considerations 465

J
JES device information services
  processing request 341
  64-bit addressable virtual storage 342
request types 342
JES device information services - SSI function code 83
description 341
JES job information services - SSI function code 71
description 87
JES Properties - SSI function code 82
description 250
JES2 subsystem 1
JES3 subsystem 1
JESSPOOL SAF resource class
  use of for Function Code 79 146, 152

K
keyboard
  navigation 591
  PF keys 591
  shortcut keys 591

L
life-of-job SSIB
  description 8
  use 17, 157
load-to-global option 463

M
maintain information about subsystem callers
  subsystem affinity entry 484
  using the subsystem affinity service 483
make a request of a subsystem
  summary of steps 11
  using the IEFSSREQ macro 9
MCSOPER/MCSOPMSG macro services
  use of 497
Modify Job function call - SSI function code 85
description 431
multi-line WTO
  for SSI function code 9 503
  SSWT contents 503
  subsequent lines 506
  WQE (minor WQE) contents 506

N
name your subsystem
  restrictions 464
navigation
  keyboard 591
Notices 595
notify user message service call - SSI function code 75
description 137

O
obtain a value from an entry 484

P
placement of function routines 463
  setting the load-to-global option 463
primary subsystem
  description 1
procedure of searching data strings 68
process SYOUT 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 SYOUT data sets call - SSI function code 1 29
    processing all data sets together 30
    processing one data set at time steps 29
PUT/GET SAPI Call
  SSI Function Code 79 144
request a function of a subsystem
  steps 7
request command processing information
  description 511, 519
request job ID call - SSI function code 20
description 49
  restrictions 55
request subsystem version information
  installation-defined keywords 70
request subsystem version information call - SSI function code 54
description 60, 528
example 71
request types
  for SYOUT Application Program Interface 143
restrictions for SSI function code 10 517
restrictions for SSI function code 14 521
return code information
  for command processing call - SSI function code 10 517
  for delete operator message - SSI function code 14 521
  for early notification of end-of-task call - SSI function code 50 528
  for end-of-address space (end-of-memory) call - SSI function code 8 495
  for end-of-task call — SSI function code 4 491, 524
  for extend status function call - SSI function code 80 208
  for Initiator Information - SSI Function Code 82 296
  for JES Device Information Services - SSI Function Code 83 355
  for JES job information services - SSI Function Code 71 93
  for JES Job Information Services - SSI Function Code 71 101, 114, 126, 132
  for JESPLEX Information - SSI Function Code 82 313
  for Job Class Information - SSI Function Code 82 328
  for modify job function call - SSI function code 85 451
  for NJE NODE Information - SSI Function Code 82 257
  for notify user message service call - SSI function code 75 141
  for process SYOUT data sets call - SSI function code 1 23
  for request job ID call - SSI function code 20 53
  for request subsystem version information call - SSI function code 54 64, 531
  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 58 538
  for SPOOL Volume Information - SSI Function Code 82 275
  for SYOUT Application Program Interface - SSI function code 79 172

R
recovery
  subsystem considerations 462
  register contents on entry to a function routine 462
return code information (continued)
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 9 509
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 79 148
secondary subsystem
description 1
sending comments to IBM xi
services for building and using your
subsystem
activating your subsystem 478
adding your subsystem 471
changing what your subsystem can
do 476
deactivating your subsystem 479
defining subsystem options 481
defining what your subsystem can
do 474
description 471
initializing your subsystem 472
maintaining information about
subsystem callers 483
querying subsystem information 482
storing and retrieving
subsystem-specific information 480
swapping subsystem functions 480
services for writing your subsystem
changing what your subsystem can
do 580
defining what your subsystem can
do 575
set a value in an entry 484
SET SMF=xx command
use of 470
set up the environment
to make a request of a subsystem 7
SETSMF command
use of 470
setting up your subsystem
planning considerations 461
shortcut keys 591
single-line WTO
WQE contents 500
SJFREQ macro
use of in Function Code 79 148
SMF console command
use of 469
SMF parmlib member (SMFPRMxx)
use of 469
SMF SUBPARM option
initializing the SMF parameters 469
initializing the subsystem 469
SMF SUBPARM option (continued)
modifying the SUBPARM value 469
processing 469
SMF SUBPARM option change call - SSI
function code 58
description 533
SMFCHSUB macro
use of 470
specifying keywords
related to function code 54 70
SSAFF macro
DATA parameter 486
description 484, 485, 486
ENTRY parameter 486
OBTAIN parameter 485
parameters 485, 486
SET parameter 485
symbol 485
syntax 485
TCB parameter 486
use of 484
SSCVT (subsystem communication vector
table) 462
address of the SSCVT 462
SSI (subsystem interface) 1
attributes 1
description 1
error handling 555
examples 561
introduction 1
troubleshooting errors 555
SSI Function Code 54
use of in SSI Function Code 79 145
SSI function code 83
JES device information services 341
SSI function code descriptions
for SSI function codes your program
can request 13
for SSI function codes your subsystem
can support 487
SSI function codes
list of 13, 487
SSI function codes your program can
request
list of 13
SSI function codes your subsystem can
support
list of 487
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
SSIBLEN field
setting the field to create an SSIB 9
SSIBSSNM field
setting the field to create an SSIB 9
SSIBUSE 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
SSRTLDIST 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 10
SSRTNTUP return code value
from a directed request 11
SSRTOK return code value
from a directed request 10
SSS2BTOK field
use of 149
SSS2BULK request
use of 144
SSS2CDS field
use of 148, 150
SSS2COUN request
use of 144
SSS2CTRL field
use of 144, 149, 151
SSS2DDES field
use of 145
SSS2DELC field
use of 151
SSS2DES2 field
use of 145
SSS2DES2R field
use of 145
SSS2DEST field
use of 145
SSS2DSN field
use of 146
SSS2ECBP field
use of 148, 151, 153, 159
SSS2EODS field
use of 148, 149
SSS2FSTWB field
use of 148
SSS2FSWT field
use of 148
SSS2FST field
use of 144
SSS2PUGE request
use of 144
SSS2RBA field
use of 159
SSS2RE2T field
use of 145
SSS2RLSE field
use of 151
SSS2ROUT field
use of 151
SSS2SE2CT field
use of 151
SSS2SWB field
use of 148
SSS2SWWT field
use of 148
SSS2WRSN field
use of 148
SSS2WRTN field
  use of 148
SSSOFOR8 field
  use of 19
SSSOFORM field
  use of 19
SSSOWTRC field
  contents of on return from the
  IEFSSREQ macro 32
SSWT contents for a WTOR (always
  single-line) 508
SSWT contents for the first line of a
  multi-line WTO 503
START command
  using to initialize your
  subsystem 467
started task
  initiating a started task 8
subsystem 1
  broadcast request 2, 461
  considerations 3
  defining to MVS 464
  description 1
  diagnosing errors 555
  directed request 2, 461
  error handling 555
  examples 561
  functions 461
  IEFSSREQ macro 9
  initialization routine 472
  initializing 466
  integrity 462
  MVS use 461
  passing accounting parameters 469
  primary 1
  recovery 462
  request types 2, 461
  requesting a function 7
  secondary 1
  setting up your subsystem 461
  subsystem affinity service 483
  types 1
  writing your own subsystem 461
subsystem affinity service
  description 483, 484, 485
  SSAFF OBTAIN request 485
  SSAFF SET request 484
subsystem communication vector
  table 462
subsystem identification block 8
subsystem initialization routine
  description 466
  examples 466
subsystem options block 7
subsystem requests
  broadcast 2
  directed 2
summary of changes
  as updated December 2013 xiii
Summary of changes xiii
SWBTTUREQ macro
  use of in Function Code 79 148
SYSOUT Application Program Interface
  (SAPI) - SSI function code 79
  description 143
sysplex
  command processing SSI call - SSI
  function code 10 518
system message
  controlling 495
  with SSI function code 9 495
system symbols
  command processing SSI call - SSI
  function code 10 518
system variable output section
  contents 68
  description 68
T
  tape device selection call - SSI function
  code 78
  description 538
  trademarks 597
troubleshoot errors in your subsystem
  common types of errors 555
  description 555
  handling initialization errors 555
TSO/E user
  initiating a LOGON 9
types of subsystem requests
  broadcast 2
  directed 2
U
  unique attributes of the SSI
  description 1
  use 484
  user destination validation/conversion -
  SSI function code 11
  description 39
  user interface
    ISPF 591
    TSO/E 591
V
  verify subsystem function call - SSI
  function code 15
  description 44
  VTSCREAT SSVT build indicator 577
  VTSLOAD load-to-global indicator 577
W
  Wildcards
    SSI Function Code 79 145
    WQE (major WQE) contents for the first
    line of a multi-line WTO 503
    WQE (minor WQE) contents for
    subsequent lines of a multi-line
    WTO 506
    write your own subsystem
    considerations 3
    writer communication area
    contents 32
    description 32