JES2 Installation Exits
Twelfth edition, September 2009

This is a major revision of SA22-7534-10.

This edition applies to Version 1 Release 11 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

International Business Machines Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

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

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrdfs@us.ibm.com

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

Make sure to include the following in your comment or note:
• Title and order number of this document
• Page number or topic related to your comment

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

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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figures</td>
<td>vii</td>
</tr>
<tr>
<td>Tables</td>
<td>ix</td>
</tr>
<tr>
<td>About this document</td>
<td>xi</td>
</tr>
<tr>
<td>Summary of changes</td>
<td>xiii</td>
</tr>
<tr>
<td>Chapter 1 - Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2 - Writing an exit routine</td>
<td>7</td>
</tr>
<tr>
<td>Controlling the loading of installation-defined load modules</td>
<td>27</td>
</tr>
<tr>
<td>Enabling an exit</td>
<td>39</td>
</tr>
<tr>
<td>Getting listings of JES2 data areas</td>
<td>41</td>
</tr>
<tr>
<td>Sample exit routines</td>
<td>43</td>
</tr>
<tr>
<td>Multiple exit routines in a single module</td>
<td>45</td>
</tr>
<tr>
<td>Testing your exit routine</td>
<td>47</td>
</tr>
<tr>
<td>Tracing status</td>
<td>53</td>
</tr>
<tr>
<td>Establishing installation-defined exits</td>
<td>55</td>
</tr>
<tr>
<td>Hints for coding JES2 exit routines</td>
<td>57</td>
</tr>
<tr>
<td>Chapter 3 - IBM-defined exits</td>
<td>59</td>
</tr>
<tr>
<td>Exit 0: Pre-initialization</td>
<td>71</td>
</tr>
<tr>
<td>Exit 1: Print/punch separators</td>
<td>75</td>
</tr>
<tr>
<td>Exit 2: JOB JCL statement scan (JES2 main task)</td>
<td>81</td>
</tr>
<tr>
<td>Exit 3: JOB statement accounting field scan (JES2 main task)</td>
<td>89</td>
</tr>
<tr>
<td>Exit 4: JCL and JES2 control statement scan (JES2 main task)</td>
<td>97</td>
</tr>
<tr>
<td>Exit 5: JES2 command preprocessor</td>
<td>105</td>
</tr>
<tr>
<td>Exit 6: JES2 converter exit (subtask)</td>
<td>113</td>
</tr>
<tr>
<td>Exit 7: Control block I/O (JES2)</td>
<td>119</td>
</tr>
<tr>
<td>Exit 8: Control block read/write (user, subtask, and FSS)</td>
<td>123</td>
</tr>
<tr>
<td>Exit 9: Output excession options</td>
<td>127</td>
</tr>
<tr>
<td>Exit 10: $WTO screen</td>
<td>131</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1988, 2009
Exit 11: Spool partitioning allocation ($TRACK) ........................................... 135
Exit 12: Spool partitioning allocation ($STRAK) ........................................ 139
Exit 14: Job queue work select – $QGET ................................................ 145
Exit 15: Output data set/copy select ....................................................... 149
Exit 16: Notify ..................................................................................... 155
Exit 17: BSC RJE SIGNON/SIGNOFF ..................................................... 159
Exit 18: SNA RJE LOGON/LOGOFF ....................................................... 163
Exit 19: Initialization statement .............................................................. 167
Exit 20: End of input ............................................................................ 171
Exit 21: SMF record ............................................................................. 177
Exit 22: Cancel/status .......................................................................... 179
Exit 23: FSS job separator page (JSPA) processing ................................. 183
Exit 24: Post–Initialization ............................................................ 187
Exit 25: JCT read ................................................................................ 191
Exit 26: Termination/resource release ................................................... 195
Exit 27: PCE attach/detach .................................................................. 199
Exit 28: subsystem interface (SSI) job termination ............................... 201
Exit 29: Subsystem interface (SSI) end-of-memory ................................. 205
Exit 30: Subsystem interface (SSI) data set OPEN and RESTART .......... 207
Exit 31: Subsystem interface (SSI) allocation ......................................... 211
Exit 32: Subsystem interface (SSI) job selection .................................... 215
Exit 33: Subsystem interface (SSI) data set CLOSE ................................. 219
Exit 34: Subsystem interface (SSI) data set unallocation ....................... 223
Exit 35: Subsystem interface (SSI) end-of-task ..................................... 227
Exit 36: Pre-security authorization call .................................................. 229
Exit 37: Post-security authorization call ................................................ 235
Exit 38: TSO/E receive data set disposition ........................................... 241
Exit 39: NJE SYSOUT reception data set disposition ............................. 245
Figures

1. Areas of JES2 Modification ......................................................... 2
2. A JES2 Exit .............................................................................. 4
3. EXIT Point Variations ................................................................. 5
4. JES2 and FSS Address Spaces .................................................... 10
5. Methods of Packaging an Exit Routine ....................................... 28
6. Example of Assembly and Link-Edit of a Installation-Written Routine ......................................................... 39
7. Example of an Exit Routine Employing a User Defined Exit ............. 40
8. Example of Providing Multiple Exits within a Single Load Module ........ 45
9. Exit Routines Load Module ......................................................... 48
10. Exit Placement .......................................................................... 50
11. $DCT and $PCE data structure for a set of NJE devices using a traditional (non-TCP/IP) transport ......................................................... 350
12. $DCT and $PCE data structure for a set of NJE devices using a TCP/IP transport ......................................................... 351
13. JQE (R4 Mode - vs - z2 Mode) ..................................................... 371
14. Job Input Sources ...................................................................... 388
## Tables

1. JES2-Provided Global Assembler Variables (&VERSION and &J2VRSN) for Currently Supported JES2 Releases ................................. 16
2. Directed Load and Use of Modules Based on LOADMOD(jxxxxxx) STORAGE= Specification 27
3. Exit Selection Table ........................................................................................................ 59
4. Exit Implementation Table .............................................................................................. 68
5. Selected JES2 Job Control Table Fields ........................................................................... 90
6. Old/New Comparison of JES2 Commands ..................................................................... 106
7. Comparison of Exit 11 and Exit 12 ............................................................................... 135
8. Comparison of Exit 12 and Exit 11 ............................................................................... 139
9. Security Function Codes ................................................................................................. 230
10. Security Function Codes ................................................................................................. 237
11. Selected JES2 Job Control Table Fields ........................................................................ 312
12. $PCE names and their related data areas ...................................................................... 347
13. Exits that are affected by changes to input processing ................................................... 348
14. NJE transmitter exits affected by changes made in z/OS V1R7 ...................................... 352
15. NJE Receiver Exits Affected by Changes in z/OS V1R7 ................................................. 352
16. Common Field Changes. ................................................................................................. 360
17. The usage of $DOGJQE ................................................................................................. 361
18. Code Locations .............................................................................................................. 362
19. New z2–mode increased limits ..................................................................................... 364
20. JobID format based on job number. ............................................................................... 365
21. z2– and R4 mode comparisons – Field and control block maximum limits ................... 365
22. JQE Indexes ...................................................................................................................... 366
23. JOE Indexes ....................................................................................................................... 367
24. Four byte job number fields ......................................................................................... 369
25. Four byte EQUATES ....................................................................................................... 369
26. $SPID four byte fields .................................................................................................... 369
27. Formatted command DSECTs ....................................................................................... 369
28. Checkpoint control block fields .................................................................................... 370
29. Reader and Converter Exits Usage ................................................................................. 373
30. Job-Related Exits ............................................................................................................ 380
31. $JCT/JMR Definitions .................................................................................................... 387
32. Job Input Service Exits - Main Task ............................................................................. 388
33. Job Input Service Exits - User Environment .................................................................. 390
34. Conversion Phase Processing ......................................................................................... 392
35. Execution Phase Exits ..................................................................................................... 394
36. Spin Phase Processing .................................................................................................... 397
37. Output Phase Processing ............................................................................................... 397
38. Hardcopy Phase Processing ........................................................................................... 398
39. NJE Hardcopy Phase Processing .................................................................................... 399
40. Purge Phase Exits ........................................................................................................... 400
About this document

This document supports z/OS® (5694-A01).

This document provides system programming information concerning the use of IBM-defined and installation-defined JES2 exit routines. It describes how to establish JES2 exit routines to tailor JES2 without in-line source code modification.

Who should use this document

This document is intended for JES2 system programmers or for anyone responsible for customizing JES2.

How this document is organized

The organization and content of this document is as follows:

- Chapter 1 describes the processing concepts of JES2 exits.
- Chapter 2 describes how to write an exit.
- Chapter 3 lists the IBM-defined exits, describes how to choose which exits to implement, and what to consider when writing an exit routine.
- Appendix A describes JES2 exit usage limitations.
- Appendix B provides sample code for Exits 17 and Exit 18.
- Appendix C describes job-related exit scenarios.
- Appendix D describes z/OS product accessibility.

Where to find more information

This document references the following documents for further details about specific topics. Abbreviated forms of these are used throughout this document. The following table lists all abbreviated titles, full titles, and their order numbers that are not listed in [z/OS Information Roadmap](#). See that document for all z/OS documents.

Most licensed documents were declassified in OS/390® V2R4 and are now included on the z/OS Online Library Collection, SKT2T-6700. The remaining licensed documents appear in unencrypted BookManager® softcopy and PDF form on the z/OS Licensed Product Library, LK2T-2499.

<table>
<thead>
<tr>
<th>Short Title Used in This document</th>
<th>Title</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Structured Approach to Describing and Searching Problems</td>
<td></td>
<td>SC34-2129</td>
</tr>
</tbody>
</table>
The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required): http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp

Additional information

Additional information about z/OS elements can be found in the following documents.

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Introduction and Release Guide</td>
<td>GA22-7502</td>
<td>Describes the contents and benefits of z/OS as well as the planned packaging and delivery of this new product.</td>
</tr>
<tr>
<td>z/OS Planning for Installation</td>
<td>GA22-7504</td>
<td>Contains information that lets users:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Understand the content of z/OS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Plan to get z/OS up and running</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Install the code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Take the appropriate migration actions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test the z/OS system</td>
</tr>
<tr>
<td>z/OS Information Roadmap</td>
<td>SA22-7500</td>
<td>Describes the information associated with z/OS including z/OS documents and documents for the participating elements.</td>
</tr>
<tr>
<td>z/OS Summary of Message and Interface Changes</td>
<td>SA22-7505</td>
<td>Describes the changes to messages for individual elements of z/OS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This document is provided in softcopy only on the message bookshelf of the z/OS collection kit.</td>
</tr>
</tbody>
</table>
Summary of changes

Summary of changes
for SA22-7534-11
z/OS Version 1 Release 11

The document contains information previously presented in z/OS JES2 Installation Exits, SA22-7534-10, which supports z/OS Version 1 Release 10.

New information

- Added new section JES2 z/OS V1R11 migration details in JES2 exit migration considerations. See “JES2 z/OS V1R11 migration details” on page 345.

Changed information

- Updated Linkage conventions for Writing an exit routine. See “Linkage conventions” on page 11.
- Updated Exit 0. See “Exit 0: Pre-initialization” on page 71.
- Updated Exit 1. See “Exit 1: Print/punch separators” on page 75.
- Updated Exit 15. See “Exit 15: Output data set/copy select” on page 149.
- Updated Exit 46. See “Exit 46: Modifying an NJE data area before its transmission” on page 275.
- Updated Exit 56. See “Exit 56: Modifying an NJE data area before its transmission” on page 333.
- Updated section Checkpoint control blocks in SPOOL control blocks for Job-related exit scenarios. See “Checkpoint control blocks” on page 382.

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

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

Summary of changes
for SA22-7534-10
z/OS Version 1 Release 10

The document contains information previously presented in z/OS JES2 Installation Exits, SA22-7534-09, which supports z/OS Version 1 Release 9.

New information

- Added a section in Chapter 2. See “Dynamic Load Modules” on page 29.

Changed information

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

Summary of changes
for SA22-7534-09
z/OS Version 1 Release 9

The document contains information previously presented in z/OS JES2 Installation Exits, SA22-7534-08, which supports z/OS Version 1 Release 8.

New information
- None.

Changed information
- Updated Exit 31. See "Exit 31: Subsystem interface (SSI) allocation" on page 211.
- Updated Exit 42. See "Exit 42: Modifying a notify user message" on page 257.
- Updated Exit 45. See "Exit 45: Pre-SJF service request" on page 269.

Deleted information
- None.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.
Chapter 1 - Introduction

JES2 is a general job entry subsystem of z/OS and sometimes cannot satisfy all installation-specific needs at a given installation. If you modify JES2 code to accomplish your specific functions, you then are susceptible to the migration and maintenance implications that result from installing new versions of JES2. JES2 exits allow you to modify JES2 processing without directly affecting JES2 code. In this way, you keep your modifications independent of JES2 code, making migration to new JES2 versions easier and making maintenance less troublesome.

Attention!

Defining exits and writing installation exit routines is intended to be accomplished by experienced system programmers; the reader is assumed to have knowledge of JES2.

If you want to customize JES2, IBM® suggests that you use JES2 installation exits to accomplish this task.

IBM does not recommend or support alteration of JES2 source code. If you assume the risk of modifying JES2, then also assure your modifications do not impact JES2 serviceability using IPCS. Otherwise, IBM Level 2 Support might not be able to read JES2 dumps taken for problems unrelated to the modifications.

Avoid expanding JES2 control blocks. Use alternatives such as:

1. Use fields dedicated for installation use that appear in many major control blocks. Place your data, or a pointer to your data, in these fields. However, beware of setting storage addresses in checkpointed or SPOOL–resident control blocks.
2. Use $JCTX services rather than modifying $JCT.
3. Use table pairs and dynamic tables. For example, use dynamic $BERTTABs with CBOFF=∗ instead of modifying $JQE.

This is a partial list. Evaluate your specific situation and take appropriate action.

Note!

JES2 operates in full–function mode (z2 mode under z/OS). All discussion in this document assumes JES2 is running in z2 mode. JES2 no longer supports compatibility mode (R4 mode under OS/390). For more discussion about migrating from R4 mode to z2 mode, see z2 mode for $ACTIVATE on page 363. As of z/OS V1R7 JES2, the $ACTIVATE command has been deleted.
Figure 1, and the text that follows it, illustrates many of those areas where you can modify JES2 processing using the JES2 exit facility.

**Initialization Processing**
You can modify the JES2 initialization process and incorporate your own installation-defined initialization statements in the initialization process. Also, you can change JES2 control blocks before the end of JES2 initialization.

**Job Input Processing**
You can modify how JES2 scans and interprets a job’s JCL and JES2 control statements. Also, you can establish a job’s affinity, execution node, and priority assignments before the job actually runs.

**Subsystem Interface (SSI) Processing**
You can control how JES2 performs SSI processing in the following areas: job selection and termination, subsystem data set OPEN, RESTART, allocation, CLOSE, unallocation, end-of-task, and end-of-memory.

**JES2-to-Operator Communications**
You can tailor how JES2 communicates with the operator and implement additional operator communications for various installation-specific conditions. Also, you can preprocess operator commands and alter, if necessary, subsequent processing.

**Spool Processing**
You can alter how JES2 allocates spool space for jobs.

**Output Processing**
You can selectively create your own unique print and punch separator pages for your installation output on a job, copy, or data set basis.

**JES2-SMF Processing**

*Figure 1. Areas of JES2 Modification*
You can supply to SMF added information in SMF records.

- **RJE Processing**
  You can implement additional security checks to control your RJE processing and
gather statistics about signons and signoffs.

**What is a JES2 exit?**

JES2 exits provide a clean, convenient, relatively stable interface between JES2 and your installation-written code. Installation-written exit routines are invoked from standard JES2 processing at various strategic locations in JES2 source code. These strategic locations in JES2 source code are called *exit points*. A JES2 exit is established by one or more exit points.

An exit point is defined by the $EXIT macro and, as illustrated in Figure 2, is the exact location in JES2 code where JES2 can pass control to your *exit routine* (that is, your installation-written code). The JES2 exit, identified by the “exit-id code” of nnn, is defined by one exit point at label JLBL in the JES2 code. It is at JLBL in JES2 processing that JES2 passes control to your exit routine.

To use the exit facility you perform the following steps, as illustrated in Figure 2.

1. Package your code into one or more exit routines, identifying each exit routine with an entry point name. (In Figure 2 there is a series of exit routines noted as entry points X1...Xn.) Then include the exit routine in a load module. In this case LMOD is the load module containing the exit routine.

2. In the JES2 initialization stream include the LOADmod(jxxxxxxx) initialization statement, which causes your exit routine’s load module to be loaded into either private (PVT), common (CSA), or to locate the module in link pack area (LPA) storage. The linkage editor RMODE attribute determines whether the system loads the module above or below 16 megabytes.

   Also include the EXIT(nnn) initialization statement, which associates your exit routines’ entry point with the exit point in the JES2 code. The EXIT(nnn) initialization statement matches the exit point “nnn” at label JLBL for the $EXIT macro in the JES2 code. The EXIT(nnn) initialization statement identifies the label “X1” as the entry point of the exit routine for exit point “nnn”. The LOAD initialization statement identifies LMOD as the load module to be loaded into storage.
JES2 can have up to 256 exits, each identified by a number from 0 to 255. You specify the number on the required “exit-id code” parameter on the $EXIT macro.

This exit-id code identifies the JES2 exit. When more than one exit point is defined for a single exit, the $EXIT macros that defined the multiple exit points have unique labels but are all specified with the same exit-id code – see Figure 3.
JES2 code includes a number of IBM-defined exits. That is, various exit points – through the $EXIT macro – have already been strategically placed in the JES2 code. The intended purpose of each of these exits is summarized in Table 3 on page 59. For these IBM-defined exits you need only write your own exit routines and incorporate them through the EXIT(nnn) initialization statement and the LOADmod(jxxxxxxx). The selection of the point in JES2 code where the exit point should be placed has already been done for you. To ensure a proper implementation, you should thoroughly understand the IBM-defined exit and its JES2 operating environment. A comprehensive description of each exit is presented in “Chapter 3 - IBM-defined exits” on page 59.

Also, the JES2 exit facility allows you to establish your own exits, should the IBM-defined exits not suffice. Exits established by you are modifications to JES2 and are called installation-defined exits, and you define them by placing the $EXIT macro yourself at appropriate points in the JES2 code (or in your own exit routine code). Note, however, that implementing your own exit can be considerably more difficult than writing an exit routine for an IBM-defined exit. You should realize that in establishing your own exits, you run a greater risk of disruption when installing a new version of JES2 code. The new JES2 code into which you have placed your exits may have significantly changed since your $EXIT macros were inserted. For more information, see “Establishing installation-defined exits” on page 55.

Every exit, both IBM-defined and installation-defined, has a status of enabled or disabled which is set at initialization through the EXIT(nnn) initialization statement and which can be dynamically altered by the $T EXIT(nnn) operator command. When an exit is enabled, JES2 checks for the existence of an associated exit routine and then passes control to the exit routine. If no associated exits are found, standard JES2 processing continues. For certain exits, called job-related exits, (see “Job-related exits” on page 51) the status can be altered on a job-by-job basis by the action of an exit routine. When an exit is disabled for a particular job (by use of the job mask), it is automatically bypassed by standard JES2 processing.

Environment

The following topics describe the environment in which the JES2 exits run.

General

JES2 operates in four environments: JES2 main task, JES2 subtask, user environment, and functional subsystem (FSS) environment. Your exit routine receives control as fully-authorized extensions of JES2, and as such receives control in one of these four environments depending on where the associated exit point is placed. JES2 main task and subtask exit points exist in the HASJES20 load module.
**Program authority**
Your exit routine has access to various control blocks and service routines to which the standard JES2 code has access at the exit point, and it runs with the same authorization as the JES2 code from which your exit routine was invoked. Exit routines invoked from the JES2 address space run in supervisor state in either the JES2 main task or JES2 subtask environment with a protect key of “1”. Exit routines invoked from the user environment execute in key 0. Exit routines invoked from the functional subsystem (FSS) address space run in the FSS environment and typically run in protect key 1 (as set by the FSS). Also, exit routines invoked from the FSS address space have access to all service routines supported by HASPFSSM.

**Exit linkage**
A JES2 exit effector provides linkage services between an exit point and exit routines. It locates and passes control to your exit routines and returns control to JES2. There are two exit effectors: one provides linkage to exit routines that run as extensions to the JES2 main task and the other provides linkage to exit routines that run as extensions to JES2 subtasks or as extensions to routines in the user address space or the FSS.

**Return codes**
Your exit routines can affect JES2 processing by directly manipulating JES2 data areas and by passing back return codes. You can have up to 256 individual exit routines associated with a single exit on the EXIT(nnn) initialization statement. These multiple exit routines are all called consecutively in the order of their appearance on the EXIT(nnn) initialization statement. Consider the following example:

```
EXIT(175) ROUTINE=(X1,X2,X3,X4,X5,...)
```

For Exit 175, the exit routine identified by label X1 is called before the exit routine identified by X2, and so forth, until all of them (X1 through X5) are called or until one of them generates a nonzero return code, which causes the exit effector to return to the JES2 mainline after the exit point.

**Installation**
IBM suggests that any modifications to JES2 code or the installation of JES2 exits be performed utilizing the functions of SMP/E (System Modification Program Extended). This requires the preparation of SMP/E control statements and constructs suitable for SMP/E processing. Applying changes in an SMP/E-controlled environment prevents down-leveling or the application of release incompatible maintenance.

In the case of JES2 exits, if the application of PTF maintenance changes any macros or other components used by the exits, then the affected modules will automatically be reassembled by SMP/E.

For more information about SMP/E, see [SMP/E User’s Guide](#).

**Note:** No exit routines are ever required as part of standard JES2 processing. The JES2 exit facility is fully optional. If you have not implemented an exit—that is, if you have not written an exit routine for it, or have not included the exit routine in a load module, or have not associated the routine with the exit at initialization time—the presence of the exit point or points that establish the exit is transparent during standard JES2 processing.
Chapter 2 - Writing an exit routine

When you are planning to write a JES2 exit routine, you need to consider the environment in which the exit routine runs and other general programming considerations (such as, the programming language to use to code your exit routine, linkage conventions that are required, return codes to set, and reentrant code requirements to follow). "Chapter 3 - IBM-defined exits" on page 59 provides the specific programming considerations you need for writing exit routines for the IBM-defined exits. You should use "Chapter 3 - IBM-defined exits" on page 59 with the information in this chapter when writing your exit routine. Should you decide to implement your own installation-defined exit in JES2, you need to investigate all the exit-specific programming considerations yourself. See "Establishing installation-defined exits" on page 55 for more information.

Note: All exit modules must be in APF authorized libraries.

Language

You must write JES2 installation exit routines in basic assembled language. To assemble JES2 or installation exit routines, use High-Level Assembler or any compatible IBM assembler.

Operating environment

For security reasons, the caller of an installation-defined exit in the user’s address space must be either in supervisor state or be an authorized program. JES2 will terminate a calling routine with neither of these attributes with a privileged operation exception.

JES2 environments

When writing an exit routine, you must consider the calling JES2 environment, because your exit routine runs as an extension of that calling environment (JES2 main task, JES2 subtask, user address space, and functional subsystem). The calling environment has broad implications to your exit routine; it determines the JES2 system services available to your exit routine, the reentry considerations you should consider, the linkage conventions that are necessary, and several other essential factors (such as, control block access, synchronization, recovery, and JES2 programmer macro usage). Specifically, the use of macros in exit routines is limited. Before attempting to use a particular macro in an exit routine, be certain to check the “Environment” section of each macro description in Chapter 4 to determine the environments in which the macro can be used.

Every exit is explicitly defined to JES2 as belonging to one of the four execution environments. The ENVIRON= operand of the $MODULE macro is specified as either “JES2”, “SUBTASK,” “USER,” or “FSS”. This specification determines which of two exit effectors (the JES2 subroutines that establish the linkage between JES2 and an exit routine) will be called when the exit is enabled. One exit effector establishes linkage to an exit routine from the JES2 main task environment; the other establishes linkage to an exit routine from either the JES2 subtask environment, the user environment or the FSS. In all environments (JES2 main task, functional subsystem, subtask, and user environment) JES2 linkage conventions (that is, $SAVE and $RETURN) are used.
You cannot define an exit “across” environments. That is, when an exit is required to serve the same purpose in two distinct environments, two separate exits must be defined, each with its own identification number. For example, Exit 11, an IBM-defined exit that can give you control to reset the spool partitioning mask, belongs to the JES2 main task environment. Exit 12, which serves the same functional purpose, belongs to the user environment. In implementing these exits, you must write a separate exit routine for each defined exit and adapt the routine to its calling environment.

To stress again, whether defining an exit or writing an exit routine, you must be aware of the operating environment; it influences where your exit is to be defined or what processing your exit routine can really perform. In the descriptions of the following general programming considerations for writing an exit routine, specific environmental influences are described.

JES2 has four execution environments - maintask, subtask, user, and functional subsystem (FSS).

1. **JES2 Main Task** - The JES2 main task is the most common operating environment for JES2 exits. The JES2 main task routines are included in the JES2 load module HASJES20 which is loaded in the private area of the JES2 address space. JES2 main task routines run under the control of the JES2 dispatcher (in HASPNUC). The load module, HASPINIT, which performs JES2 initialization, runs under the main task but is not controlled by the JES2 dispatcher.

The execution of maintask routines, with the exception of asynchronous routines such as I/O appendages, are controlled by the JES2 dispatcher and are represented by a dispatching unit called processor control elements ($PCEs). $PCEs, which are analogous to task control blocks (TCBs) in MVS™, are the dispatchable elements in JES2 maintask.

There are two important coding considerations in the JES2 maintask environment.

- **JES2 Reentrancy** - An exit routine called from the JES2 main task must be reentrant in the JES2 sense. Because JES2 processors ($PCEs) do not relinquish control to another JES2 processor involuntarily, an exit routine, invoked out of a main task processor may use a JES2 nonreentrant work area. Therefore, the work area is serialized unless the exit routine issues a $WAIT macro (or service called from an exit routine issues the $WAIT macro). When the exit routine issues the $WAIT macro directly or through a called routine, control returns to the JES2 dispatcher and the serialization on the nonreentrant work area ceases. The nonreentrant work area may also be passed between exit routines, or between an exit routine and JES2, before a $WAIT macro call. Work areas to be used “across” a $WAIT must either be within the processor’s work area established as part of the $PCE or else must be directly owned by the processor. In the same JES2 reentrant sense, an exit routine may search or manipulate a JES2 queue (for example, job queue or job output table) providing it has ownership of the queue (through the $QSUSE macro) and doesn’t issue a $WAIT macro until the search routine is completed.

- **MVS WAITs** - The JES2 dispatcher controls all processing within the maintask environment; therefore, no routine or exit may issue any macro or call any service that could result in the execution of an MVS WAIT macro. Issuing MVS WAITs in JES2 maintask is contrary to the design of JES2 and will cause performance problems.
An exception to this rule is JES2 initialization and JES2 termination. During initialization and termination, maintask processing is essentially single threaded. That is, there is only one $PCE dispatched so that JES2 reentrancy is not a factor. This also removes the concern about MVS WAITs causing a performance problem because during JES2 initialization and termination JES2 is not providing system services for other subsystems, started tasks, time sharing sessions and batch jobs. Therefore, there are no restrictions about MVS WAITs and MVS macros that can result in MVS WAITs in JES2 exits 0, 19, 24, and 26.

If it is necessary to invoke MVS services from JES2 maintask exits that may cause MVS waits, these services should be invoked from a subtask environment. The $SUBIT macro can be used to cause a routine to execute in a subtask environment. The WAIT/POST synchronization of the subtask is provided as part of this service.

2. JES2 Subtask - JES2 subtasks run in the private area of the JES2 address space but run asynchronously with the JES2 main task. Subtasks run under the control of the MVS dispatcher (not the JES2 dispatcher) and their asynchronous operation allows them to perform the WAIT/POST type processing without imposing the same WAIT/POST operations on the JES2 main task. System-wide MVS services are available to programs in this environment. Many JES2 maintask data areas are directly addressable, but users of these resources must understand when and where serialization of these resources is relevant. Most importantly, subtask should not directly reference the checkpoint area (job queue, job output table, and so on), because in certain portions of the checkpoint cycle this storage area is not addressable. If a subtask requires a view of the checkpoint, use the JES2 checkpoint versioning facility and the appropriate SSI calls.

3. User Environment - Some JES2 routines are loaded into common storage (located either in extended or non-extended LPA, PLPA, or CSA) execute in the user's address space. This environment, which permits user programs to interface with JES2, differs greatly from the JES2 maintask environment. System-wide MVS services are available to programs in this environment, but the environment is also more complex. It involves many integrity, synchronization, locking and cross-address space communications considerations. JES2 services in the user environment are limited. A special operating environment you can use called (USER,ANY). It is intended for environments where a routine is able to be invoked in the USER run-time environment, or under the JES2 main task. For example, Use (USER,ANY) to write a common routine invoked by both Exit 2 and Exit 52. To use it, you can code ENVIRON=(USER,ANY) on your $MODULE statement or on a $ENVIRON macro invocation. The (USER,ANY) environment is similar to the USER environment (for instance, R11 is the HCCT address) except for the following differences in the way that $SAVE and $RETURN services are implemented:
   a. If the routine is called by the JES2 main task, JES2 main task $SAVE/$RETURN services are called. This allows the possibility of a $WAIT within the routine. With a user-environment $SAVE that uses the linkage stack, this processing is not possible.
   b. In any environment, a PSV-type save area is obtained rather than using a BAKR to save the registers and environment. This allows services such as $STORE and $RESTORE to be used in any environment.

4. FSS Environment - The functional subsystem (FSS) resides in the functional subsystem address space. This environment is similar to the user environment in that JES2 services are limited. You must consider task interaction within the FSS. All data areas and control blocks are not accessible from the FSS. The
accessible control blocks are the job output element (JJOE) JOE information block (JJIB), FSS control block (FSSCB), and FSA control block (FSACB). System-wide MVS services are available to programs in this environment.

Figure 4. JES2 and FSS Address Spaces

Synchronization

An exit routine must use synchronization services appropriate to its calling environment.

An exit routine called from the JES2 main task must use the JES2 $WAIT macro to wait for a JES2 event, resource, or post of a MVS ECB. An exit routine called from a JES2 subtask or from the user environment must use the MVS WAIT macro to
wait for a system event. An exit routine called from a functional subsystem must also use MVS WAIT; $WAIT and $POST are not valid in this environment.

A JES2 main task exit routine should not invoke operating system services which may wait (WAIT), either voluntarily or involuntarily. Be aware of any product that interfaces with JES2 and attempts to issue MVS services such as STIMER, STIMERM, WAIT, or TTIMER under the JES2 main task, or which invoke MVS services such as allocation, which may issue such macros. An MVS wait from a JES2 main task exit routine would stop all of the JES2 main task processors, including any devices—such as readers, printers, and remote terminals—under their control.

Reentrant code considerations

Reentrant code considerations are contingent on the calling environment.

An exit routine called from the JES2 main task must be reentrant in the JES2 sense. The JES2 dispatching unit, commonly called JES2 processors, running under a processor control element (PCE) perform the processing for the JES2 main task. The JES2 dispatcher controls what PCE is currently active (that is, what JES2 processor is currently running). Because a JES2 processor doesn’t relinquish control to another JES2 processor involuntarily, an exit routine, invoked out of a JES2 main task processor may use a nonreentrant work area; the work area is serialized if the exit routine doesn’t issue a $WAIT macro or until the exit routine or service called from an exit routine does issue the $WAIT macro. When the exit routine issues the $WAIT macro directly or through a called routine, control returns to the JES2 dispatcher and the serialization on the nonreentrant work area ceases. The nonreentrant work area may also be passed between exit routines, or between an exit routine and JES2, before a $WAIT macro call. Work areas to be used “across” a $WAIT must either be within the processor work area established as part of the processor control element (PCE) or else must be directly owned by the processor. In the same JES2 reentrant sense, an exit routine may search or manipulate a JES2 queue providing it has ownership of the queue and doesn’t issue a $WAIT macro until this action is completed.

An exit routine called from a JES2 subtask, from the user environment, or from the FSS environment must be reentrant in the MVS sense. The exit routine must be capable of taking an MVS interrupt at any point in its processing. The exit routine must be able to handle the simultaneity of execution with other subtasks and user address space, or functional subsystem (FSS) routines and with the JES2 main task.

The following actions may produce unpredictable results:

- Modifying control block fields designed for use by the JES2 main task only (for example, $DOUBLE, $GENWORK, and so on.)
- Accessing checkpointed data from the subtask, user, or FSS environment.

Linkage conventions

When control is passed to an exit routine, certain general registers contain linkage information. Register 15 always contains the entry point address of the exit routine, and can be used to establish addressability for the exit routine’s code. Register 14 contains the address (in the exit effector) to which the exit routine must return control. In the JES2 main task environment, register 13 always contains the address of the processor control element (PCE) of the processor that invoked the
exit. In the JES2 subtask environment or the user environment, register 13 always contains the address of an 18-word save area. In the JES2 main task and subtask environments, register 11 always contains the address of the HCT; and in the functional subsystem environment (HASPFS), register 11 always contains the address of the HASP functional subsystem communications table (HFCT). In the user environment, register 11 always contains the address of the HASP common communication table (HCCT). Depending on the exit, registers 0 and 1 might be in use as parameter registers. The use of registers 2 through 10 and 12, typically used as pointer registers, is also exit-dependent.

Some JES2 services are running in 64-bit addressing mode. These services, regardless of whether they are called directly or invoked by a macro, need register 11 to contain a 64-bit pointer to the HCT, HCCT, or HFCT. When JES2 invokes an exit, it ensures that register 11 is a valid 64-bit pointer. Because exits should not need to know which services are running in 64-bit addressing mode, the invoked exit should not corrupt the high order 33 bits of register 11 before invoking any JES2 service.

The use of registers 0 through 15 is documented, for each IBM-defined exit, in the category REGISTER CONTENTS WHEN CONTROL IS PASSED TO THE EXIT ROUTINE. Note that if you install an optional installation-defined exit, you are responsible for modifying JES2 code, preceding your exit, to load any parameters in registers 0 and 1 and any pointers in registers 2 through 10 and 12 that are required by your exit routine.

For multiple exit routines, the exit effector passes registers 2 through 13 to each succeeding exit routine just as they were originally loaded by JES2 when the exit was first invoked. However, register 15 contains the entry point address of the current exit routine and, again, can be used to establish addressability for the exit routine’s code. Register 14 contains the address to which the exit routine must return control. This allows you to pass the information to consecutive exit routines. For more information, see "Multiple exit routines in a single module" on page 45.

When any exit routine receives control, it must save the caller’s registers. An exit routine called from any environment can save the caller’s registers by issuing the JES2 $SAVE macro.

When any exit routine relinquishes control, it must restore the caller’s registers, except for registers 0, 1, and 15. An exit routine called from any environment must restore the caller’s registers by issuing the JES2 $RETURN macro.

Just before returning control to JES2, an exit routine must place a return code in register 15 and must place any parameters that it intends to pass, either back to JES2 or to the next consecutive exit routine, in registers 0 and 1. If the return code is greater than zero, or if the current exit routine is the last or only exit routine associated with its exit, this return code is passed back to JES2 at the point of invocation, along with any parameters placed in registers 0 and 1. If, however, the return code is zero and the current exit routine is not the last or only exit routine associated with its exit, the exit effector passes control to the next consecutive exit routine, along with any parameters placed in registers 0 and 1.

IBM suggests that when using BAKR/PR instructions for routine linkage, that you do not use the JES2 dispatching service, $WAIT, or call any other routines that may result in a $WAIT. JES2 uses a process of sub-dispatching units of work (PCEs), under a single task.
BAKR is an instruction where a linkage-stack branch stat entry is formed. If a stack entry is created while a unit of work (PCE) is in control and that unit of work is suspended by use of the $WAIT macro, then the next unit of work to get control could change the state of these stack. Unpredictable results will occur when the PCE that was $WAITED gets control back and issues a PR instruction.

Special processing in the JES2 dispatcher detects when a PCE issues a $WAIT while there is something on the linkage stack. An abend, with reason code $DP2, will be issued to prevent this logic error from propagating more problems. Note that you can use the $STORE macro before the $RETURN macro to modify the returned values of registers 0 and 1.

Addressing mode of JES2 exits

All JES2 code (except those sections of code associated with restricted MVS services) runs in 31-bit addressing mode. In this manner, JES2 is able to take advantage of the increased virtual storage provided by the operating system 31-bit addressing mode. (See z/OS MVS Programming: Assembler Services Guide for a more complete discussion of 31-bit addressing and required operating systems considerations.)

Addressing mode requirements

All JES2 exit routines:

• are entered in 31-bit addressing mode
• return in 31-bit addressing mode
• must have all input address parameters to the exit in 31-bit fields. (Although some addresses may be restricted to below a 16-megabyte address for example, the $PRPUT, $PBLOCK, and $SEPPDIR service routines. These should use the $GETBUF macro to obtain HASP-type buffers because of this restriction.)
• must be compatible with all referenced control blocks

The addressing mode may be changed within an exit by using the $AMODE macro. It is the user’s responsibility to understand the addressing mode considerations of each exit and control the mode accordingly. See the $AMODE macro description for more information.

Residency mode requirements

All JES2 installation exits can have a residency mode (RMODE) of ANY. To set the residency mode of an exit assembly module, use the RMODE= parameter on the $MODULE macro. To set the residency mode of a load module, use the linkage editor’s MODE statement.

Received parameters

Received parameters, passed by either JES2 or the preceding exit routine in registers 0 and 1, provide a method of passing information to an exit routine and of informing an exit routine of the current point of processing. For any IBM-defined exit that passes parameters (to the first or only associated exit routine), the specific parameters are documented in the REGISTER CONTENTS WHEN CONTROL IS PASSED TO THE EXIT ROUTINE category of the exit’s description. IBM-defined Exit 6, which allows you to receive control both during and after the conversion of a job’s JCL to converter/interpreter (C/I) text, presents a typical example. After a single JCL statement has been converted to an C/I text image, Exit 6 places a zero in register 0. After all of the JCL for a particular job has been converted to C/I text,
Exit 6 places a 4 in register 0. Your exit routine can determine what action to take by checking this code when it first receives control.

For some exits, the parameter registers also contain pointers to control blocks, to certain control block fields, or to other parameter lists. For a discussion of an exit routine’s use of control blocks, see the “Control Blocks” section below.

The received parameters are passed, as modified, from routine to routine. Note that if you install an installation-defined exit, you must ensure that JES2 passes any parameters required by your exit routine in registers 0 and 1; this may require some modification of JES2 source code.

Return codes

A return code provides a convenient way for an exit routine to affect the course of following JES2 processing.

The standard return codes are 0 and 4. If 0 is returned by an exit routine that is not the last or the only exit routine associated with its exit, the exit effector calls the next consecutive exit routine. However, a 0 returned by the last or only exit routine associated with its exit directs JES2 to proceed with standard processing. A 4 returned by any exit routine directs JES2 to proceed unconditionally with standard processing; any succeeding exit routines remain uncalled.

Note that a standard return code does not necessarily suggest that an exit routine has opted to take no action. You can write an exit routine to manipulate certain JES2 data areas and then, by generating a standard return code, direct JES2 to continue with normal processing based on this altered data.

The definition of return codes that are greater than 4 is exit-dependent. The specific implementation of return of return codes greater than 4 is documented for each exit under the category, RETURN CODES in each exit’s description. A brief indication of the standard processing that results from the return of 0 or 4 is also included for each exit. Note that if you install an optional installation-defined exit, you are responsible for modifying JES2 code, following your exit, to receive and act on any return code greater than 4 generated by your exit routine.

A return code is always a multiple of 4. If your exit routine passes a return code other than 0 or another multiple of 4 to JES2, results are unpredictable. Also, the $EXIT exit-point definition macro has a MAXRC= operand that specifies the exit’s maximum acceptable return code. If your exit routine generates a return code that exceeds this specification and the exit was called from the JES2 main task, the exit effector issues the $ERROR macro. If the exit was called from a JES2 subtask, from the user environment, or from the FSS environment, the exit effector issues the ABEND macro.

Control blocks

An exit routine has access to various control blocks available in the environment from which it was called.

To simplify exit coding IBM-defined exit routines provide in registers 0-13 pointers to control blocks currently in main storage. Register 1 can contain a pointer to a parameter list, which contains the addresses of control blocks currently in main storage. For a list of the specific pointers provided by an IBM-defined exit, see the REGISTER CONTENTS WHEN CONTROL IS PASSED TO THE EXIT ROUTINE
category of the particular exit’s description. Note that if you install an installation-defined exit, you have to ensure that any pointers required by your exit routine have been placed in the call registers by JES2 before invocation of your exit; this may require some modification of JES2 source code.

An exit routine can access information available in control blocks. For example, IBM-defined Exit 5, which allows you to perform your own JES2 command preprocessing, passes the address of the PCE to an associated exit routine. You can write your own command validation algorithm by writing an exit routine that checks various command-information fields in the PCE.

CAUTION:
Because an exit routine runs fully authorized, it is free to alter any field in any control block to which it has access. By altering specific fields in specific JES2 control blocks, an exit routine can pass information to JES2 and to succeeding exit routines and can thereby affect the course of later JES2 processing. Note that JES2 has no protection against any change made to any control block by an exit routine. If you modify a checkpointed control block, you must ensure that it is written to the checkpoint data set either by your exit routine or by JES2. For this reason, you should exercise extreme caution in making control block alterations.

Avoid expanding JES2 control blocks. Use alternatives such as:

- Use fields dedicated for installation use that appear in many major control blocks. Place your data, or a pointer to your data, in these fields. However, beware of setting storage address in checkpointed or SPOOL resident control blocks.
- Use $JCTX services rather than modifying $JCT.
- Use table pairs and dynamic tables. For example, use dynamic $BERTTABs with CBOFF=* instead of modifying $JQE.

This is a partial list. Evaluate your specific situation and take appropriate action.

Except where it would seriously degrade system performance, JES2 provides a reasonable amount of space in its standard control blocks for use by your exit routines. Some storage-resident control blocks, such as PCEs and DCTs, have storage reserved for exit routine use. You can use this storage to establish your own exit-related field or fields within a standard control block or, if you require more storage, you can use four of the bytes as a pointer to a work area acquired by an exit routine using the JES2 $GETMAIN, $GETBUF, and $GETWORK macros or the MVS GETMAIN macro. Disk-resident control blocks provide considerably more space for exit routine use. For performance reasons, no checkpoint-resident control blocks reserve space for use by exit routines.

In addition to using reserved space in the standard JES2 control blocks, you can define and use your own installation-specific control blocks by using the JES2 exit facility. An exit routine can use the JES2 $GETMAIN, $GETBUF, and $GETWORK macros or the MVS GETMAIN macro to acquire storage and build a control block at the appropriate point in processing. For example, a job-related control block can be built by an exit routine associated with IBM-defined Exit 2. You can then use IBM-defined Exits 7 and 8 to write your exit. installation-defined control blocks to spool and to read them from spool into main storage.

Note that if an exit routine references the symbolic name of a control block field, the DSECT for that control block must be requested in the exit routine’s module at assembly time (through the $MODULE macro). Each exit description includes a list of DSECTs normally required at assembly.
An exit routine that needs to access checkpoint control blocks must use appropriate access services. See "Checkpoint control blocks" on page 382 for more information.

Determining the JES2 release level

Other code, whether other IBM program product code, Solution Developer code, or installation-written code might need to determine what level of JES2 is installed. This can be important so that such code can determine what support is required within that code or what support JES2 provides for a particular release. The JES2-provided global assembler variables, &VERSION and &J2VRSN, provide this indication. Table 1 on page 16 provides the variable string associated with currently supported releases of JES2.

Table 1. JES2-Provided Global Assembler Variables (&VERSION and &J2VRSN) for Currently Supported JES2 Releases

<table>
<thead>
<tr>
<th>JES2 Version and Release</th>
<th>&amp;VERSION and &amp;J2VRSN String</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP5.1.0</td>
<td>'SP 5.1.0'</td>
</tr>
<tr>
<td>SP5.2.0</td>
<td>'SP 5.2.0'</td>
</tr>
<tr>
<td>OS/390 V1 R1 and higher</td>
<td>'SP 5.3.0'</td>
</tr>
</tbody>
</table>

Based on the &VERSION or &J2VRSN value, the value of the string increases for each successive JES2 release. Note that for OS/390 R1 JES2 IBM uses a string value of 'SP 5.3.0' to protect this collating sequence. Consider this value stable and not to be changed or incremented in the future.

To accommodate future JES2 releases, use the following assembly-time variables (also valid for JES2-supported releases if you have installed APAR OW17462):

Variable Description and Use

&J2LEVEL

- **Value:** Same as listed in Table 1 on page 16 except for:

<table>
<thead>
<tr>
<th>Release</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/390 R1</td>
<td>'OS 1.1.0'</td>
</tr>
<tr>
<td>OS/390 R3</td>
<td>'OS 1.3.0'</td>
</tr>
<tr>
<td>OS/390 R4</td>
<td>'OS 2.4.0'</td>
</tr>
<tr>
<td>OS/390 R5</td>
<td>'OS 2.5.0'</td>
</tr>
<tr>
<td>OS/390 R7</td>
<td>'OS 2.7.0'</td>
</tr>
<tr>
<td>OS/390 R8</td>
<td>'OS 2.8.0'</td>
</tr>
<tr>
<td>OS/390 R10</td>
<td>'OS 2.10'</td>
</tr>
<tr>
<td>z/OS V1R2</td>
<td>'z/OS 1.2'</td>
</tr>
<tr>
<td>z/OS V1R4</td>
<td>'z/OS 1.4'</td>
</tr>
<tr>
<td>z/OS V1R5</td>
<td>'z/OS 1.5'</td>
</tr>
<tr>
<td>z/OS V1R7</td>
<td>'z/OS 1.7'</td>
</tr>
<tr>
<td>z/OS V1R8</td>
<td>'z/OS 1.8'</td>
</tr>
<tr>
<td>z/OS V1R9</td>
<td>'z/OS 1.9'</td>
</tr>
<tr>
<td>z/OS V1R10</td>
<td>'z/OS 1.10'</td>
</tr>
</tbody>
</table>

- **Description:** 8-byte string defined as are &VERSION and &J2VRSN
- **HCT Field:** $LEVEL is &J2LEVEL (OS/390 only)
- **HCCT Field:** CCTLEVEL is &J2LEVEL (OS/390 only)
- **Note:** The format of this field is an 8-byte EBCDIC string; however, do not rely upon the string data for release-to-release comparisons, use &J2PLVL for that purpose.
&J2PLVL

- **Value**: A numeric value that increases by at least a value of 1 for each successive JES2 release.
- **Description**: A value that corresponds to a specific JES2 product release level as follows:

<table>
<thead>
<tr>
<th>JES2 Version/ Release</th>
<th>&amp;J2PLVL Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP5.1.0</td>
<td>24</td>
</tr>
<tr>
<td>SP5.2.0</td>
<td>25</td>
</tr>
<tr>
<td>OS/390 R1</td>
<td>26</td>
</tr>
<tr>
<td>OS/390 R3</td>
<td>27</td>
</tr>
<tr>
<td>OS/390 R4</td>
<td>28</td>
</tr>
<tr>
<td>OS/390 R5</td>
<td>29</td>
</tr>
<tr>
<td>OS/390 R7</td>
<td>30</td>
</tr>
<tr>
<td>OS/390 R8</td>
<td>31</td>
</tr>
<tr>
<td>OS/390 R10</td>
<td>32</td>
</tr>
<tr>
<td>z/OS 1.2</td>
<td>33</td>
</tr>
<tr>
<td>z/OS 1.4</td>
<td>34</td>
</tr>
<tr>
<td>z/OS 1.5</td>
<td>35</td>
</tr>
<tr>
<td>z/OS 1.7</td>
<td>36</td>
</tr>
<tr>
<td>z/OS 1.8</td>
<td>37</td>
</tr>
<tr>
<td>z/OS 1.9</td>
<td>38</td>
</tr>
<tr>
<td>z/OS 1.10</td>
<td>39</td>
</tr>
</tbody>
</table>

- **HCT Field**: $PLVL is &J2PLVL (OS/390 only)
- **HCCT Field**: CCTPLVL is &J2PLVL (OS/390 only)
- **Note**: The value itself has no inherent meaning.

&J2SLVL

- **Value**: 0 when a new &J2PLVL is created
- **Description**: A service level within the product level updated for significant JES2 updates
- **HCT Field**: $SLVL is &J2SLVL(OS/390 only)
- **HCCT Field**: CCTSLVL is &J2SLVL (OS/390 only)
- **Note**: This value will never decrease within a specific value of &J2PLVL

**Programming Notes:***

- **OS/390**
  Run-time field SSCTSUSE points to a 10-byte field structured as follows:

  | Byte 1-8 | CCTLEVEL |
  | Byte 9-10 | CCTPLVL and CCTSLVL (concatenated) |

- **Pre-OS/390**
  Run-time field SSCTSUSE points to an 8-byte field structured as follows:

  | Byte 1-8 | CCTPVRSM |

Run-time field CCTPVRSM in the HCCT is an 8-byte field that provides the &VERSION / &J2VRSN String as listed in Table 1 on page 16 or stabilized to ‘SP 5.3.0’ for OS/390.

---

**Service routine usage**

Many service routines available to the JES2 main task are also available on an exit routine called from the JES2 main task. You can include an executable JES2 macro instruction at any appropriate point in a JES2 main task exit routine. Not all service routines are available to the functional subsystem environment; those that can be
called must be appropriate. Depending on the macro, it provides inline code expansion at assembly time or else calls a JES2 service routine, as a subroutine, in execution.

An exit routine called from a JES2 subtask or from the user environment can use any JES2 service routine that can be called from its environment and any MVS service routine (SVC) that can be called from its environment. You can include a JES2- or MVS-executable macro instruction at any appropriate point in the subtask or user routine. Again, depending on the macro, it provides inline code expansion at assembly time or else calls a JES2 or MVS service routine, as a subroutine, in execution.

Exit logic

Using an exit for other than its intended purpose can increase the risk of degraded performance and system failure and may cause migration problems.

Within the scope of an exit’s intended purpose, you have a wide degree of flexibility in devising exit algorithms. For example, you can base spool partitioning on a simple factor, such as job class, or on a complex comparison of several job attributes and current spool volume usage. However, you should remember that as you increase an algorithm’s sophistication, you also increase overhead and the risk of error. Exit-specific logic considerations are provided in the “Other Programming Considerations” category for each exit description.

Logic considerations for installing installation-defined exits and for implementing them are provided in “Establishing installation-defined exits” on page 55.

Note, for both IBM-defined and installation-defined exits, that the ability to associate multiple exit routines with a single exit allows you to devise modular logic segments. Each separate function to be performed after exit invocation can be isolated in its own exit routine. This can be especially useful when you need to provide alternate types of exit processing for different received parameters.

Exit-to-exit communication

Communication among exit routines must be accomplished through mutually accessible control blocks.

Exit point-to-exit routine communication

Several JES2 installation exits, such as installation exits 27 through 35 contain a condition byte that provide a means of passing information to your exit routine. JES2 sets this byte to indicate the status of the environment at the time the exit is called. Check the bit settings in this byte to determine what (if any) processing should be done by your exit routine. See the “Register Contents When The Exit Routine Gets Control” section of each exit description for the meaning of the condition byte.

Exit routine-to exit point communication

These same exits provide an interface for your exit routine to inform the caller of your exit of the results of your exit’s processing. You turn on bits in the response byte to pass this information to the caller. This gives the caller a cumulative response from all exit routines invoked to help the caller determine how to proceed.
when control is returned to it. Your exit should not turn bits in the response byte off, as there are some occasions when some bits of the response byte are turned on initially before control is given to your exit.

Exit-to-operator communication

Except for exit routines called from the HASPCOMM module of HASJES20 and exit routines called from JES2 initialization and termination, exit routines called from the JES2 main task environment can communicate with the operator through the $WTO macro. Exit routines called from the HASPCOMM module can communicate with the JES2 operator through the $CWTO macro. Exit routines called from a JES2 subtask or during JES2 initialization and termination can communicate with the operator through the $$WTO and $$WTOR macros or through the MVS WTO and WTOR macros. Exit routines called from the user environment or functional subsystem environment can communicate with the operator through the MVS WTO and WTOR macros. Note that, if a message is to be associated with jobs processed by a functional subsystem, the job id must be included with the message. Exit 2, 3, and 4 allow you to send an exit-generated message to the operator along with certain return codes by setting a flag in the RXITFLAG byte.

Exit 5 allows you to control the standard $CRET macro “OK” message and to send your own exit-generated message text through the $CRET macro. Exit 9 allows you to control the standard output overflow message. Exit 10 allows you control over the text and routing of all $WTO messages. For details, see the individual exit descriptions.

Required mapping macros

Depending on the environment in which an exit executes, you will need to provide the appropriate set of mapping macros to map storage areas. Below, listed by environment, are the standard mapping macros required in order that your exit routine will assemble properly. The DSECTID for the mapping macro should be specified on the $MODULE macro. You should also note that individual exits also require other specific mapping macros. These are listed under the “DSECTIDs TO BE SPECIFIED ON $MODULE” heading provided for each exit.

Note: The addition of $MODULE in each exit will cause JES2 to pull in required mapping macros. However, all macros should be explicitly coded to prevent the return of MNOTEs and the possibility of assembly errors. Be certain your exit routines conform to JES2 coding conventions. This will allow easier diagnosis if an error should occur.

JES2 main task environment exits

0-5
7
10-11
13-22
24
26-27
38
39
40
44
46-47
49
51
Assuming you minimally code the following for each exit
COPY $HASPGBL
$MODULE
$ENTRY
$SAVE
$RETURN
$MODEND
END

Required macros
$CADDR (required by $MODULE)
$HASPEQU (required by $MODULE)
$HCT (required by $MODULE)
$MIT (required by $MODULE)
$PADDR (required by $MODULE)
$PARMLST (required by $MODULE)
$PSV (required by $MODULE)
$PCE (required by $MODULE)
$USERCBS (required by $MODULE)

JES2 subtask environment exits
6
8
12

Assuming you minimally code the following for each exit
COPY $HASPGBL
$MODULE
$ENTRY
$SAVE
$RETURN
$MODEND
END

Required macros
$CADDR (required by $MODULE)
$HASPEQU (required by $MODULE)
$HCT (required by $MODULE)
$MIT (required by $MODULE)
$PADDR (required by $MODULE)
$PARMLST (required by $MODULE)
$PSV (required by $MODULE)
$PCE (required by $MODULE)
$USERCBS (required by $MODULE)

Functional subsystem address space environment exits
23
25

Assuming you minimally code the following for each exit
COPY $HASPGBL
$MODULE
$ENTRY
$SAVE
$RETURN
$MODEND
END
Required macros
$CADDR (required by $MODULE)
ETD (required to support $HFCT)
FSIP (required to support $HFCT)
$HASPEQU (required by $MODULE)
$HFCT (required by $MODULE)
$MIT (required by $MODULE)
$PADDR (required by $MODULE)
$PARMLST (required by $MODULE)
$PSV (required by $MODULE)

User environment exits
8-9
12
28-37
41-43
45
48
50
52-57

Assuming you minimally code the following for each exit
COPY $HASPGBL
$MODULE
$ENTRY
$SAVE
$RETURN
$MODEND
END

Required macros
$CADDR (required by $MODULE)
$HASPEQU (required by $MODULE)
$HCCT (required by $MODULE)
$MIT (required by $MODULE)
$PADDR (required by $MODULE)
$PSV (required by $MODULE)
$USERCBS (required by $MODULE)

The following programming considerations describe some specific requirements for coding your exit routine:

- Naming and Identifying an Exit Routine
  You must begin each exit routine with the JES2 $ENTRY macro, which you use to name the routine and to identify it to JES2.
  For more information, see “Packaging Exit Routines” later in this chapter.
  Note that you have flexibility in naming your exit routines, under standard labeling conventions except for Exit 0 (see the description of Exit 0 in "Chapter 3 - IBM-defined exits" on page 59 for more detail).

- Exit Addressability
  The $ENTRY macro is also used to generate a USING statement for your exit routine. The BASE= operand is used to specify the register or registers which provide addressability when the exit routine gets control. However, the $ENTRY macro does not load the base register.

- Source Module Conventions
The construction of a source module must follow certain conventions depending on how you intend to package the exit routine. Through these conventions, JES2 is able to locate both exit routines and exit points within a module.

- **Security**
  When deciding on whether to implement a specific exit routine, you should consider whether installing a security product with your other system software could satisfy your requirements. You should also consider the effect an exit routine could have in terms of your installation’s security policy. Your security auditing may be inaccurate if you change security information in a control block in an exit that occurs after access to a resource has already been granted without additional validation. Similarly, changes made to security information by an exit that occurs before validation, could cause the validation to fail.

- **DBCS Assembly Option**
  DBCS (Double-byte Character Set) is an option that may be invoked when doing assemblies. DBCS is a means of providing support for languages which contain too many symbols to be represented by a single byte character set such as EBCDIC. JES2 supports the High-Level Assembler DBCS option for JES2 exit routines. All JES2 macros integral in a customer’s JES2 exit will abide by DBCS option rules, including the continuation line logic. JES2 macros will not have the same characters specified in both columns 71 and 72. This would be interpreted as a special DBCS continuation character. IBM does not support the DBCS option for reassembly of its modules.

### User environment exit considerations

#### Reentrancy

JES2 main task exits do not need to be reentrant because there is only one task running in the module at a time. However, multiple tasks can be running code in a user environment exit simultaneously. All user environment exits should be reentrant. The following are some reentrancy problems often overlooked in JES2 exits:

- Building messages directly in data constants in the local CSECT instead of using a work area.
- $$WTO$$ processing that sets the command character at the start of a message, even though the message does not have any replaceable text.
- Inline parameter lists used by MVS macros, such as ENQ and DEQ.
- Storing routine addresses into local (CSECT) storage areas.

#### Accessing CKPTed Data Area

If you are running code in one of the user environment exits, you might need to access data that is in the JES2 checkpoint data set. To facilitate this, JES2 maintains a “live checkpoint version” in the checkpoint version data space. This live version is an IARVSERV shared copy of the instorage checkpoint data set. It is updated by the main task as your exit is looking at the data. It is not advisable to run chains in the live version because the chains can be altered by the main task as you run them. However, if you know where a needed data area is located (a JQE or a JOE for example), and the data area is not going away (it is busy on your device), using a live version is a way to obtain the latest checkpoint data.

If you are in a user environment exit working with a NJE/TCP device (that is you are running in a NETSERV address space), the following code accesses an
IASDSERV data area that points to the live version (xx in xWNSST is SR, ST, JR, or JT for the appropriate device dependent area):

```
USING DSERV,R5  Est DSERV addressability
SPACE 1
L R5,xxWNSST  Get NSST address
LAE R5,0(R5)  Clear access register
L R5,NSSNSCT-NSST(R5)  Get NSCT address
L R5,NSCDSERV-NSCT(R5)  Get live DSERV addr
```

If you are not sure whether or not you are in a NETSERV address space, you can obtain an IAZDSERV for the live version using the $DSERV macro. For example:

```
$DSERV FUNC=GET, Get DSERV
LIVE=YES, Use "live" version
DSERV=(R2)  Save address in R2
```

Code using DSERV in R2:

```
$DSERV FUNC=FREE, Free DSERV
DSERV=(R2)  Address of DSERV to free
```

**Accessing $CATs**

Input processing exits might need to access a $CAT to get values for a job being received or being submitted. To access a $CAT, you need to get an IAZDSERV for a live version, and then obtain a $CAT from that live version. For example:

```
$DSERV FUNC=GET, Get DSERV
LIVE=YES, Use "live" version
DSERV=(R2)  Save address in R2
SPACE 1
$DOGCAT ACTION=(FETCH,READ), Get CAT for job class
JOBCLASS=JRDBLE,
DSERV=(R2)
LR R3,R1  Get CAT address
SPACE 1
USING CAT,R3  DECLARE CAT ADDRESSABILITY
: Process $CAT in R3
: $DOGCAT ACTION=RETURN,CAT=CAT Return CAT storage
SPACE 1
$DSERV FUNC=FREE, Free DSERV
DSERV=(R2)  Address of DSERV to free
SPACE 1
DROP R3  DROP CAT ADDRESSABILITY
```

If you are implementing code that will only be running in a NETSERV address space, you can replace the $DSERV calls with the code from the "Accessing CKPTed Data Area" example to obtain the IAZDSERV from the $NSCT.

---

**One time exit initialization code**

Some exits want to perform initialization code the first time they are called, for example loading a service module or building a table needed for processing. However, if this is a user environment exit, it is not running in the JES2 address space and is not main task serialized. Without some special serialization (such as an ENQ), it is possible that the code is actually being run simultaneously by two exit invocations. Also, if a data area is being obtained or a module is being loaded, it is possible that the storage is freed when the current address space terminates.

It is easiest to place any one time initialization logic in the post initialization exit 24. If data addresses need to be passed to other exits, either a $CUCT (an area
pointed to by CCTCUCT in the $HCCT) can be used for a data address or a $UCADDR (an area pointed to by CCTUCADD and used by $CALL) can be used for a routine address. Another option is to use a named token. $TOKENSR provides a JES2 interface to the MVS Name/Token service. You can use tokens to store data that is needed at some later point in processing.

**Tracing**

Minimal tracing of exit invocation can be performed automatically as part of the exit facility. For this tracing to occur, three conditions are necessary:

1. The trace ID for exit tracing (ID 13) must be enabled.
2. The EXIT(nnn) initialization statement or the $T EXIT(nnn) operator command must have enabled tracing. For more information, see “Tracing status” on page 53.
3. Tracing must be active (TRACEDEF ACTIVE=YES).

This automatic tracing produces a limited trace entry containing such general information as exit point identification, register contents at the time of exit invocation, and the contents of the $XPL (if part of the $EXIT interface).

Also, to further trace execution of exit routine code, issue the standard JES2 $TRACE macro call within an exit routine. This results in a full trace record of exit routine processing.

It is recommended that you use tracing to its fullest extent only in your testing cycle, and that you limit its use in those areas of the standard processing environment—for example, in conversion processing—where it is most likely to degrade system performance.

**Recovery**

An exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of an exit routine and, therefore, any standard JES2 recovery that happens to be in effect when your exit routine is called is, at best, minimal for your particular needs. In other areas of processing, no JES2 recovery environment is in effect, and an exit routine error has the potential to cause JES2 to fail. Consequently, you should provide your own recovery mechanisms within your exit routines.

For all exits routines for which you provide an $ESTAE routine, also be certain to add the error recovery area DSECT, $ERA, to the $MODULE macro. On entry into the recovery routine set up by $ESTAE, register 1 points to the ERA.

You can use the standard JES2 $ESTAE recovery mechanisms in implementing your own recovery within the JES2 main task. You can use the MVS ESTAE recovery mechanism in implementing your own recovery in the SUBTASK, USER, or FSS environments. When recovering in the SUBTASK environment, JES2 frees the save areas associated with the abending subtask. Your recovery should not depend on the presence of a particular save area.

At minimum, a recovery mechanism should place a 0 or 4 return code in register 15. Beyond this, recovery depends on the particular purpose of an exit routine.
Loading non-JES2 modules

The $MODLOAD service of JES2 allows for the directed load of modules. It loads all the modules that JES2 needs for processing. Directed loading allows for modules to be placed in requestor obtained storage. Modules loaded using the directed load service do not get the normal contents directory entries (CDE) and thus cannot be found by other LOADs. However, this implies that these modules are not deleted as part of task or address space termination unless the storage they were loaded into is freed.

With logic moving into common storage, non-JES2 modules might need to be available to JES2 code (and exit code) in common storage. The JES2 $MODLOAD service supports directed loading non-JES2 modules. This includes placing non-JES2 modules in common storage. An exit can load a necessary module into common storage during exit 24 (post initialization) processing, and then use it as needed. JES2 then deletes the module during JES2 shutdown ($PJES2) processing when it deletes the other JES2 common storage modules.

Non-JES2 modules can be loaded dynamically after initialization completes. See "Dynamic Load Modules" on page 29 for more information.
Controlling the loading of installation-defined load modules

Loading and placement of installation load modules

Use the LOADmod(jxxxxxxx) initialization statement or the $ADD LOADmod(jxxxxxxx) command to direct the loading of all installation-defined load modules (such as user-defined exits). Exit routines must be loaded in this manner, rather than linking to JES2 load modules. **JES2 only searches for installation-defined exit routines in user modules defined by the LOADmod(jxxxxxxx) initialization statement or the $ADD LOADmod(jxxxxxxx) command, in the reserved module names HASPXJ00 – J31, or in HASPXIT0; JES2 does not search for such routines in IBM-defined modules.** The STORAGE= parameter specifies the area of storage where the load module is to be loaded. This is the copy that JES2 will use. Table 2 presents a summary of the manner in which JES2 directs the load of a load module based on initial placement of that load module and the LOADmod(jxxxxxxx) STORAGE= specification.

Note the following restrictions:

- STORAGE=LPA is invalid if the load module is initially placed in STEPLIB only, LINKLIST only, or both STEPLIB and LINKLIST. JES2 issues message $HASP003 RC(31), MODULE COULD NOT BE LOADED.
- All other STORAGE= requests are valid, but you may not receive the expected result (see Table 2).
- You cannot load a module into the link pack area (LPA) following MVS initialization. You may only request that the copy of the module in LPA be used if multiple copies are found.

**Table 2. Directed Load and Use of Modules Based on LOADMOD(jxxxxxxx) STORAGE= Specification**

<table>
<thead>
<tr>
<th>Location of Module is:</th>
<th>STORAGE=PVT, module is found in</th>
<th>STORAGE=CSA, module is found in</th>
<th>STORAGE=LPA, module is found in</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB Only</td>
<td>PVT</td>
<td>CSA</td>
<td>$HASP003 RC=31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPA Only</td>
<td>LPA</td>
<td>LPA</td>
<td>LPA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LNKLST Only</td>
<td>PVT</td>
<td>CSA</td>
<td>$HASP003 RC=31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEPLIB and LPA (STEPLIB)</td>
<td>PVT</td>
<td>CSA</td>
<td>LPA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEPLIB and LNKLST</td>
<td>PVT</td>
<td>CSA</td>
<td>$HASP003 RC=31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPA and LNKLST</td>
<td>LPA</td>
<td>LPA</td>
<td>LPA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STEPLIB, LPA and LNKLST</td>
<td>PVT</td>
<td>CSA</td>
<td>LPA</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1988, 2009
To place the load module either above or below 16 megabytes, use the linkage editor MODE statement or specify the RMODE= parameter on the $MODULE macro.

Figure 5 illustrates two ways to package an exit routine:

1. As a totally separate load module
2. As part of HASPXITO

Figure 5. Methods of Packaging an Exit Routine

A JES2 $MODULE macro must be the first code-generating statement (immediately preceded by COPY $HASPGBL) in a source module to be assembled and either link edited separately and loaded at initialization or a source module to be added to a standard JES2 load module.

Note: The $MODULE macro call must occur prior to the first use of $ENTRY or $EXIT, and a JES2 $MODEND macro must be coded at the end of both types of source modules.

You can only code one $MODULE and one $MODEND macro in each source module. Further, when link editing exits into their own load modules (other than HASJES20), each source module must be linked into its own load module.

To locate the MITs of modules that are added to the standard JES2 load modules, JES2 uses weak external address constants. To locate the MITs of modules that are linked in their own load modules, JES2 assumes that the MIT, generated by $MODULE, is located at the front of the load module to which it points. The MITETBL, generated by $MODULE, is located at the end of a module loaded at initialization.
**Note:** For all exit routine source modules, that if an exit routine references the symbolic name of a control block field, the mapping macro for that control block must be included in the $MODULE macro list in the same source module as the exit routine at assembly time.

Furthermore, see Appendix C, “Hints for Coding JES2 Exit Routines” for a list of required mapping macros for individual exits. These macros are environment dependent and must be coded to prevent assembly errors and error messages.

The ENVIRON= operand of the $MODULE macro should be used to specify which JES2 operating environment the exit routine(s) is to execute. Each exit description in the “IBM-Defined Exits” reference section in “Chapter 3 - IBM-defined exits” on page 59 includes a list of mapping macros normally required at assembly.

**Dynamic Load Modules**

Dynamic load modules provide the following functions:

- Load, refresh, and delete installation load modules, which are not part of the IBM base JES2 code, after JES2 initialization processing. The dynamic table pairs and exit routine addresses are updated as needed. The load modules provide load and delete routines to perform any processing that might be needed to adjust data pointers JES2 does not process.

**Note:** This function does not support base JES2 modules, so it can NOT be used to apply IBM service.

- Alter the list of routines associated with an exit point through operator command.

When building a load module with exit routines and dynamic tables, you must decide whether you want to support dynamically loading or deleting these modules after initialization. This is especially useful in a test environment where new versions of a failing exit can be activated without a restart. Depending on the processing done in your exits, supporting dynamic loading and deleting might require no additional code or only a reorganization of your existing logic.

Making load modules dynamically loadable will increase the amount of testing you need to do. You need to not only test the function implemented by your modules, but also ensure that everything works after the module is dynamically loaded, refreshed or deleted. The advantage of dynamic load modules is that when you find a problem in your module, you can correct the problem and get a new version of your code running without major disruptions. If the problem is bad enough, you can delete the module, fix it, and load the fixed version.

If the code is tested and placed in a production environment, IBM suggests that you do not make loading, refreshing, or deleting load modules as a part of your normal operations. This is because it is not always possible to delete old modules from storage. JES2 only deletes a module from storage if it will surely not be used. Some of these old modules will take up space until JES2 or z/OS is restarted (depending on where the module is loaded). Loading, refreshing, and deleting load modules in a production environment must be reserved for emergency situations (if it would save or delay an IPL) or for modules that IBM service has provided to collect additional diagnostic information. If a production environment needs to be altered on a regular basis, it would be better to alter the list of routines associated with an exit point rather than altering what modules are loaded.
Dynamic Load Module Considerations

When writing new load module or updating an existing module to support dynamic processes, you need to consider the following things:

- The data areas that the modules access. You need to consider the following questions:
  - Does the module accesses a data area that has been created by the installation?
  - How are these data areas created?
  - What points to the data area?
  - Is the area dynamically obtained or is it an area within the load module?

If the exits and tables only access JES2 and z/OS data areas, this is not a concern. Also, if the data area is contained within the module and there are no external pointers to the data area, then that also is not a concern. However, if the data area is installation specific and the address is obtained using a pointer external to the load module (such as the $UCT pointer in the $HCT), then you need to consider:
  - How is the data area set up? If it is only used by this module, then is a $$$$LOAD routine needed to initialize it?

  **Note:** See "$$$$LOAD Routine" on page 31 for a description of the $$$$LOAD routine.

  - Does the code deal with the case where the data area already exists (or does it create a second data area)?
  - Is the data area in common storage?
  - Does it need to be deleted when this module is deleted or when JES2 terminates? Is a $$$$DEL routine needed to free the data area?

  **Note:** See "$$$$DEL Routine" on page 34 for a description of the $$$$DEL routine.

  - Does anything special need to be done if the module is refreshed instead of being deleted?
  - If the data area is in the load module, are there pointers that need to be cleared if the module is deleted or refreshed?
  - If the data area is managed by an exit 24 (JES2 initialization) and exit 26 (JES2 termination) pair, should that processing be moved to a $$$$LOAD and a $$$$DEL routine?

In general, $$$$LOAD and $$$$DEL routines can solve most data area problems to ensure the proper flexibility to alter the data area as needed.

- The creation of installation PCEs (subdispatchable units in the JES2 address space) or DTEs (subtasks in the JES2 address space). If the PCEs or DTEs are defined using dynamic tables or traditional table pairs, the appropriate PCE or subtask is started as part of normal JES2 initialization. However, with dynamic load modules, the installation code decides attaching and detaching the PCEs or DTEs as needed. In general, the simplest way to deal with PCEs and DTEs is to use the appropriate $PCEDYN or $DITEDYN macro to detach the old (existing) PCEs or DTEs in the $$$$DEL routine and reattach them in the $$$$LOAD routine. To ensure that the PCEs can be attached after initialization, be sure to code the DYNAMIC=YES keyword on the $PCETAB macro that defines the PCE.

You also need to consider some other things when creating dynamic load modules:
• If you are converting an existing exit to be dynamic, is there logic in exit 24 (post initialization) that should be moved to a $$ $$LOAD routine?
• If you are converting an existing exit to be dynamic, is there logic in exit 26 (JES2 termination) that should be moved to a $$ $$DEL routine?
• Does the installation module include code that front ends or replaces a JES2 service? Front ending is the process of replacing the address of a JES2 service in the $CADDR, $PADDR, $HCT or other data area, with the address of a routine in the module and then calling the JES2 service only after the installation routine runs. If so, care must be taken to ensure that the routine addresses are updated if the installation load module is refreshed or deleted. This is especially true at JES2 termination processing since some are called after installation load modules are deleted at JES2 termination.

Note: IBM recommends that you do not front end IBM services. Designing a function that requires front ending IBM services could limit your ability to exploit future IBM functionality to refresh IBM services dynamically.

• Traditional (non-dynamic) tables that are set in the $MCT data area (or other table pairs) must be updated as modules are loaded or deleted. In general, use of non-dynamic tables can be converted to dynamic tables (which JES2 will automatically process). Otherwise, code can be added to the $$ $$LOAD and $$ $$DEL routines to handle updating these pointers.

If your load module cannot support dynamic processes, there are a number of options to prevent unintended processing:
• Setting DYNAMIC=NO on the $MODULE statement of the load module will prevent all dynamic processing for this load module. Initialization processing is not affected. Any $$ $$LOAD or $$ $$DEL routines in the module will be called out of JES2 initialization and termination processing.
• From a $$ $$LOAD routine, set the LMT2NDYN bit in flag byte LMTFLG2. The LMT of the module being loaded is passed to the $$ $$LOAD routine in the $CSVPARM data area. If done during initialization, this has the same effect as setting DYNAMIC=NO on the $MODULE. However, if the module was not loaded during initialization, using this technique allows the module to be loaded after initialization but not deleted or refreshed later.
• If you can support dynamic processes but there are tables or routines in your module that cannot be deleted, then you can set a return code 8 from a $$ $$DEL routine. This prevents the module from being physically deleted. You should be careful not to set it for every call to the $$ $$DEL routine since if the module is refreshed multiple times, you only need to keep the first copy of the load module in storage. The $$ $$DEL processing should determine if the specific copy of the module that being deleted is the one that needs to remain.

$$ $$LOAD and $$ $$DEL are reserved routine names on the EXIT ROUTINES=xxxxx initialization statement and the $T EXIT,ROUTINES=xxxx command. The two reserved routines process when a module is loaded at initialization or is logically deleted at normal JES2 termination.

$$ $$LOAD Routine

When a load module is loaded by the LOADMOD initialization statement, the $ADD LOADMOD command, or the $T LOADMOD,REFRESH command, JES2 searches the load module for a $ENTRY macro with the name $$ $$LOAD. If the module is found, JES2 calls it after all dynamic tables are linked in.

Controlling the loading of installation-defined load modules
If the load module is loaded by the $T LOADMOD,REFRESH command, JES2 processes the following steps:

1. Load new copy of module into storage and verify it is valid.
2. Call the $$$$DEL routine for the old module.
3. Replace any exit routine addresses that point into the old module with corresponding addresses in the new load module. If no corresponding routine is found in the new module, the routine address is nullified (the routine is not called).
4. Replace dynamic tables that point into the old module with corresponding tables in the new module.
5. Delete any dynamic tables that still point to the old module.
6. Connect any dynamic tables in the new module that have not been connected yet.
7. Call the $$$$LOAD routine for the new module.
8. Attempt to delete the old module from storage.

**Note:** The $T LOADMOD,REFRESH command can be issued for an LPA module that is not altered. The new and the old modules are at the same address with two LMTs representing the two modules correspondingly. In this case, the $$$$LOAD and $$$$DEL routines are called.

**Environment:** $$$$LOAD is called in the JES2 main task limited environment (JES2 initialization) and the JES2 main task environment.

**Recovery:** $ESTAE recovery is in effect. However, the $$$$LOAD routine should not depend on JES2 for recovery. You should provide your own recovery within your $$$$LOAD routine.

**Point of processing:** After module has been loaded but before control is returned to the requestor of the load.

**Register contents when $$$$LOAD gets control:**

R0
Not applicable

R1
Address of a parameter list mapped by $CSVPARM

R2-R10
Not applicable

R11
Address of the HCT

R12
Not applicable

R13
Address of current PCE (may be initialization PCE)

R14
Return address

R15
Entry address

$CSVPARM (pointed to by register 1 on entry) contains the following bits:
CSVPID
   Eye catcher (‘CSVP’)

CSVPSIZE
   Size of parameter list

CSVPVER
   Current version of base section (1)

CSVPTYPE
   Routine identifier

CSVLOAD
   Indicates $$$$LOAD routine

CSVPLMT
   Address of LMT being loaded

CSVPMIT
   Address of module/MIT being loaded

CSVPLCMD
   Reason for load:
   CSVPLCJS
      JES2 performing load
   CSVPLCIN
      LOADMOD init statement
   CSVPLCAL
      $ADD LOADMOD command
   CSVPLCRC
      $T LOADMOD,REFRESH command

CSVPLLOC
   Where the module was loaded:
   CSVPLPVT
      Loaded to JES2 private
   CSVPLCSA
      Loaded to common storage
   CSVPLLPA
      Loaded to LPA

CSVPLOLD
   Address of LMT being replaced (for the $T LOAD,REFRESH command)

CSVPLSDE
   Address of an additional $$$$DEL routine (see LPA processing below). This routine gets control before a $$$$DEL routine in the module is processed.

Register contents when $$$$LOAD passes control back to JES2:

R0-R1
   Not applicable (ignored)

R2-R13
   Not applicable (unchanged)

R14
   Not applicable (ignored)
R15
   Zero (CSVPLROK)

JES2 does not recognize any return codes from this routine. However, IBM suggests setting R15 to zero to indicate successful processing in case future development adds a return code to this routine.

$$$$DEL Routine

When a load module is deleted because of the $DEL LOADMOD command, the $T LOADMOD,REFRESH command, or a second LOADMOD initialization statement for the same module, JES2 searches the load module for a $ENTRY macro with the name $$$$DEL. If the module is found, JES2 calls it as the first step in the delete processing for the module.

If the load module is deleted by the $T LOADMOD,REFRESH command, JES2 processes the following steps:
1. Load new copy of module into storage and verify it is valid.
2. Call the $$$$DEL routine for the old module.
3. Replace any exit routine addresses that point into the old module with corresponding addresses in the new load module. If no corresponding routine is found in the new module, the routine address is nullified (the routine is not called).
4. Replace dynamic tables that point into the old module with corresponding tables in the new module.
5. Delete any dynamic tables that still point to the old module.
6. Connect any dynamic tables in the new module that have not been connected yet.
7. Call the $$$$LOAD routine for the new module.
8. Attempt to delete the old module from storage.

Note: The $T LOADMOD,REFRESH command can be issued for an LPA module that is not altered. The new and the old modules are at the same address with two LMTs representing the two modules correspondingly. In this case, the $$$$LOAD and $$$$DEL routines are called.

Environment: $$$$DEL is called in the JES2 main task limited environment (JES2 initialization) and the JES2 main task environment.

Recovery: $ESTAE recovery is in effect. However, the $$$$DEL routine should not depend on JES2 for recovery. You should provide your own recovery within your $$$$DEL routine.

Point of processing: As the first step in the processes of deleting a module, before any tables have been unplugged or routine addresses cleared.

Register contents when $$$$DEL gets control:

R0
   Not applicable

R1
   Address of a parameter list mapped by $CSVParm

R2-R10
   Not applicable
R11
Address of the HCT

R12
Not applicable

R13
Address of current PCE (may be initialization PCE)

R14
Return address

R15
Entry address

$CSVPARM (pointed to by register 1 on entry) contains the following bits:

CSVPID
Eye catcher (‘CSVP’)

CSVPSIZE
Size of parameter list

CSVPER
Current version of base section (1)

CSVPTYPE
Routine identifier

CSVDEL
Indicates $$$$$DEL routine

CSVPLMT
Address of LMT being deleted

CSVMIT
Address of module/MIT being deleted

CSVPLCND
Reason for delete:

CSVDCJS
JES2 performing delete

CSVDCIN
LOADMOD init statement

CSVDCDL
$DEL LOADMOD command

CSVDCRL
$T LOADMOD,REFRESH command

CSVDCRS
$PJES2 processing

CSVDCSC
Secondary call

CSVPDIND
Call flags:

CSVDSND
Second call after a RC 4/8
CSVPDFRC
Module being force deleted

CSVPDFRE
Storage for module has been freed

CSVPDNEW
Address of LMT for new module that was loaded (for the $T LOAD,REFRESH command)

Register contents when $$$$DEL passes control back to JES2:

R0-R1
Not applicable (ignored)

R2-R13
Not applicable (unchanged)

R14
Not applicable (ignored)

R15
Return code (ignored if this is a force delete)

Return code processing: Return codes from the $$ $$ $$ $$DEL routine are ignored if the module is being force deleted (CSVPDFRC bit on). Otherwise the following processing occurs based on the return code:

CSVPDROK (0)
Continue deletion normally. This routine will not be called again.

CSVPDRNN (4)
Do not delete the module now. JES2 will delete dynamic tables and exit routines without freeing the storage. $$ $$ $$ $$DEL will be called again if all users of the module are gone (with CSVPDSND set). If the second call give a return code 4, $$ $$ $$ $$DEL will be called again at about a five minute interval. However, if needed, JES2 can make a force delete call prior to the timer expiring.

CSVPDRDND (8)
Process the same as RC=4 except that JES2 will not call the $$ $$ $$ $$DEL routine again except for the following two cases:

• A force delete of the module is required because of a JES2 termination or an LPA deletion.
• A JES2 hot start and the load module is in CSA or LPA. In this case, any processing for this module on a hot start is allowed though this is a call to the $$ $$ $$ $$DEL routine. Normal return code processing occurs.

Special Considerations for LPA Modules
Special considerations need to be given to installation load modules placed in LPA. These modules are not actually loaded, deleted or refreshed by JES2. Instead they are managed by MVS using dynamic LPA services and commands.

When JES2 loads a module in LPA, it simply locates the address of the module with a specified name in LPA. If this loading is caused by a $T LOADMOD,REFRESH command, the LPA module might not be changed and JES2 will reset all its pointers. Therefore, there will be two LMTs, one representing the module being deleted and one representing the same module being loaded. The appropriate $$ $$ $$ $$DEL and $$ $$ $$ $$LOAD routines are called. Special logic might be needed in these routines to properly handle the fact that the new and old modules
are at the same address. In particular, if there is a code in the $$$$DEL routine that examines pointers to see if they point into the module being deleted, then in this case, there will be pointers into the old module. However, these pointers are not residual and need to be maintained.

Another consideration with dynamic LPA is the ability for a module to be deleted out from under JES2 using the MVS dynamic LPA commands. It is not expected that this would happen under normal circumstances but JES2 attempts to deal with this situation, should it arise. JES2 is notified after a module has been physically deleted from storage. It marks the LMT to indicate the module has been freed and schedules the module for logical deletion (removal of pointers to the deleted module). Normally logical deletion occurs first but in this case JES2 has no control over the physical deletion. As part of logical deletion JES2 will attempt to call a $$$$DEL routine. Unfortunately, since the module is no longer in storage, the module cannot be searched for a normal $$$$DEL routine. However, at the time a module is loaded, the $$$$LOAD routine has the ability to specify the address of an additional $$$$DEL routine in the $CSVPARM data area (field CSVPL$DR). This routine cannot be in the module since it is intended for the case when module has been deleted. Instead, it should be in code the $$$$LOAD routine has obtained and copied a routine into. It is expected that this routine would set some indicator that the function implemented by this routine is no longer active. Or issue a message that things are no longer functioning.
Enabling an exit

Figure 6 shows how an exit routine (HASPUEX) can be assembled and link-edited, and how to use the load module name. The source is in SYS1.JESEXITS, and the load module is linked into SYS1.SHASLNKE with the name of HASPUEX. This name must also appear on the LOADmod(jxxxxxxx) initialization statement.

The following JES2 initialization statements can be used to load and associate Exit 1 with the above routine. **Note that the name on the LOADmod(jxxxxxxx) statement must match the load module specified to the linkage editor, and the name on the ROUTINE= parameter on the EXIT(nnn) statement must be the same name as on the $ENTRY macro.**

```
LOADMOD(HASPUEX) STORAGE=PVT
EXIT(1) ROUTINE=UEXIT1,STATUS=ENABLED,TRACE=NO
```

Figure 7 shows an example exit routine for a user defined exit (UEXIT1). The source is in SYS9.TECH, and the load module is linked into SYS9.TECH.LINKLIB with the name of UEXIT1. This name must also appear on the LOADmod(jxxxxxxx) initialization statement.
Figure 7. Example of an Exit Routine Employing a User Defined Exit
Getting listings of JES2 data areas

When writing and debugging an installation exit, it is sometimes useful to get listings of JES2 data areas similar to what is available in the z/OS data areas books. There are a number of ways to do this depending on what data you need.

To get a listing of all the JES2 data areas, you can assemble the module HASPDOC; the JES2 source code distribution library SYS1.SHASSRC provides this module. You can assemble this module by using either SMP/E, the sample JES2 assembly PROC HASIASM in SYS1.SHASSAMP, or using your own assembly procedure. The output listing contains all the JES2 data areas. If you request the assembler produce a full cross reference using the XREF(FULL) parameter, you will get an alphabetic listing of all the symbols.

You can also use the same source module to get a listing of the z/OS data areas that JES2 uses. To do this, include the assembler parameter SYSPARM((,,GEN,GEN)) on the assembly. You can find the operands of SYSPARM for any JES2 module in [z/OS JES2 Macros] under the SYSP= operand of the $MODULE macro.

If you need a listing of just one data area (either JES2 or z/OS), you can create an assembler module with only a $MODULE statement listing the data areas you want listings for and an END statement. The following is an example of an assembler module that creates a listing of the JES2 $HCT data area. The assembler listing produced will have only the $MODULE expansion and the $HCT data area:

```
$MODULE ($HCT,GEN)
END
```

This method works for any mapping macro supported by $MODULE. All required macros for the assembly are automatically included and only the requested data area is generated in the listing. You can get more than one data area by just adding it to the $MODULE list:

```
$MODULE ($HCT,GEN),($PCE,GEN)
END
```

This gets the $HCT and the $PCE data areas.

You can also add the GEN operand to data area specifications in the $MODULEs in your exits. This puts any requested data areas on to the listing for your exits.

If there is no label on the $MODULE and the only operands specified are the data areas to generate, $MODULE will not generate the JES2 $MIT data structure. If you do place a label on the $MODULE invocation or add any other operands, $MODULE will attempt to build a JES2 load module. Without other structures, it might get assembly errors. Using a $MODULE without operands or a label can be useful when you need to include JES2 mapping macros in code that is not going to be run as a JES2 exit.
Sample exit routines

For most exits, IBM provides sample exit routines in SYS1.SHASSAMP. The documentation for each exit indicates whether a sample routine has been provided.
Multiple exit routines in a single module

When developing and testing installation exits, it is probably easier to keep each exit routine in its own source and load module. In this manner, the routines can be assembled, loaded, and tested independently. If there are many routines, you may want to eventually combine them into a single source and load module for easier maintenance procedures.

Figure 8 shows three exit routines in a single module with a general structure that you may want to follow.

```assembly
XIT1 RTN1
ENTRY BASE=R12
EXIT ROUTINE ENTRY POINT
$SAVE
LR R12,R15 LOAD BASE REGISTER
LA R15,8 SET RETURN CODE
RETURN1 $RETURN RC=(R15) RETURN TO HASPPRPU
TITLE 'SAMPLE SEPARATOR PAGE EXIT - ROUTINE 2'
XIT1 RTN2
ENTRY BASE=R12 EXIT ROUTINE ENTRY POINT
$SAVE
LR R12,R15 LOAD BASE REGISTER
```

Figure 8. Example of Providing Multiple Exits within a Single Load Module (Part 1 of 2)
The following JES2 initialization statements can be used to load and associate exit points 1 and 2 with the above routines.

```
LOADMOD(HASPUEx) STORAGE=PVT
EXIT(1) ROUTINE=(XIT1RTN1,XIT1RTN2),STATUS=ENABLED,TRACE=NO
EXIT(2) ROUTINE=XIT2RTN1,STATUS=ENABLED,TRACE=NO
```

Figure 8. Example of Providing Multiple Exits within a Single Load Module (Part 2 of 2)
Testing your exit routine

To test your exit routine you need to integrate your exit routine in the system, ensure that it gets control and executes, and verify that the functions it is intended to perform are performed. Verifying that the exit routine performed its function is exit routine-dependent and unique for each exit routine.

You should test and debug your exit routine by running it on a secondary JES2 first. In this way, any errors that occur do not directly affect your main JES2 production system. When the errors in the exit routine are fixed and tested, you can then integrate it into the production JES2 system. Note that the following restrictions apply to JES2 functions when using a secondary JES2:

- Started tasks (STCs) can be directed to either a primary or secondary JES. However, following an IPL, started tasks do not complete start processing until the primary subsystem has been started and completed initialization.
- Time-sharing users (TSUs) may only interface with the primary JES2.
- The MVS I/O attention table can only be associated with the primary JES. Therefore, secondary JESs cannot receive the “unsolicited interrupt” required to support pause-mode for print and punch devices and “hot readers” (that is, readers started through the physical start button without the $S RDRn JES2 command).
- The MVS log console (SYSLOG) can only be associated with the primary JES.
- Secondary subsystems are started individually rather than automatically during IPL by a start command in the master scheduler JCL (MSTJCL) as is the primary subsystem.

Dynamic loading of modules can simplify the testing of exit routines. JES2 commands allow you to incorporate a new version of your exit routine without the need for an IPL (for user or FSS environment exits), or a restart of JES2 (for JES2 main task or subtask exits). Installation modules can be dynamically loaded, deleted, and refreshed using the $ADD LOADmod(jxxxxxxx), $DEL LOADmod(jxxxxxxx), and $T LOADmod(jxxxxxxx),REFRESH commands. The list of routines associated with a JES2 exit can be dynamically changed with the $T EXIT(nnn),ROUtines= or the $T EXIT(nnn),REFRESH command. See "Dynamic Load Modules" on page 29 for more detailed information about the dynamic loading of modules. See z/OS JES2 Commands for more information about the commands mentioned above.

Packaging the exit

Exit routines need to be packaged into load modules before they can be loaded into the system and tested.

Modules that contain exit routines which execute in the JES2 main task or subtask environment can be linkedited into a load module; these exits should be loaded into private storage. Modules that contain exits in the user or functional subsystem environment can be linkedited together and must be in either LPA or CSA; these exits must be loaded into common storage. Do not linkedit multiple exit points that must be loaded into different areas of storage into the same load module.

You can also link edit your exit routines with HASJES20. When you package your exit routines in this manner, it is required that you use a collection of weak external names for the module names. These names should be the same as the label used
on the $MODULE macro of your exit routine. For HASJES20 the “weak external names” are as follows: HASPXJ00, HASPXJ01, ..., HASPXJ31.

You may choose to use one of these packaging techniques exclusively, or you may choose to use both methods in combination, assembling and link editing some routines into the standard JES2 load modules and assembling and link editing others separately and then loading them at initialization. Creating separate load modules for your exit routines is recommended. JES2 never makes unconditional direct references to external addresses or entry points in installation-written code. The association between exit routines and JES2 source code is resolved during initialization, or when processing JES2 commands that dynamically change the installation exit environment (for example, $T EXIT(nnn)).

Figure 9 illustrates a separately linked loaded module for an exit routine and the MIT and MITETBL structure associated with it. JES2 initialization uses this load module and the information in the MIT and MITETBL to initialize the exit routine in the system. The next topic describes this initialization process.

**Figure 9. Exit Routines Load Module**

**Initializing the exit in the system**

Initializing an exit and its exit routines involve the use of the following JES2 initialization statements or JES2 commands:

- **LOADMOD(jxxxxxxx) or $ADD LOADmod(jxxxxxx)**
  Use the LOADMOD(jxxxxxxx) initialization statement or the $ADD LOADmod(jxxxxxxx) command to load the modules containing your exit routines. The subscript of the LOADMOD initialization statement or the $ADD LOADmod(jxxxxxxx) command specifies the name of the module to be loaded as defined on the NAME control statement for the linkage editor. The module must be named according to MVS naming conventions. Exit routines to be called from the user or FSS environment can be loaded into CSA or you can request the LPA version be used by specifying the STORAGE=LPA | CSA parameter specification on the LOADMOD(jxxxxxxx) initialization statement or the $ADD
LOADmod(jxxxxxx) command. Exit routines to be called from the JES2 main task and subtask environments should be loaded in the private area of the JES2 address space. To place the load module either above or below 16 megabytes, use the linkage editor MODE statement or specify the RMODE= parameter on the $MODULE macro.

- **$DEL LOADmod(jxxxxxx) or $T LOADmod(jxxxxxx),REFRESH**
  Use the $DEL LOADmod(jxxxxxx) or the $T LOADmod(jxxxxxx),REFRESH command to delete or refresh the modules that contain your exit routines. The subscript of the commands specifies the name of the module that was previously loaded by a $ADD LOADmod(jxxxxxx) command, or a LOADMOD(jxxxxxxx) initialization statement.

- **EXIT(nnn) or $T EXIT(nnn),ROUtines=(xxxxxxxx) command**
  Use the EXIT(nnn) initialization statement or the $T EXIT(nnn),ROUtines=(xxxxxxxx) command to associate one or more exit routines with an exit.

Replace *nnn*, the exit number, with the corresponding exit identification number specified on the $EXIT macro or macros that define the exit point or points that establish the exit. The ROUTINES= parameter can then specify 1 to 255 exit routine names, as specified on the $ENTRY macro symbol field or macros that identify the corresponding exit routines. For example, you can specify EXIT(123) ROUTINES=(rtn1, rtn2, rtn3). The JES2 exit effector calls multiple exit routines in the sequence of their specification on the EXIT(nnn) statement. If you specify more than one EXIT(nnn) statement with the same identification number, JES2 honors the last statement it encounters during initialization. This specification can be changed post-initialization with the $T EXIT(nnn),ROUtines=(xxxxxxxx) command. This command not only allows the complete replacement of the list of routines associated with an exit, but also allows routines to be added to or removed from the existing list. See [z/OS JES2 Commands](https://www.ibm.com/support/knowledgecenter/SSEKG2_1.11.0/com.ibm.jes.doc/czooooex.html) for more information about changing the list of routines associated with an exit.

**Note**: The LOADMOD(jxxxxxx) and EXIT(nnn) initialization statements are not positional and do not have to be specified in any required order.

JES2 associates an exit with a routine in the module that was most recently loaded (by either a LOADMOD(jxxxxxxx) initialization statement or a $ADD LOADmod(jxxxxxxx) command).

**Note**: A $ADD LOADmod(jxxxxxxx) command does not automatically update the exits which refer to routines in the newly loaded module. The exits must be refreshed (by a $T EXIT(nnn),REFRESH command) or changed (by a $T EXIT(nnn),ROUtines= command) to use those routines.

However, a refresh is not needed to update dynamic tables. Dynamic tables are automatically added, deleted, or refreshed when the applicable JES2 command is issued. In addition, a refresh is not needed to update exits that refer to routines in a deleted or refreshed module. When an exit is associated with a routine that resides in a deleted module, even if the module resides in LPA, the routine will no longer be invoked for the exit (routine address of the exit is nullified). When an exit is associated with a routine that resides in a refreshed module, if the routine exists in the newly loaded module, the routine in the newly loaded module will be invoked for the exit; if the routine is absent in the newly loaded module, the routine will no longer be invoked for the exit.

In all cases, a $T EXIT(nnn),REFRESH command refreshes those exits so that they will invoke routines in the most recently loaded module.
Figure 10 illustrates the primary parts of JES2 and their location in storage when initialization completes.

A  User environment
B  User environment
C  JES2 main task and subtasks

Figure 10. Exit Placement
Passing control to exit routines

Every exit has a status of enabled or disabled. If an exit is enabled, JES2 calls its associated exit routine(s) whenever one of the exit’s exit points is encountered in processing JES2 code. (Note: The TYPE=TEST form of the $EXIT macro is an exception; a TEST-type exit point occurs before a TYPE=ENTER exit point to allow JES2 to determine whether the exit is implemented and enabled. If the exit is not both implemented and enabled, JES2 saves processing time by bypassing the call to the exit effector when it encounters the ENTER-type exit point.) When an exit is disabled, its exit points are transparent during JES2 processing and JES2 does not call the exit’s associated exit routine(s).

An exit’s status is first set at initialization. You can specify either STATUS=ENABLED or STATUS=DISABLED on the EXIT(nnn) initialization statement. If you leave the status of the exit unspecified, STATUS=ENABLED is the default.

An exit’s status can then be dynamically controlled by the operator, using the $T EXIT(nnn) command. Again, the operator has the option of identifying any exit by number, a range of exits, or all exits, and specifying either STATUS=ENABLED or STATUS=DISABLED. The operator can display an exit’s status by identifying the exit by number on the $D EXIT(nnn) command.

When you suspect that an exit routine associated with a particular exit is causing an error, a simple way of isolating the problem is to disable the exit, through an operator command ($T EXIT(nnn)), to determine if the error still occurs when the exit routine is not allowed to execute. You can also enable tracing as a debugging aid.

An exit can also be dynamically controlled on a job-related basis, using the exit facility.

Job-related exits

Certain exits are identified as job-related exits. For these exits, the JOBMASK parameter is specified on the $EXIT macro or macros defining their exit point or points. JOBMASK is specified with the address of the job exit mask, a 256-bit mask in the job control table (JCT), of which each bit corresponds to an exit identification number; bit 0 corresponds to Exit 0, bit 1 corresponds to Exit 1, bit 2 to Exit 2, and so on. (This means, of course, that bit 2 corresponding to Exit 2 is really the third bit in the mask, and so on.) Initially, when the JCT is created, all the bits in the job exit mask are set to one.

For a job-related exit, the status of its corresponding bit in the job-exit mask becomes an additional factor in determining its exit status. If an exit has been enabled in the standard way, by either the EXIT(nnn) initialization statement or the $T EXIT(nnn) command, and its corresponding bit in the job exit mask is set to one, the exit has a status of enabled and the exit effector calls its associated exit routine(s). If, however, the exit has been enabled in the standard way but its corresponding bit in the job exit mask is set to zero, the exit has a status of disabled and the exit effector does not call its associated exit routine(s) for that particular job. If the exit has been disabled in the standard way, the status of its corresponding bit in the job exit mask is not taken into account; the exit remains disabled. Note that if JOBMASK is not specified on the $EXIT macro, or if the JCT is not in storage, the job exit mask can have no effect on the status of an exit.
Bits in the job exit mask can be manipulated by an exit routine on a job-by-job basis. The recommended IBM-defined exits for setting the job exit mask are Exit 2 and Exit 52. Exit 2 or Exit 52 is, in most cases, the first exit to be taken for a job, and provides access to most of the job’s attributes specified in its JCL and placed in its JCT. For more information, see the descriptions of Exit 2 and Exit 52 in “The IBM-Defined Exits” reference section in “Chapter 3 - IBM-defined exits” on page 59.

For each exit description in “The IBM-Defined Exits”, the JOB EXIT MASK category lists the exit as either job-related or not job-related. Note that Exits 11 and 12 present special cases.

Appendix C, “Job-related exit scenarios,” on page 379 provides scenarios for job-related exits.
Tracing status

You can also control the status of exit invocation tracing.

Initially, for the tracing to occur automatically, three conditions are necessary:

1. The trace ID for exit tracing (ID 13) must be enabled.
2. The TRACE= operand of the EXIT(nnn) initialization statement must be specified as, or allowed to default to, TRACE=YES.
3. Tracing must be active (TRACEDEF ACTIVE=YES).

If one of these conditions is absent, tracing does not occur.

The status of exit tracing can then be dynamically controlled by the operator, using the $T EXIT(nnn) command. The operator has the option of identifying any exit by number, a range of exits, or all exits, and specifying either TRACE=YES or TRACE=NO. The operator can display the status of exit tracing by identifying the exit by number on the $D EXIT(nnn) command.

The status of exit tracing cannot be controlled on a job-related basis.
Establishing installation-defined exits

JES2 can contain up to 256 exits. IBM has defined some of these. If none of the IBM-defined exits is suited to a particular modification you would like to make, you can consider installing an optional installation-defined exit.

Typically, establishing your own exit is much more difficult than writing an exit routine for an existing IBM-defined exit; it requires a thorough knowledge of the area of processing in which you would like your exit to occur. You should attempt to place a installation-defined exit in a stable area of processing; the risk of error increases with the complexity of the JES2 code in which you place the exit. If possible, you should use your exit in replacing a JES2 function that is already isolated. As an example, IBM-defined Exit 3 allows you to provide an exit routine to completely replace the standard HASPRSCN accounting field scan routine.

You must consider whether the exit will require a single exit point or more than one. You can determine this based on the requirements of your intended modification and on the structure of the IBM code in the area of processing that you intend to modify. You must also consider whether the function you want to modify is contained within a single JES2 execution environment. If it occurs in a second environment, you may have to install a second exit as well.

When you have determined the exact point of processing at which an exit point must occur, use the $EXIT macro to define it.

First, you should specify the positional ID parameter with the exit’s identification number. It is recommended that you begin numbering installation-defined exits with 255 and work down. (If additional IBM-defined exits are added later, your exit numbers will not conflict with the new IBM-defined exit numbers.)

You must define the exit’s environment to JES2 using the ENVIRON= operand on the $MODULE macro. This is specified as either JES2, SUBTASK, USER, or FSS.

If the exit is to be job-related, specify the address of the job exit mask for the JOBMASK= operand. Note that if the JCT is not in storage you will have to point to a copy of the job exit mask.

Use the TYPE= operand to specify the mode of $EXIT macro operation. To avoid special processing overhead, you can define a TYPE=TEST $EXIT macro at some location shortly before a TYPE=ENTER $EXIT macro in JES2 code. A TEST-type $EXIT macro tests the status of the exit and sets a condition code (not a return code):

- cc=0: No exit routines are to be called
- cc=1: Call exit routines, without tracing
- cc=2: Call exit routines, with tracing

When JES2 encounters the TYPE=ENTER $EXIT macro, it does not have to retest the exit’s status; it just checks the condition code and either bypasses the exit point or calls the exit effector, with or without tracing. Note that a TYPE=TEST $EXIT macro and a TYPE=ENTER $EXIT macro must always be used together. If you omit the TYPE= parameter, the resulting exit point causes JES2 to both determine the status of the exit and then, depending on the status, either to bypass the exit point or to call the exit effector.
Use the AUTOTR= operand to specify that automatic exit effector tracing should (AUTOTR=YES) or should not (AUTOTR=NO) occur.

For more information about exit effector tracing, see “Tracing” in “Writing an Exit Routine” and “Tracing Status” in “Controlling Exit Status” earlier in this chapter.

Along with inserting the $EXIT macro in JES2 source code, you may have to modify the code before the exit point to pass parameters and pointers to the exit routines, and you may have to modify the code following the exit point to receive exit-generated parameters and to receive any return code greater than 4. For more information, see “Linkage Conventions,” “Received Parameters,” and “Return Codes” in “Writing an Exit Routine” earlier in this chapter.

**Note:** When using the $EXIT macro, you may need to include additional control block DSECT mappings in that module. If, for example, the module you are modifying did not previously require the mapping provided by the $XIT macros, but this macro is required to map the exit parameter list and exit information table (XIT), you must add it ($XIT) to the $MODULE macro coded at the beginning of the module.
Hints for coding JES2 exit routines

Following these hints can help you in the following ways:

- Improve your code’s readability and simplify debugging of your exit code.
- Ease migration to a new release or maintenance level.
- Reduce the number of errors in your exit code.

Assembler instructions

- All USING/DROP statements should be paired. No overriding USINGs should be used except when PUSH/POP is used. This helps prevent errors caused by incorrect base registers.
- All TM (test-under-mask) instructions should use BO/BOR/BNO/BNOR/BM/BMR branch instructions rather than BZ/BZR/BNZ/BNZR branch instructions. If this technique is used, the logic of the branch instruction does not have to be modified when adding or deleting flags in the instruction mask.
- Branches to *- or */ should not be used except in macro code. This reduces the possibility of causing errors when inserting new lines of code that change the offset of the instruction to which the code is branching.
- Branch tables should be fully coded and documented. Branches to a non-labeled line immediately after the branch table should not be used.
- To increase code readability, all branch instructions should use the extended mnemonic instructions for both RX and RR machine instruction formats.
- All flag bits in flag-byte fields should be defined by equated symbols. Explicit hexadecimal constants should not be used within instructions to represent flag-bit settings. This allows easy reference to a given flag setting. The SI format instructions TM, OI, NI, and XI should also use equated symbols. To provide easy reference, these instructions should use equated symbols for their masks.
- When the implied length of the target field cannot be used, instructions containing length fields should use equated symbols, not hard-coded lengths. Therefore, only a reassembly is necessary if the length of the field is changed.

Constants

- Rather than using literals, the HCT/HCCT/HFCT DSECTs define many constants which you should use whenever possible. The following are a few examples from the HCT:
  - $ZEROES – doubleword of binary zeroes
  - $F1 – fullword binary one
  - $H4 – halfword binary four
  - $BLANKS – doubleword of EBCDIC blanks (X’40’)

DSECTs

- For ease of migration, mapping DSECTs used as templates should not be explicitly duplicated within source code. An example of this technique is the use of JES2 $PDDB macro.
- Whenever possible, the use of locally-defined DSECTs, macros, or equated symbols should be avoided. This technique helps to avoid future migration problems.
- If you leave a control section (CSECT or RSECT) to define a DSECT, to return to the control section, use the &J2SECTN and &J2SECTT; assembly variables.
  - &J2SECTN contains the control section name.
&J2SECTT contains the control section type, either CSECT or RSECT.

For example:

MYMOD $MODULE ENVIRON=USER,

::

***********************************************************************
* *
* DEFINE DATA *
* *
***********************************************************************
MYDATA DSECT

::

***********************************************************************
* *
* RETURN TO CONTROL SECTION *
* *
***********************************************************************
&J2SECTN &J2SECTT

Registers

• Equated symbols for general purpose registers 0 to 15 (R0-R15) should be used.

• The general-purpose register equates used throughout JES2 are as follows:
  - R0: Parameter passing
  - R1: Parameter passing
  - R11: HCT addressability (JES2 main task)
  - R11: HCT addressability (JES2 subtasks)
  - R11: HFCT addressability (FSS)
  - R11: HCCT addressability
  - R12: Local addressability if $SAVE/$RETURN
  - R13: PCE addressability (JES2 main task)
  - R13: Save area address (FSS)
  - R13: Save area address
  - R14: Return address
  - R15: Entry address/return code

Miscellaneous

• Returned information used for routines and subroutines should use return codes, not condition codes. All return codes should be passed in register 15.

• Except in critical performance areas, the use of dynamic work areas rather than PCE work areas (for example, using $GETCMB to obtain a message building work area) is recommended. Dynamic work areas should be used to prevent unnecessary wasted storage caused by defining many unique PCE work area fields.

• The inclusive OR instruction (OC) should not be used to test whether a field is zero or non-zero. The OC can cause unnecessary page-outs, thus incurring needless system overhead. Rather, the CLC (compare logical) instruction can be used to compare the field with an appropriate constant (for example, $ZEROES).

• All code should be documented clearly and concisely. A good rule is to document every line of code. In addition, block comments should be used to document every module, routine, and subroutine. These comments should include detailed information about the function of the routine, register values required on entry and exit, register usage within the routine, and possible return codes.
Chapter 3 - IBM-defined exits

This reference chapter provides the information you need to write exit routines for the IBM-defined exits.

The exits are described in the order of their identification numbers, the ID numbers assigned to them on their respective $EXIT macros. Each exit description begins with a discussion of its recommended use, followed by a breakdown of environmental considerations, linkage conventions, and other programming considerations specific to the particular exit being described. (Note: For convenience, except where single or multiple exit routines are mentioned specifically, the following descriptions imply either one or more exit routines by the inclusive term “exit routine.” For example, “your exit routine may replace the standard routine” should be understood to mean “your exit routine or exit routines may replace the standard routine.”) Table 4 summarizes for each exit the CSECT in JES2 from which your exit routine can get control.

Exit selection table

When considering an alteration to a standard JES2 function, you should determine whether one of the IBM-defined exits accommodates your intended change.

The exit selection table (Table 3) summarizes the available exits and their functions. If you use an IBM-defined exit for other than its intended purpose, you increase the risk of performance degradation and system failure.

Appendix C, “Job-related exit scenarios,” on page 379 contains some scenarios relating to job-related exits. The scenarios may be helpful to you in deciding what exits to use in particular situations.

Table 3. Exit Selection Table

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
</table>
| 0    | PRE-INITIALIZATION| Control the initialization process | • Provide verification of JES2 initialization options, specifically $HASP426 and $HASP427 messages.  
• Acquire user control blocks and user work areas for use in initialization (such as the user control table (UCT)).  
• Provide addresses of user tables in the master control table (MCT).  
• Determine whether JES2 initialization is to continue.  
• Allow implementation of installation-defined initialization options and parameters. |
<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>JES2 PRINT/PUNCH JOB SEPARATOR</td>
<td>Create your own print and punch job separators and control production of standard separators.</td>
<td>• Selectively produce unique separators or variations on the standard separators. &lt;br&gt; • Unconditionally produce standard separators. &lt;br&gt; • Unconditionally suppress production of the standard separators. &lt;br&gt; • Selectively produce separators for particular users or particular job classes. &lt;br&gt; • Provide a different separator card on a punch device. &lt;br&gt; • Place the company's logo on header page. &lt;br&gt; • Provide accounting information on the trailer page.</td>
</tr>
<tr>
<td>2</td>
<td>JOB STATEMENT SCAN (Main Task)</td>
<td>Scan the complete JOB statement image and set corresponding fields in the appropriate JES2 control blocks.</td>
<td>• Alter JOB statement parameters including a job's class, priority, and other attributes. &lt;br&gt; • Supply additional JOB statement parameters. &lt;br&gt; • Selectively cancel or purge jobs. &lt;br&gt; • Set the job exit mask in the JCT for subsequent exits. &lt;br&gt; • Set the spool partitioning mask in the JCT. &lt;br&gt; • Initialize or modify other fields in the JCT, including your own installation defined fields. &lt;br&gt; • Modify other job-related control blocks. &lt;br&gt; • Build your own installation-defined job-related control blocks. &lt;br&gt; • Enforce security and standards.</td>
</tr>
<tr>
<td>3</td>
<td>JOB STATEMENT ACCOUNTING FIELD SCAN (Main Task)</td>
<td>Scan the JOB statement accounting field and set corresponding fields in the appropriate JES2 control blocks.</td>
<td>• Alter accounting field information. &lt;br&gt; • Supply additional accounting field information. &lt;br&gt; • Perform your own accounting field scan. &lt;br&gt; • Process nonstandard accounting fields. &lt;br&gt; • Selectively cancel jobs. &lt;br&gt; • Set the job exit mask in the JCT for future exits. &lt;br&gt; • Initialize or modify other fields in the JCT, including your own installation-defined fields. &lt;br&gt; • Pass information to subsequent exits through the JCT user fields. &lt;br&gt; • Modify other job-related control blocks. &lt;br&gt; • Enforce security and standards.</td>
</tr>
</tbody>
</table>
### Table 3. Exit Selection Table (continued)

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
</table>
| 4    | JCL AND JES2 CONTROL STATEMENT SCAN (Main Task)                           | Scan JCL (not including JOB statements).                                | • Alter JCL parameters and JES2 control statements.  
• Supply additional JCL parameters.  
• Supply a JCL continuation statement.  
• Alter JES2 control statements.  
• Supply an additional JES2 control statement.  
• Perform your own JES2 control statement processing.  
• Suppress standard JES2 processing.  
• Process your own installation defined JES2 control statement subparameters.  
• Selectively cancel or purge jobs.  
• Enforce security and standards. |
| 5    | JES2 COMMAND PREPROCESSOR                                                 | Process JES2 commands received by the JES2 command processor.           | • Alter received commands  
• Alter particular fields, such as those pertaining to command authority, in the command processor work area for the PCE to affect subsequent command processing.  
• Perform your own command validation checking.  
• Process your own installation-defined commands, operands, and suboperands.  
• Selectively terminate command processing and notify the operator of command cancellation. |
| 6    | CONVERTER/INTERPRETER TEXT SCAN                                           | Scan converter/interpreter text after conversion from individual JCL images and after all of the converter/interpreter text for a particular job has been created. | • Scan the resolved JCL, including PROCLIB expansion that will be used by the job.  
• Modify individual converter/interpreter text images.  
• Enforce security and standards. |
| 7    | CONTROL BLOCK READ/WRITE (JES2)                                           | Receive control whenever control block I/O is performed by the JES2 main task. | • Read or write your own installation-defined job-related control blocks to spool along with the reading and writing of JES2 control blocks. |
| 8    | CONTROL BLOCK READ/WRITE (USER)                                           | Receive control whenever control block (CB) I/O is performed by a JES2 subtask or by a routine running in the user address space. | • Read or write installation-defined job-related control blocks to spool along with reading and writing of the JES2 control block. |
| 9    | JOB OUTPUT OVERFLOW                                                      | Receive control whenever an executing job is producing more output than was estimated. | • Selectively allow JES2 to follow the defined output overflow error procedure.  
• Selectively direct JES2 to take special action for the current job only to:  
  - Cancel the job  
  - Cancel the job with a dump  
  - Allow the job to continue  
  - Extend the job’s estimated output to a specific new limit  
  - Control how often the output overflow message is displayed  
  - Suppress the default error message |
<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
</table>
| 10   | $WTO SCREEN | Receive control whenever JES2 is ready to queue a $WTO message. | • Scan messages.  
• Change the text of a message.  
• Alter a message's console routing.  
• Selectively suppress messages. |
| 11   | SPOOL PARTITIONING ALLOCATION – $TRACK | Receive control from the main task when there are no more track groups available on the spool volumes from which the current job is permitted to allocate space. | • Expand the spool partitioning mask.  
• Suppress spool partitioning by allowing JES2 to use the allocation default. |
| 12   | SPOOL PARTITIONING ALLOCATION – $STRAK | Receive control from the JES2 subtask or user address space when there are no more track groups available on the spool volumes from which the current job is permitted to allocate space. | • Expand the spool partitioning mask.  
• Suppress spool partitioning by allowing JES2 to use the allocation default. |
| 14   | JOB QUEUE WORK SELECT | Receive control to search the job queue for work. | • Use tailored search algorithms to select work from the job queue.  
• Selectively bypass searching the job queue for work. |
| 15   | OUTPUT DATA SET/COPY | Receive control to handle the creation of separator pages on a data set or copy basis. | • Selectively generate separator pages for each data set to be printed.  
• Selectively generate separator pages for each copy made of a data set.  
• Selectively vary the number of copies made of a data set.  
• Selectively pick data sets and generate separator pages for them.  
• Change default print translation tables. |
| 16   | NOTIFY | Receive control to examine or modify messages that are sent. | • Alter routing of the notify message.  
• Examine the notify message before it is sent to the receiver and make selective changes.  
• Suppress sending the notify message to the receiver.  
• Replace the notify message before it is sent to the receiver with an entirely new one. |
| 17   | BSC RJE SIGN-ON/SIGN-OFF | Receive control to manage and monitor RJE operations for BSC. | • Selectively perform additional security checks over and above the standard password processing of the signon card image.  
• Selectively limit both the number and types of remote devices that can be on the system at any one time.  
• Selectively bypass security checks.  
• Implement installation-defined scanning of signon card images.  
• Collect statistics concerning RJE operations on the BSC line and report the results of the activity. |
Table 3. Exit Selection Table (continued)

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>SNA RJE LOGON/LOGOFF</td>
<td>Receive control to manage and monitor RJE operations for SNA.</td>
<td>• Selectively perform additional security checks over and above the standard password processing of the logon image.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Selectively limit both the number and types of remote devices that can be on the system at any one time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Selectively bypass security checks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Implement installation-defined scanning of images.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Collect statistics concerning RJE operations on the line and report the results of the activity.</td>
</tr>
<tr>
<td>19</td>
<td>INITIALIZATION STATEMENT</td>
<td>Receive control for each initialization statement.</td>
<td>• Insert installation initialization statements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Scan an initialization statement before the JES2 scan and perform parameter checking.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Selectively alter values supplied on an initialization statement to meet specific installation needs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Optionally cause JES2 to bypass a particular initialization statement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Optionally cause JES2 to terminate.</td>
</tr>
<tr>
<td>20</td>
<td>END OF JOB INPUT (Main Task)</td>
<td>Alter the status of the job at the end of job input</td>
<td>• Selectively assign a job’s system affinity, execution node, and priority based on an installation’s unique requirements and processing workload.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Based on an installation’s own defined criteria, terminate a job’s normal processing and selectively print or not print its output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• JCT is available for updating.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Provide job tracking.</td>
</tr>
<tr>
<td>21</td>
<td>SMF RECORD</td>
<td>Receive control when JES2 is about to queue an SMF buffer.</td>
<td>• Selectively queue or not queue the SMF record for processing by SMF.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Obtain and create SMF control blocks before queuing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Alter content and length of SMF control blocks before queuing.</td>
</tr>
<tr>
<td>22</td>
<td>CANCEL/STATUS</td>
<td>Receive control to implement an installation’s own algorithms governing job selection and ownership for TSO/E CANCEL/STATUS.</td>
<td>• Allow an installation to implement its own algorithms for job queue searching and for TSO/E CANCEL/STATUS.</td>
</tr>
<tr>
<td>23</td>
<td>FSS JOB SEPARATOR</td>
<td>Receive control to modify the job separator page area (JSPA) that is used by page-mode printers such as the AFP printer to generate the job separator page for an output group.</td>
<td>• Control what information is passed to a page-mode printer functional subsystem application (FSA) through the JSPA.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Suppress the printing of job separator pages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Suppress the printing of the JESNEWS data set.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exit Title</td>
<td>Purpose</td>
<td>Some specific uses</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 24   | POST INITIALIZATION        | Receive control to make modifications to JES2 control blocks before the end of JES2 initialization. | • Make final modifications to selected JES2 control blocks before the end of JES2 initialization.  
• Initialize any special installation-defined control blocks.  
• Terminate JES2 during the initialization process. |
| 25   | JCT READ (FSS)             | Receive control whenever JCT read I/O is performed by a JES2 functional subsystem address space (HASPFOSSM). | • Read or write your own installation-defined job-related control blocks to spool along with the reading of the JCT. |
| 26   | TERMINATION / RESOURCE RELEASE | Free resources obtained during previous installation exit routine processing during any JES2 termination. | • Free resources obtained by user-exit routine processing that JES2 continues to hold following a $P JES2 command, JES2 initialization termination, or JES2 abend. |
| 27   | PCE ATTACH/DETACH          | Allocate and deallocate resources. Deny a PCE attach.                  | • Obtain resources whenever a PCE is attached.  
• Free resources before the detach of a PCE.  
• Deny the attach of a PCE. |
| 28   | SSI JOB TERMINATION        | Receive control before the freeing of job-related control blocks.       | • Free resources obtained by Exit 32.  
• Suppress job termination-related messages.  
• Replace JES2 job termination messages with installation-defined messages. |
| 29   | SSI END-OF-MEMORY          | Free resources obtained on the address space level.                      | • Free resources obtained by Exit 32. |
| 30   | SSI DATA SET OPEN/RESTART  | Receive control during SSI data set OPEN and RESTART processing.        | • Examine data set characteristics for validity checking, authorization, and alteration. |
| 31   | SSI DATA SET ALLOCATION    | Receive control during SSI data set allocation.                          | • Affect how JES2 processes data set characteristics.  
• Fail an allocation. |
| 32   | SSI JOB SELECTION          | Receive control during SSI job selection processing.                     | • Perform job-related processing such as allocation of resources and I/O for installation-defined control blocks.  
• Suppress job selection-related messages.  
• Replace job selection-related messages with installation-defined messages. |
| 33   | SSI DATA SET CLOSE         | Receive control during SSI data set CLOSE processing.                    | • Examine data set characteristics for validity checking, authorization, or alteration.  
• Free resources obtained at OPEN. |
| 34   | SSI DATA SET UNALLOCATION  | Receive control during SSI unallocation processing.                      | • Free resources obtained by Exit 30  
• Undo processing performed by Exit 30, such as changing data set characteristics. |
<p>| 35   | SSI END-OF-TASK            | Receive control during end of task processing.                           | • Free task-related resources. |</p>
<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
</table>
| 36   | Pre-security Authorization Call | Receive control before calling SAF. | • Provide additional information to SAF  
• Change information provided to SAF  
• eliminate call to SAF  
• Perform additional security authorization checking above what SAF provides |
| 37   | Post-security Authorization Call | Receive control after calling SAF. | • Change the result of SAF verification  
• Perform additional security authorization checking above what SAF provides |
| 38   | TSO/E Receive Data Set Disposition | Receive control during processing of a TSO/E RECEIVE command | • Change the default processing (delete) if a TSO/E user cannot receive a data set with any security information in the user profile. |
| 39   | NJE SYSOUT Reception Data Set Disposition (Main Task) | Receive control when your system receives a data set from another node that fails security checks. | • Override the security decision and accept the data set  
• Change the security information and accept the data set  
• Delete the data set |
| 40   | Modifying SYSOUT characteristics | Receives control before JOEs are created for the job. | • Change the class of a SYSOUT data set to affect grouping.  
• Change the destination of a SYSOUT data set. |
| 41   | Modifying Output Grouping Key Selection | Receives control during JES2 initialization after the default output grouping keys have been selected, but before any grouping is done. | • Change which OUTPUT JCL keywords JES2 uses for generic grouping. |
| 42   | Modifying a Notify User Message | Receives control after input has been validated and authorization checking has been done for the userid and node. | • Cancel the message  
• Change the destination of the message  
• Change the message text |
| 43   | Transaction Program Select/Terminate Change | Receives control during transaction: • select processing  
• termination processing  
• change processing | • Create installation-specific control blocks for the TP  
• Modify output limits associated with any SYSOUT data sets created by the TP  
• Issue messages to the TP’s message log |
| 44   | Exit for Converter Main Task | Receives control after the converter subtask has converted the job’s JCL and before JES2 writes the job-related control blocks to spool. | • Change fields in the $JQE and $JCT  
• Detect and hold duplicate TSO logons |
| 45   | Pre-SJF Service Request | Receives control from a request for scheduler JCL facility (SJF) services. | • Examine the request to determine if the system should continue to process the request for SJF services.  
• Redirect error messages for a request. |
| 46   | Transmitting an NJE Data Area | Receives control before JES2 transmitting an NJE job header, NJE data set header, or a NJE job trailer. | • Remove installation-defined sections that were previously added to an NJE data area  
• Add or change information in an NJE data area before transmitting it to another node in the network. |
### Table 3. Exit Selection Table (continued)

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
</table>
| 47   | Receiving an NJE Data Area | Receives control before receiving an NJE job header, NJE data set header, or an NJE job trailer. | - Add or remove installation-defined sections that were previously added to an NJE data area  
- Add or change information in an NJE data area before transmitting it to another node in the network. |
| 48   | SSI SYSOUT data set unallocation | Receive control after JES2 has merged the characteristics from the SSOB into the PDDB. | - Control whether JES2 spins the SYSOUT data set. |
| 49   | Job Queue Work Select - QGOT | Receives control whenever JES2 work selection has located a pre-execution job for a device. | - Provide an algorithm to accept or not accept a JES2-selected job.  
- Control WLM initiator job selection. |
| 50   | END OF JOB INPUT (User Environment) | Alter the status of the job at the end of job input | - Selectively assign a job’s system affinity, execution node, and priority based on an installation’s unique requirements and processing workload.  
- Based on an installation’s own defined criteria, terminate a job’s normal processing and selectively print or not print its output.  
- JCT is available for updating.  
- Provide job tracking. |
| 51   | Job phase change exit ($QMOD) | Change job phase | - Track jobs as they move from phase to phase.  
- Perform main task processing for jobs that arrive through the internal reader or NJE/TCP  
- Cause or prevent re-execution of jobs  
- Implement phase change rules for jobs |
| 52   | JOB STATEMENT SCAN (User Environment) | Scan the complete JOB statement image and set corresponding fields in the appropriate JES2 control blocks. | - Alter JOB statement parameters including a job’s class, priority, and other attributes.  
- Supply additional JOB statement parameters.  
- Selectively cancel or purge jobs.  
- Set the job exit mask in the JCT for subsequent exits.  
- Set the spool partitioning mask in the JCT.  
- Initialize or modify other fields in the JCT, including your own installation defined fields.  
- Modify other job-related control blocks.  
- Build your own installation-defined job-related control blocks.  
- Enforce security and standards. |
<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Purpose</th>
<th>Some specific uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>JOB STATEMENT ACCOUNTING FIELD SCAN (User Environment)</td>
<td>Scan the JOB statement accounting field and set corresponding fields in the appropriate JES2 control blocks.</td>
<td>• Alter accounting field information. • Supply additional accounting field information. • Perform your own accounting field scan. • Process nonstandard accounting fields. • Selectively cancel jobs. • Set the job exit mask in the JCT for future exits. • Initialize or modify other fields in the JCT, including your own installation-defined fields. • Pass information to subsequent exits through the JCT user fields. • Modify other job-related control blocks. • Enforce security and standards.</td>
</tr>
<tr>
<td>54</td>
<td>JCL AND JES2 CONTROL STATEMENT SCAN (User Environment)</td>
<td>Scan JCL (not including JOB statements).</td>
<td>• Alter JCL parameters and JES2 control statements. • Supply additional JCL parameters. • Supply a JCL continuation statement. • Alter JES2 control statements. • Supply an additional JES2 control statement. • Perform your own JES2 control statement processing. • Suppress standard JES2 processing. • Process your own installation defined JES2 control statement subparameters. • Selectively cancel or purge jobs. • Enforce security and standards.</td>
</tr>
<tr>
<td>55</td>
<td>NJE SYSOUT Reception Data Set Disposition (User Environment)</td>
<td>Receive control when your system receives a data set from another node that fails security checks.</td>
<td>• Override the security decision and accept the data set • Change the security information and accept the data set • Delete the data set</td>
</tr>
<tr>
<td>56</td>
<td>Transmitting an NJE Data Area (User Environment)</td>
<td>Receives control before JES2 transmitting an NJE job header, NJE data set header, or a NJE job trailer.</td>
<td>• Remove installation-defined sections that were previously added to an NJE data area • Add or change information in an NJE data area before transmitting it to another node in the network.</td>
</tr>
<tr>
<td>57</td>
<td>Receiving an NJE Data Area (User Environment)</td>
<td>Receives control before receiving an NJE job header, NJE data set header, or an NJE job trailer.</td>
<td>• Add or remove installation-defined sections that were previously added to an NJE data area • Add or change information in an NJE data area before transmitting it to another node in the network.</td>
</tr>
</tbody>
</table>
Exit implementation table

The following table is a reference to the various CSECTs from which IBM-defined exits can be taken and the JES2 environment in which the exit may be taken, including an indication regarding whether the exit is subject to job exit mask suppression. Use this table to help you implement your exit routines. See the $MODULE macro for descriptions of the environments.

Table 4. Exit Implementation Table

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Containing CSECT</th>
<th>Environment ($MODULE ENVIRON=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRE-INITIALIZATION</td>
<td>HASPIRMA</td>
<td>JES2 (Initialization) Job Exit Mask – N/A</td>
</tr>
<tr>
<td>1</td>
<td>PRINT/PUNCH SEPARATOR</td>
<td>HASPPRPU</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>2</td>
<td>JOB STATEMENT SCAN</td>
<td>HASPRDR</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>3</td>
<td>JOB STATEMENT ACCOUNTING FIELD SCAN</td>
<td>HASPRDR</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>4</td>
<td>JCL AND JES2 CONTROL STATEMENT SCAN</td>
<td>HASPRDR</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>5</td>
<td>JES2 COMMAND PREPROCESSOR</td>
<td>HASPCOM</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>6</td>
<td>CONVERTER/INTERPRETER TEXT SCAN</td>
<td>HOSCNVT subtask of HASPCNVS</td>
<td>SUBTASK Job Exit Mask</td>
</tr>
<tr>
<td>7</td>
<td>CONTROL BLOCK READ/WRITE (JES2)</td>
<td>HASPNUC</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>8</td>
<td>CONTROL BLOCK READ/WRITE (USER)</td>
<td>HASCSRDS</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>9</td>
<td>JOB OUTPUT OVERFLOW</td>
<td>HASCHAM</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>10</td>
<td>$WTO SCREEN</td>
<td>HASPCON</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>11</td>
<td>SPOOL PARTITIONING ALLOCATION – $TRACK</td>
<td>HASPTRAK</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>12</td>
<td>SPOOL PARTITIONING ALLOCATION – $STRAK</td>
<td>HASCSRIC</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>14</td>
<td>JOB QUEUE WORK SELECT</td>
<td>HASPJQS</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>15</td>
<td>OUTPUT DATA SET/COPY SEPARATORS</td>
<td>HASPPRPU</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>16</td>
<td>NOTIFY</td>
<td>HASPHOPE</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>17</td>
<td>BSC RJE SIGN-ON/SIGN-OFF</td>
<td>HASPBSC</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>18</td>
<td>SNA RJE LOGON/LOGOFF</td>
<td>HASPSNA</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>19</td>
<td>INITIALIZATION STATEMENT</td>
<td>HASPIRPL</td>
<td>JES2 (Initialization) Job Exit Mask – N/A</td>
</tr>
<tr>
<td>20</td>
<td>END OF JOB INPUT</td>
<td>HASCSRIP</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>21</td>
<td>SMF RECORD</td>
<td>HASPNUC</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>22</td>
<td>CANCEL/STATUS</td>
<td>HASPSTAC</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>23</td>
<td>JOB SEPARATOR PROCESSING (JSPA)</td>
<td>HASPFSMM</td>
<td>FSS Job Exit Mask</td>
</tr>
<tr>
<td>24</td>
<td>POST INITIALIZATION</td>
<td>HASPIRA</td>
<td>JES2 (Initialization) Job Exit Mask – N/A</td>
</tr>
<tr>
<td>Exit</td>
<td>Exit Title</td>
<td>Containing CSECT</td>
<td>Environment (SMODULE ENVIRON=)</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>25</td>
<td>JCT READ I/O (FSS)</td>
<td>HASPFSSM</td>
<td>FSS Job Exit Mask</td>
</tr>
<tr>
<td>26</td>
<td>TERMINATION/RESOURCE RELEASE</td>
<td>HASPTERM</td>
<td>JES2 (Termination) Job Exit Mask – N/A</td>
</tr>
<tr>
<td>27</td>
<td>PCE ATTACH/DETACH</td>
<td>HASPDYN</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>28</td>
<td>SSI JOB TERMINATION</td>
<td>HASCJBST</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>29</td>
<td>SSI END-OF-MEMORY</td>
<td>HASCJBTR</td>
<td>USER Job Exit Mask – N/A</td>
</tr>
<tr>
<td>30</td>
<td>SSI DATA SET OPEN and RESTART</td>
<td>HASCDSOC</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>31</td>
<td>SSI DATA SET ALLOCATION</td>
<td>HASCDSAL</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>32</td>
<td>SSI JOB SELECTION</td>
<td>HASCJBST</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>33</td>
<td>SSI DATA SET CLOSE</td>
<td>HASCDSOC</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>34</td>
<td>SSI DATA SET UNALLOCATE</td>
<td>HASCDSAL</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>35</td>
<td>SSI END-OF-TASK</td>
<td>HASCJBTR</td>
<td>USER Job Exit Mask – N/A</td>
</tr>
<tr>
<td>36</td>
<td>Pre-Security Authorization Call</td>
<td>HASCSRIC</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>37</td>
<td>Post-Security Authorization Call</td>
<td>HASCSRIC</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>38</td>
<td>TSO/E Receive Data Set Disposition</td>
<td>HASPPSO</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>39</td>
<td>NJE SYSOUT Reception Data Set Disposition</td>
<td>HASPNET</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>40</td>
<td>Modifying SYSOUT Characteristics</td>
<td>HASPHOPE HASPSEQ</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
<tr>
<td>41</td>
<td>Modifying Output Grouping Key Selection</td>
<td>HASCGGKY</td>
<td>USER Job Exit Mask – N/A</td>
</tr>
<tr>
<td>42</td>
<td>Modifying a Notify User Message</td>
<td>HASCSIIRQ</td>
<td>USER Job Exit Mask – N/A</td>
</tr>
<tr>
<td>43</td>
<td>Transaction Program Select/Terminate/Change</td>
<td>HASCTP</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>44</td>
<td>JES2 Converter Exit</td>
<td>HASPCNVNT</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>45</td>
<td>Pre-SJF Exit Request</td>
<td>HASCSJFS</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>46</td>
<td>Transmitting an NJE Data Area</td>
<td>HASPNET</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>47</td>
<td>Receiving an NJE Data Area</td>
<td>HASPNET</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>48</td>
<td>SSI SYSOUT Data Set Unallocation</td>
<td>HASCDSAL</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>49</td>
<td>Job Queue Work Select - QGOT</td>
<td>HASPJQS</td>
<td>JES2 Job Exit Mask – N/A</td>
</tr>
</tbody>
</table>
### Exit Implementation Table (continued)

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Containing CSECT</th>
<th>Environment ($MODULE ENVIRON=)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>END OF JOB INPUT (User Environment)</td>
<td>HASCSRIP</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>51</td>
<td>Job phase change exit ($QMOD)</td>
<td>HASPJQS</td>
<td>JES2 Job Exit Mask</td>
</tr>
<tr>
<td>52</td>
<td>JOB STATEMENT SCAN (User Environment)</td>
<td>HASCINJR</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>53</td>
<td>JOB STATEMENT ACCOUNTING FIELD SCAN (User Environment)</td>
<td>HASCINJR</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>54</td>
<td>JCL AND JES2 CONTROL STATEMENT SCAN (User Environment)</td>
<td>HASCINJR</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>55</td>
<td>NJE SYSOUT Reception Data Set Disposition (User Environment)</td>
<td>HASCNJSR</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>56</td>
<td>Transmitting an NJE Data Area (User Environment)</td>
<td>HASCNJE</td>
<td>USER Job Exit Mask</td>
</tr>
<tr>
<td>57</td>
<td>Receiving an NJE Data Area (User Environment)</td>
<td>HASCNJE</td>
<td>USER Job Exit Mask</td>
</tr>
</tbody>
</table>
Exit 0: Pre-initialization

Function

This exit allows you to control the start of the initialization process through various means, such as:

- Processing JES2 initialization options, specifically the JES2 cataloged procedure parameter field or the replies to the $HASP426 and $HASP427 WTORs. The options can optionally be altered or bypassed.
- Acquiring installation-defined control blocks and installation work areas for later initialization.
- Providing user fields and addresses of installation-defined tables in the MCT. The table pointers in the master control table (MCT) allow your installation to extend JES2 processing of user tables to define JES2 initialization to extend or tailor certain table-driven JES2 functions. Define user table pointers in the MCT as MCTstmTU, where ‘stm’ is the JES2 initialization statement that you are replacing. See “Defining JES2 Tables” for a list of the MCT names.
- Determining whether JES2 initialization is to continue.

Environment

Task

JES2 main task (Initialization) – JES2 dispatcher disabled. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 0 in supervisor state and PSW key 1

Recovery

JES2 does not have a recovery environment established at the processing point for Exit 0 (the JES2 ESTAE will process termination but not recover).

Job exit mask

Exit 0 is not subject to suppression.

Mapping macros normally required

$HASPEQU, $HCT, $MIT, $PCE, $CIRWORK

Point of processing

This exit is taken in the initialization routine that processes the initialization options (IROPTS, in module HASPIRMA). The initialization options are taken from the parameter field specified through the JES2 procedure or START command, or are requested from the operator through the $HASP426 WTOR message if necessary. The point of processing for this exit is just before parsing and analyzing the options and setting appropriate flags. Exit 0 may be called a multiple number of times,
because new options may be requested repetitively through the $HASP427 WTOR message until valid options are specified or the exit directs JES2 to bypass the options analysis.

The exit control blocks and the exit effector are not initialized at this point in IROPTS when Exit 0 gets control. Therefore, the normal JES2 exit facility initialization parameters cannot be used. IROPTS searches for module HASPXIT0 in the HASPINIT load module and then, if necessary, in the HASJES20 load module. The name HASPXIT0 is defined as a weak external reference (WXTRN) in both load modules. If HASPXIT0 is not found through this search, JES2 attempts to locate a separate load module named HASPXIT0. Creating separate load modules for your exit routines is recommended. If HASPXIT0 is found in STEPLIB or LINKLIST, a temporary XIT and XRT are built for the exit facility and the $EXIT macro. The HASPXIT0 module’s MIT is searched for all entry point names of the form ‘EXIT0nnn’ and the entry point names found and the associated addresses are placed in the temporary XRT in the order they are found.

If HASPXIT0 is found during JES2 initialization, an entry for that module is placed in the exit facility LMT as if a LOADmod(jxxxxxxx) initialization statement had been processed for it and the module is not deleted. Therefore other exit routines (e.g., for Exits 19 and 24) and installation-defined tables (e.g., initialization statement $SCANTAB tables) can be assembled in the same module with the Exit 0 routines without having them deleted by JES2 after initialization completes. Note, however, that HASPXIT0 will be deleted from storage with HASPINIT if HASPXIT0 is linkeded with the HASPINIT load module. Exit 0 can also be invoked using the MVS Dynamic exit facility. JES2 invokes exit HASP.$EXIT0 immediately after the call to routines in HASPXIT0. The interface to any routines called in this fashion is identical to those invoked from HASPXIT0.

**Programming considerations**

1. Tracing for this exit is disabled because of its sequence in the initialization process.
2. Because Exit 0 is called early in JES2 initialization, some main task services may not be functional and most control blocks and interfaces are not yet established. The JES2 dispatcher is not yet functional, so MVS protocol should be used in Exit 0 routines (such as, WAIT rather than $WAIT, ESTAE rather than $ESTAE.).
3. If Exit 0 returns a return code of 12, IROPTS issues message $HASP864 indicating that Exit 0 terminated initialization. IROPTS then returns to the IRLOOP with return code 8, indicating that the $HASP428 message should be issued before final termination.
4. The initialization options string passed to Exit 0 is first ‘folded’, that is all the characters are ‘folded’ up to their capitalized versions.
5. The processing that JES2 does for the initialization options string after calling Exit 0 is performed using the JES2 $SCAN facility and a table that defines the options input allowed and how to process it. The table is actually composed of two tables, an installation-defined table followed by a JES2-defined table. By specifying installation-defined tables, an installation can implement its own initialization options or replace the JES2 definition for existing options. Thus this function can be accomplished without implementing Exit 0, or with an implementation of Exit 0. Also, the $SCAN facility itself can be used from an Exit 0 to process initialization options.
6. If HASPXIT0 contains dynamic tables, the tables will automatically be used when HASPXIT is loaded. It is also possible to include dynamic tables in a
module invoked by HASP:$EXIT0. However, when using HASP:$EXIT0, include any tables in a separate load module and invoke the $MODLOAD service to access the modules. If HASP:$EXIT0 is refreshed, any tables that the load module contains might move to a different storage location without JES2’s knowledge, resulting in unpredictable results.

Attention: This exit should be thoroughly tested in an environment that is totally inaccessible to your production JES2 environment (the data set containing the test version of the module that contains exit 0 should not be in the link list).

This exit cannot be disabled other than by replacing or removing the load module. A situation where JES2 cannot be initialized may occur if the exit is improperly coded. This risk can be minimized by using Exit 24 to define user tables for commands, rather than Exit 0. However, for installation defined installation statements, Exit 0 must be used.

Also, if the MCT table entries are modified, the associated tables must not be in the HASPINIT load module. This is because the HASPINIT load module is deleted after initialization, and the tables will become inaccessible. Note that this restriction applies regardless of whether the tables define initialization statements, commands, or messages.

Register contents when Exit 0 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code indicating where the initialization options were specified</td>
</tr>
<tr>
<td></td>
<td>0 Options passed are from the EXEC card, the PARM field</td>
</tr>
<tr>
<td></td>
<td>4 Options passed are from the $HASP426 message WTOR reply</td>
</tr>
<tr>
<td></td>
<td>8 Options passed are from a $HASP427 message WTOR reply</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 2-word parameter list with the following structure:</td>
</tr>
<tr>
<td></td>
<td>Word 1 (+0) address of the initialization options string</td>
</tr>
<tr>
<td></td>
<td>Word 2 (+4) length of the initialization options string</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of $HCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of the initialization $PCE – the PCE work area for this $PCE is the common initialization routine work area, mapped by the $CIRWORK macro.</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 0 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return Address</td>
</tr>
</tbody>
</table>
Exit 0

A return code

A return code of:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit, continue with normal IROPTS processing.

4  Tells JES2 to ignore any additional exit routines associated with this exit and to continue with normal IROPTS processing.

8  Tells JES2 to bypass processing of the options string and assume the current values for the JES2 initialization options flags are correct.

12 Tells JES2 to terminate processing. This results in the $HASP864 error message to the operator.

Coded example

Modules HASX00A and HASX00B in SYS1.SHASSAMP contain samples of exit 0.
Exit 1: Print/punch separators

Function

This exit allows you to:

- Produce your own print/punch separators
- Control production of standard print/punch separators for batch jobs or transaction programs (TP)
- Create separators that include the security label for the job output for JES2 managed printers, if your security policy requires it.

When using this exit to control the production of standard separators, you can:

- Unconditionally suppress production of standard separators
- Direct JES2 to unconditionally produce standard separators
- Allow JES2 to produce any standard separators that are in effect.

JES2 determines whether standard separators are in effect for any particular device by using the initialization statement or the operator command separator options provided by your installation at any given time; "Programming considerations" on page 76 describes these options.

For punch devices, JES2 provides the option of producing start-of-job header cards and trailer cards. For printers, JES2 provides the option of producing start-of-job header pages, continuation-of-job header pages, and trailer pages. Start-of-job header pages are produced at each output data set group (represented by a work JOE) within a job. Continuation-of-job header pages are produced for the continuation of a data set group if printing has been interrupted. Therefore, you are able to control the production of separators on a job-by-job basis and, for printers/punches on a data set group basis. See z/OS JES2 Initialization and Tuning Guide for a sample separator page.

Each time your exit routine is called, you can direct JES2:

- To produce only your own separator (unconditionally suppressing production of the standard separator)
- To produce only the standard separator, if it is in effect (without producing your own separator)
- To produce the standard separator unconditionally
- To produce your own separator followed by the standard separator, if the standard separator is in effect (for example, your own start-of-job header page followed by the standard start-of-job header page)
- To produce your own separator and then to produce the standard separator unconditionally
- To produce no separator (by not producing your own separator and by suppressing production of the standard separator)
- To print or suppress the JESNEWS data set, regardless of whether a separator is produced

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.
Exit 1

AMODE/RMODE requirements
AMODE 31, RMODE ANY

Restrictions
You cannot use this exit to modify the standard separator routines directly. If you intend to produce a modified version of a standard separator, your exit routine must replace the standard separator routine entirely, and is responsible for producing the standard separator elements that you want to retain and your new or modified separator elements.

Recovery
$ESTAE recovery is in effect. If a program check occurs in the exit, JES2 interrupts the output currently processing on the device. The recovery routine does not create a trailing separator and will not call Exit 1 to free allocated resources. JES2 places the interrupted output groups in system hold with an indication that a failure occurred during separator exit processing. As with every exit, you should supply your own recovery within your exit routine.

Job exit mask
Exit 1 is subject to job exit mask suppression. The installation can implement exit 2 to set the 1st bit in the job exit suppression mask (JCTXMASK) or the installation can indicate the exit is disabled in the JES2 initialization stream.

Mapping macros normally required
$BUFFER, $DCT, $DSCT, $HASPEQU, $HCT, $JCT, $JCTX, $JOE, $JQE, $PCE, $PDDB, $XPL

Point of processing
JES2 calls Exit 1 during print/punch processing before the check for standard separator pages. The exit is called for job header and job trailer separators.

Programming considerations
1. This exit is available to provide a user-written separator page for local or RJE printers only. There is no separator page for JES2 or user-supplied networking output. If you require separator pages for networking output jobs, the destination node must supply them (through use of this exit) when the output prints.
2. For each device, initialization statements first determine whether standard separators are in effect—that is, whether without an exit routine, JES2 would normally produce or suppress standard separators.
   For a local printer, the SEP=NO parameter of the PRT(nnnn) statement instructs JES2 not to produce separator pages, and the SEP=YES parameter instructs JES2 to produce separator pages. However, even if you specify SEP=YES, if SEPPAGE=(LOCAL=NONE) appears on the PRINTDEF statement, JES2 does not produce separator pages.
   For a remote printer, the SEP=NO parameter of the R(nnnn).PR(m) statement instructs JES2 not to produce separator pages, and the SEP=YES parameter instructs JES2 to produce separator pages. However, even if you specify SEP=YES, if SEPPAGE=(REMOTE=NONE) appears on the PRINTDEF statement, JES2 does not produce separator pages.
For a local card punch, the SEP=NO parameter of the PUN(nn) statement instructs JES2 not to produce separator cards, and the SEP=YES parameter instructs JES2 to produce separator cards.

For a remote card punch, the SEP=NO parameter of the R(nnnn).PU(m) statement instructs JES2 not to produce separator cards, and the SEP=YES parameter instructs JES2 to produce separator cards.

After you start JES2, the operator uses the S option of the $T PRT(nnnn) or $T PUN(nnn) command to change the status of any printer or card punch. For any device, if the operator issues the $T command with S=Y, JES2 produces standard separators; with S=N, JES2 does not produce standard separators.

3. Use the $PRPUT macro to produce any new separators your exit routine creates. $PRPUT passes back a return code of 4 in register 15 if the creation of the separator page is suspended or terminated.

4. Use the $PBLOCK macro to create block letters on any new separator page your exit routine creates.

5. If you are using the spooling capabilities of a remote SNA device such as the 3790, use the $SEPPDIR macro to send a peripheral data information record (PDIR) to the device.

6. **Locating Extensions to the JCT Control Block:** You can use the $JCTXGET macro to locate extensions to the job control table ($JCT) control block from Exit 1.

7. **Using Buffers in this Exit Routine:** JES2 provides this exit with a buffer to use for I/O. JES2 page-fixes the buffer, when needed, so the buffer can be used by the $PRPUT, $PBLOCK, and $SEPPDIR macros. The exit routine accesses the buffer by coding a USING statement for label BFPDSECT. The exit routine must not free the supplied buffer.

   Although IBM suggests using the buffer that JES2 provides, the installation has the option of obtaining its own buffer. Use the $GETBUF macro if your routine obtains its own buffer and the $FREEBUF macro to free the buffer. Code the following on the $GETBUF macro for any buffers you are using with $PBLOCK, $PRPUT, and $SEPPDIR:
   - TYPE=HASP
   - FIX=YES for buffers used for local devices
   - FIX=NO for buffers used for remote devices.

   Although you could page-fix all buffers using the FIX parameter on $GETBUF, this may lead to performance problems.

   When using $PRPUT with WAIT=NO, I/O does not occur synchronously. The device does not physically process the buffer until either you issue a $PRPUT macro specifying WAIT=YES or the CCW area fills. Therefore, issue $PRPUT with WAIT=YES before freeing the buffer.

8. If a hardware error or intervention situation interrupts $PRPUT processing, Exit 1 relinquishes control. When this occurs, JES2 can not deallocate any resources your exit routine allocated. You can prevent this situation from occurring by saving the addresses of allocated resources in a PCE field such as PCEUSER0 and checking for the address(es) on entry to the exit routine.

   Your routine can then reuse previously allocated resources and before returning to JES2, the routine can release the resources and zero the pointer field(s).

9. Some printers do not reposition to “top of forms” after the trailer page. To avoid feeding blank pages through your printer, include a page eject statement in your exit routine following the trailer separator page.

10. Use SWBTUREQ REQUEST=RETRIEVE to retrieve any parameters a user specifies on the OUTPUT JCL statement you need to build your separator.
11. You can determine if Exit 1 is being invoked for transaction program by examining field X001DSCT. If it contains an address, Exit 1 was invoked on behalf of a TP. Zeroes in this field indicate Exit 1 was invoked on behalf of a batch job.

12. For a TP, you will need to obtain the owner’s userid from the $JOE instead of the $JQE. You can continue to obtain the owner’s userid from the $JQE for batch jobs.

Register contents when Exit 1 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher - $XPL</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number - 1</td>
</tr>
<tr>
<td>XPLIND</td>
<td>Indicator byte. This byte indicates whether the exit was invoked for a job header, a job trailer, or a continuation.</td>
</tr>
<tr>
<td>X001JHDR</td>
<td>If this bit setting is on, then Exit 1 was invoked for a job header.</td>
</tr>
<tr>
<td>X001JTLR</td>
<td>If this bit setting is on, then Exit 1 was invoked for a job trailer.</td>
</tr>
<tr>
<td>X001JCNT</td>
<td>If this bit setting is on, then Exit 1 was invoked for a continuation.</td>
</tr>
<tr>
<td>X001RESP</td>
<td>Response byte. This response byte will indicate whether JES2 will produce standard separator pages or not, and whether it will produce JESNEWS or not. The response byte on entry can have the following values:</td>
</tr>
<tr>
<td>X001DFSP</td>
<td>If this bit setting is on, then the production of the standard separator page will be suppressed. Otherwise, the standard separator page will be produced.</td>
</tr>
<tr>
<td>X001JNWS</td>
<td>If this bit setting is on, then the production of JESNEWS will be suppressed. Otherwise, JESNEWS will be printed.</td>
</tr>
<tr>
<td>X001DCT</td>
<td>Address of $DCT</td>
</tr>
<tr>
<td>X001JCT</td>
<td>Address of $JCT</td>
</tr>
<tr>
<td>X001DSCT</td>
<td>Contains the address of the $DSCT for TPs or zeros for batch jobs.</td>
</tr>
<tr>
<td>X001JQE</td>
<td>Address of $JQE</td>
</tr>
<tr>
<td>X001JOA</td>
<td>Address of the artificial JOE (JOA). The JOA contains both the Work-JOE and the Characteristics-JOE.</td>
</tr>
</tbody>
</table>

Note: If the exit must update JOE fields, it should obtain and return an update mode JOA. For
more information, see “Checkpoint control blocks for JOEs” on page 384.

X001PDB
Address of the first PDB in the JOE. This field is zero for job trailers.

X001SWBT
Address of the scheduler work block text unit (SWBTU) pointer list for the first PDB in the JOE. The SWBTU pointer list is mapped by SJTRSBTL DSECT in the IEFSJTRP parameter list. This field is zero if there is no OUTPUT JCL statement associated with the first PDB. JES2 uses the SWBTU associated with the first PDB to retrieve the output identification and delivery information for the entire output group. From this information, JES2 builds the detail box in the default standard separator page.

X001NSWB
Number of SWBTUs JES2 despooled. 

X001HBUF
Address of a HASP buffer for this exit’s use. Mapping macro $BUFFER maps the buffer and label BUFSTART points to the beginning of the buffer work area. You must have a USING on field $BFPDSECT. Field $BUFSIZE in the $HCT contains the size of the buffer work area. The exit routine should not update any other fields in the buffer as errors will occur when control returns to JES2.

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X001RESP</td>
<td>This response byte can be set by the exit before returning to JES2 if you want to change the value on entry. Set the response byte as follows:</td>
<td></td>
</tr>
<tr>
<td>X001DFSP</td>
<td>Turn this bit setting on to suppress the standard separator page.</td>
<td></td>
</tr>
<tr>
<td>X001JNWS</td>
<td>Turn this bit setting on to suppress production of JESNEWS.</td>
<td></td>
</tr>
</tbody>
</table>

| Register contents when control passes back to JES2: |
| 0 | Unchanged |
| 1 | Pointer to a parameter list mapped by $XPL: |

A return code of:

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X001RESP</td>
<td>This response byte can be set by the exit before returning to JES2 if you want to change the value on entry. Set the response byte as follows:</td>
<td></td>
</tr>
<tr>
<td>X001DFSP</td>
<td>Turn this bit setting on to suppress the standard separator page.</td>
<td></td>
</tr>
<tr>
<td>X001JNWS</td>
<td>Turn this bit setting on to suppress production of JESNEWS.</td>
<td></td>
</tr>
</tbody>
</table>

0 | Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. |
Exit 1

4 Tells JES2 to ignore any additional exit routines associated with this exit.

Coded example

Modules HASX01A and HASX01B in SYS1.SHASSAMP contain a sample of Exit 1.
Exit 2: JOB JCL statement scan (JES2 main task)

Function

Exit 2 allows you to process information specified on the JOB JCL statement for jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload. (For jobs submitted through internal readers or TCP/IP NJE, exit 52 is called for JOB JCL statements.) Exit 2 is invoked for the initial JOB statement each continuation of the JOB card. The initial JOB card and all continuations are read before invoking the exit.

Using Exit 2 you can:
- Add, delete, change information specified on the JOB statement. If you are adding information, such as accounting information, you can create an additional JOB continuation statements.
- Indicate which spool volumes from which a job or transaction program should allocate spool space, if the installation did not implement spool partitioning through the JES2 initialization stream.
- Add JCL statements or JES2 control statements (JECL) to the job.
- Cancel, purge, or continue processing the job.
- Indicate whether additional job-related exits should be invoked for the job.

Recommendations for implementing Exit 2

Exit 2 is called for each card in the job statement (the original card and all continuations). Each time the exit is called, it will pass the current card image and the statement buffer. The statement buffer includes all the operands for the JOB statement concatenated in a single buffer. For example:

```
//TEST JOB (ACCOUNT),'PROGRAMMER', COMMENT 1
  // CLASS=A,MSGCLASS=A,  COMMENT 2
  // USER=TEST,PASSWORD=TEST  COMMENT 3
```

In this case the exit will be called 3 times, once for each card and will pass (on all 3 calls) the following data in the statement buffer (pointed to by X002STMT):

```
(ACCOUNT),'PROGRAMMER',CLASS=A,MSGCLASS=A,USER=TEST,PASSWORD=TEST
```

To alter the processing of the JOB card, the exit can:
- Update the card image passed in X002CARD. This change shows up in the listing of the job.
- Update the statement buffer in X002STMT to add or modify the operands. This change does not show up in the listing of the job and is not passed to conversion processing (it only affects keywords input processing scans from the JOB card).
  If you update the statement buffer (X002STMT) in Exit 2 and change the length of the buffer, you must update the field X002STME to indicate the new end of buffer (one byte past the last meaningful character).
- Add additional card images to the JCL stream.

You can add card images to the JCL stream by either queuing a single RJCB or a chain of RJCBs to the XPL, or by placing a card image after the current card into the area pointed to by X002JXWR and setting X002XSNC. In either case, when a card is added, the current card is re-scanned and the statement buffer is re-built.
Exit 2

Exit 2 is driven again for the updated statement, with X002SEC set to indicate this card has been presented to the exit previously.

When adding cards using RJCBs, use the RGETRJCB service (located in HASCSRIP) to obtain a free RJCB; then add it to one of the three RJCB queues in the XPL. Use the $CALL macro to invoke the RGETRJCB service. Register 1 on entry must be the JRW address. The RJCB address is returned in register 1.

The 80-byte card image to be added is placed into the field RJCBCARD. RJCBs are chained together using the RJCBRJCB field in the $RJCB. They are added to the job stream in the order they exist in the chain. To add an element to the chain you would move the current RJCB queue head in the $XPL into the RJCBRJCB field of the last RJCB you are adding and then set the address of the first RJCB element into the $XPL queue head. Be aware that multiple exit 2s might be using these queues to ensure that you do not lose existing entries on the queue.

X002RJCP
 Adds the card images before the first card in the current JOB statement.

X002RJCA
 Adds the card images after the last card in the current JOB statement. In this case, the card(s) are assumed to not be a continuation of the current job statement and the job card is not re-scanned.

X002RJCC
 Adds the card images after the current card. It is the callers’ responsibility to ensure that the proper continuation processing will occur.

When processing the last card in a JOB statement, the difference between adding a card to the X002RJCA queue and the X002RJCC queue is that the first will not re-scan the job card and the second will. You can also add a single card image after the current card using the X002JXWR field. In this case, the job card will be re-scanned just as if the card was added to the X002RJCC queue. To add information to the job JCL statement:

1. Move a comma into the last byte of the job statement image exit 2 is currently processing. The comma indicates that additional information follows on the job statement.

2. Move the information you want to add to the job statement to the area pointed to by X002JXWR and set the X002XSNC bit in the X002RESP byte to one. Setting X002RESP to X002XSNC indicates that the installation has supplied an additional job statement image.

3. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the job.

You can also add an additional job level JCL statement to the job as follows:

1. Ensure that the job statement image exit 2 is currently processing is the last. Exit 2 is processing the last job statement image if a comma is not in the last byte of the job statement image.

2. Place the job-level JCL statement in the area pointed to by X002JXWR and set the X002XSNC bit in the X002RESP byte to one. Setting X002RESP to X002XSNC indicates that the installation has supplied an additional job statement image.

3. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the job.

If you want to issue messages when you cancel or purge the job:
1. Generate the message text in exit 2.
2. Move the message text to area pointed to by X002JXWR and set the X002XSEM bit in X002RESP to one. Setting X002RESP to X002XSEM indicates that the installation exit has supplied an error message that will be added to the JCL listing.
3. Set register 15 to X'08' to indicate JES2 should cancel or purge the job.

The following indicators in the XPL can assist you in adding a card image to the current job statement:

**X002LOPR**
Current card has the last operand in the job statement. There may be additional continued comments after the current card.

**X002QUOT**
A quoted string is being continued from the current card to the next card. Pay attention if a card is being added after this card.

**X002CCMT**
The current card is a continued comment. Operand added to this card or after this card will not be processed.

**X002LAST**
This is the last card image in the JOB statement.

To assist you in processing the operands on a statement, you can use either of the following services to parse the statement buffer passed in X004STMT:

- Use the $SCAN facility to parse the operands with the standard $SCAN rules for statements. This give you the flexibility of $SCAN, but the parsing rules are not the same as normal JCL. See the $SCAN and $SCANTAB macros for additional information.
- Use the RCARDSCN service and $STMTTAB macro to parse the operands with standard JCL rules. This is the service used by JES2 input processing to parse the statement buffer. However, the RCARDSCN service only parses the operands and calls a processing routine to do all the conversions and storing of data. Conversion of data to binary to store into data areas is the responsibility of the processing routines. See the $STMTTAB macro for more information.

**Environment**

**Task**
JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

**AMODE/RMODE requirements**
AMODE 31, RMODE ANY

**Supervisor/problem program**
JES2 places exit 2 in supervisor state and PSW key 1.

**Restrictions**
- See [Appendix A, “JES2 exit usage limitations,” on page 373](#) for a listing of specific instances when this exit will be invoked or not invoked.
- Installation Exit 2 is not invoked for jobs such as SYSLOG, $TRCLOG, or JESMSG.
Exit 2

- Do not use this exit to set fields in the JCT; they will likely be overwritten by future processing.
- Installation Exit 2 is not invoked for jobs submitted through the internal reader or TCP/IP NJE.

Recovery

$ESTAE is in effect and provides minimal recovery. Input Services will attempt to recover from any program check errors experienced by exit 2. However, you should not depend on JES2 for recovery.

Job exit mask

Exit 2 and all subsequent job-related installation exits can be suppressed after Exit 2 processes the initial job statement image. You can set the 2nd bit in the job exit suppression mask (JCTXMASK) or you can indicate the exit is disabled in the JES2 initialization stream.

Storage recommendations

If exit 2 requires work areas or additional storage, you can:
- Use the 80-byte work area, JCTXWRK, in the JCT
- Issue $GETMAIN to obtain additional storage

Mapping macros normally required

$PCE, $RDWORK, $JCT, $JCTX, $HCT, $BUFFER, $MIT, $HASPEQU, $JRW

Point of processing

Installation Exit 2 can be invoked when JES2 encounters either:
- the JOB statement, this is called the initial job statement image.
- or a continuation of the JOB statement, this is called an additional JOB continuation statement image.

Module HASPRDR invokes installation Exit 2 for initial JOB statement images. Input service has obtained and initialized the job control table (JCT) and the IOT before calling installation Exit 2. After performing the processing you coded in Exit 2, input services complete scanning the JOB statement and allocate spool space for the job.

Module HASPRDR invokes installation Exit 2 for continuation JOB statement images.

Extending the JCT control block

1. You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (2, 3, 4, and 20) and the $QMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3K. You can have up to 8K of local extension regardless of SPOOL buffer size.
2. If you need to change the scheduling environment, use the JCTSCHEN field in the JCT.
Programming considerations

1. Be aware that when a JOB card image is passed to Exit 2, any /* comment cards embedded within that statement are also passed to the exit. For example, all of the following are passed:

   //ABC JOB
   /* COMMENT CARD
   // CLASS=A

   If within a /* comment you embed valid JOB card parameters, there is potential to cause confusion in your scan routine and lead to unpredictable results. Consider the following:

   /* CHANGED CLASS FROM ORIGINAL CLASS=B

2. When this exit adds or modifies cards, whether the change is sent over NJE (including SPOOL offload) depends on the statement type and the setting of option flags in the $XPL or $RJCB. Modified JECL cards (original and modified card are both JECL) are not sent over NJE. By default, all other changes are sent over NJE. To limit changes to only the local node, you can set the X002RLOC in the XPL (affects the current card) or set the RJCBB3LOC bit in any RJCBS that are added.

3. Updating the statement buffer is only valid for parameters that have $STMNTABs in HASCSRIP.

4. Updates to the statement buffer are not passed to the converter and will not be seen by Exit 6.

Register contents on entry to exit 2

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td><strong>Field Name</strong></td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>XPLEXLEV</td>
</tr>
<tr>
<td></td>
<td>X002IND</td>
</tr>
<tr>
<td></td>
<td>X002JOBC</td>
</tr>
<tr>
<td></td>
<td>X002COND</td>
</tr>
<tr>
<td></td>
<td>X002CONT</td>
</tr>
<tr>
<td></td>
<td>X002SEC</td>
</tr>
<tr>
<td></td>
<td>X002RESP</td>
</tr>
<tr>
<td>X002XSNC</td>
<td>Exit supplied next card in X002JXWR</td>
</tr>
<tr>
<td>X002XSEM</td>
<td>Exit supplied error message in X002JXWR</td>
</tr>
<tr>
<td>X002JCMT</td>
<td>Skip processing card</td>
</tr>
</tbody>
</table>
Exit 2

X002KILL  Kill current job (queue job to OUTPUT processing)
X002PURG  Purge current job
X002RLOC  Changed or added cards are not sent through NJE (set RJCB3LOC in current RJCB)

XPLSIZE  Size of parameter list, including base section
X002CARD  80-byte card image address
X002FLGX  Pointer to exit flags (same as JRWFLAGX)
X002JXWR  80-byte exit work area address (same as JCTXWRK)
X002JCT  JCT address
X002JQE  JQE address
X002AREA  JRW address
X002STMT  Concatenated statement buffer. This is all the operands on all continuations cards for this statement
X002STME  End of statement+1 pointer (in buffer)
X002STML  Statement label (job name)
X002STMV  Statement verb (JOB)
X002RJCP  RJCBS to add before this JOB statement
X002RJCA  RJCBS to add after this JOB statement
X002RJCC  RJCBS to add after the current card
X002FLG1  Statement flag byte
X002LOPR  Last operand is on the current card
X002QUOT  Unfinished quote at end of current card
X002CCMT  Current card is a continued comment
X002LAST  Last card in job statement
X002OCLS  Override job class (batch jobs only)

1  Address of a 3-word parameter list with the following structure:
   Word 1 (+0) points to the JOB statement image buffer
   Word 2 (+4) points to the exit flag byte, JRWFLAGX, in the $JRW
   Word 3 (+8) points to the JCTXWRK field in the $JCT

2-9  Not applicable
10  Address of the $JCT
11  Address of the HCT
12  Not applicable
13  Address of the PCE
14  Return address
15  Entry address

Register contents when exit 2 passes control back to JES2

Upon return from this exit, the register contents must be:
0-13  Not applicable
14  Return address
15 Return code

A return code of:

0 Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit, continue with normal HASPRDR processing.

4 Tells JES2 to ignore any additional exit routines associated with this exit and to continue with normal HASPRDR processing.

8 Tells JES2 to cancel the job; output (the incomplete JCL images listing) is produced.

12 Tells JES2 to purge the job; no output is produced.

Note: If register 10 contains 0 (the JCT is unavailable), JES2 ignores any return code greater than 4.

Coded example

Module HASX02A in SYS1.SHASSAMP contains a sample of exit 2.
Exit 3: JOB statement accounting field scan (JES2 main task)

Function

This exit allows you to provide an exit routine for scanning the JOB statement accounting field and for setting the corresponding fields in the appropriate JES2 control blocks. Exit 3 get control for jobs submitted through card readers, RJE, SNA and BSC NJE and SPOOL reload. For jobs submitted through internal readers or TCP/IP NJE exit 53 is called to JOB statement accounting field.

You can use your exit routine to interpret the variables in the accounting field and, based on this interpretation, decide whether to cancel the job.

Use this exit to record alterations to the accounting field; they will not appear on the user’s output but are reflected in the JCT and when the SMF type 6 record is written.

This exit is associated with the existing HASPRSCN accounting field scan subroutine. You can write your exit routine as a replacement for HASPRSCN or you can use a return code to input processing to call HASPRSCN after your exit routine has executed. In either case, when this exit is implemented and enabled, JES2 treats your exit routine as the functional equivalent of HASPRSCN. The specification of the ACCTFLD parameter on the JOBDEF initialization statement, which normally determines whether JES2 is to call HASPRSCN, becomes an additional factor in determining whether your exit routine is to be called. The exit is taken only if the ACCTFLD= parameter on the JOBDEF initialization statement is specified as either REQUIRED or OPTIONAL. The exit is not taken if ACCTFLD=IGNORE is specified. When it is called, your exit routine, rather than the ACCTFLD parameter, determines whether HASPRSCN is to be executed as an additional scan of the accounting field. For a complete explanation of how the ACCTFLD parameter is specified, see [z/OS JES2 Initialization and Tuning Reference] The relationship of HASPRSCN to this exit is described in greater detail in the “Other Programming Considerations” below.

You can use this exit for input processing - Accounting field.

Related exits

Use Exit 2 to alter the accounting information and supply new accounting information at the time the entire JOB statement is first scanned.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places Exit 3 in supervisor state and PSW key 1.
Exit 3

Restrictions

See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.

Recovery

$ESTAE recovery is in effect. Input processing recovery routine will attempt to recover from program check errors, including program check errors in the exit routine. However, as with every exit, your exit routine for this exit should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine. Therefore, it can provide no more than minimal recovery. You should provide your own recovery within your exit routine.

Job exit mask

Exit 3 is subject to suppression. You can suppress Exit 3 by either implementing exit 2 to set the 3rd bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$PCE, $RDRWORK, $JCT, $JCTX, $HCT, $MIT, $BUFFER, $HASPEQU, $JRW

Point of processing

This exit is taken from the JES2 main task, from the JOB statement processing routine of HASPRDR. The exit occurs after JES2 has scanned the entire JOB statement, but before the execution of the HASPRSCN accounting field scan subroutine, if HASPRSCN is to be called. The JCT has been initialized with the JES2 and installation defaults; in addition, those fields of the JCT that correspond to JOB statement parameters other than accounting field parameters have been set. The accounting field image is passed in X003ACCT and the length in X003ACTL.

Table 5 lists some of the fields in the JCT that you can modify.

<table>
<thead>
<tr>
<th>Field Name in JCT</th>
<th>Length (Bytes)</th>
<th>Field</th>
<th>Bit</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCTSMFLG</td>
<td>1</td>
<td>SMF Flags</td>
<td>0–1</td>
<td>These bits are not part of the interface</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>If set, IEFUSO exit not taken</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3–4</td>
<td>These bits are not part of the interface</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>If set, no type 6 SMF records produced</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>If set, IEFUJP exit not taken</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>If set, no type 26 SMF record produced</td>
<td>1,2</td>
</tr>
<tr>
<td>JCTJOBFL</td>
<td>1</td>
<td>Job Flags</td>
<td>0</td>
<td>Background job</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>TSO/E (foreground) job</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Started task</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>No job journaling</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>No output</td>
<td>1,2</td>
</tr>
</tbody>
</table>
Table 5. Selected JES2 Job Control Table Fields (continued)

<table>
<thead>
<tr>
<th>Field Name in JCT</th>
<th>Length (Bytes)</th>
<th>Field</th>
<th>Bit</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPRUN=SCAN</td>
<td>5</td>
<td>TYPRUN=COPY</td>
<td>6</td>
<td>1,2,3</td>
<td></td>
</tr>
<tr>
<td>Job restartable</td>
<td>7</td>
<td></td>
<td></td>
<td>1,2,8</td>
<td></td>
</tr>
<tr>
<td>TYPRUN=COPY</td>
<td>6</td>
<td></td>
<td></td>
<td>2,3</td>
<td></td>
</tr>
<tr>
<td>JCTJBOPT</td>
<td>1</td>
<td>Job Options</td>
<td>0</td>
<td>/*PRIORITY card was read and value is in priority field (JCTIPRIO)</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>/*SETUP card was read</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>TYPRUN=HOLD was specified</td>
<td>1,2,4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>No job log for this job</td>
<td>1,2,6,8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Execution batch job</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>The job was read through an internal reader</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>The job was rerun</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>This bit is not part of the interface</td>
<td>–</td>
</tr>
<tr>
<td>JCTJOBID</td>
<td>8</td>
<td></td>
<td></td>
<td>JES2 JOB identifier</td>
<td>–</td>
</tr>
<tr>
<td>JCTJNAME</td>
<td>8</td>
<td></td>
<td></td>
<td>Job name</td>
<td>3</td>
</tr>
<tr>
<td>JCTPNAME</td>
<td>20</td>
<td></td>
<td></td>
<td>Programmer name</td>
<td>3</td>
</tr>
<tr>
<td>JCTMCLAS</td>
<td>1</td>
<td></td>
<td></td>
<td>Message class</td>
<td>1,4</td>
</tr>
<tr>
<td>JCTJCLAS</td>
<td>1</td>
<td></td>
<td></td>
<td>Job class</td>
<td>1,4</td>
</tr>
<tr>
<td>JCTIPRIO</td>
<td>1</td>
<td></td>
<td></td>
<td>Priority</td>
<td>1,5</td>
</tr>
<tr>
<td>JCTROUTE</td>
<td>4</td>
<td>Route code of input device (binary)</td>
<td>4</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>JCTINDEV</td>
<td>8</td>
<td></td>
<td></td>
<td>Input device name</td>
<td>–</td>
</tr>
<tr>
<td>JCTACCTN</td>
<td>4</td>
<td></td>
<td></td>
<td>Account number</td>
<td>1,6</td>
</tr>
<tr>
<td>JCTROOMN</td>
<td>4</td>
<td></td>
<td></td>
<td>Room number</td>
<td>1,6,8</td>
</tr>
<tr>
<td>JCTETIME</td>
<td>4</td>
<td>Estimated real-time job will run</td>
<td>4</td>
<td>1,6,8</td>
<td></td>
</tr>
<tr>
<td>JCTESTLN</td>
<td>4</td>
<td>Estimated count of output lines (in thousands)</td>
<td>4</td>
<td>1,6,8</td>
<td></td>
</tr>
<tr>
<td>JCTESTPU</td>
<td>4</td>
<td>Estimated number of output cards punched</td>
<td>4</td>
<td>1,6,8</td>
<td></td>
</tr>
<tr>
<td>JCTESTBY</td>
<td>4</td>
<td>Estimated number of SYSOUT bytes</td>
<td>4</td>
<td>1,6,8</td>
<td></td>
</tr>
<tr>
<td>JCTESTPG</td>
<td>4</td>
<td>Estimated number of output pages</td>
<td>4</td>
<td>1,6,8</td>
<td></td>
</tr>
<tr>
<td>JCTFORMS</td>
<td>8</td>
<td></td>
<td></td>
<td>Job Forms</td>
<td>1,6,8</td>
</tr>
<tr>
<td>JCTCPYCT</td>
<td>1</td>
<td>Job copy count (binary)</td>
<td>1</td>
<td>1,6,8</td>
<td></td>
</tr>
<tr>
<td>JCTLINCT</td>
<td>1</td>
<td>Lines per page (binary)</td>
<td>1</td>
<td>1,6,8</td>
<td></td>
</tr>
</tbody>
</table>

Exit 3: JOB statement accounting field scan (JES2 main task) 91
Exit 3

Table 5. Selected JES2 Job Control Table Fields (continued)

<table>
<thead>
<tr>
<th>Field Name in JCT</th>
<th>Length (Bytes)</th>
<th>Field</th>
<th>Bit</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCTPROUT</td>
<td>4</td>
<td>Default print routing (binary)</td>
<td></td>
<td>1,7</td>
<td></td>
</tr>
<tr>
<td>JCTPUOUT</td>
<td>4</td>
<td>Default punch routing (binary)</td>
<td></td>
<td>1,7</td>
<td></td>
</tr>
<tr>
<td>JCTPROCN</td>
<td>8</td>
<td>Procedure DD name</td>
<td></td>
<td>1,2,8</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. Can be modified by installation routine.
2. Preset from JOBCLASS(v) initialization statement according to job class
3. Preset from JOB statement
4. From JOB statement, if specified; otherwise according to input device as established at JES2 initialization (for example, in RDR(nn)).
5. Exit 3 can use field JCTIPRIO to force a priority for a job subject to the limitations of the input device’s priority increment and priority limit values. When exit 3 receives control, a value of C’+’ in JCTIPRIO indicates a priority has not been forced by an exit routine. If you want to force a priority in exit 3, set JCTIPRIO to a value between 0 and 15 in the low-order four bits on the field.

Note: Whether you may set field JCTIPRIO and the allowable values depend on the specific exit.
6. Set by the routine (HASPRSCN) used by JES2 to scan the account field of the JOB statement. Exit 3 can specify that JES2 cannot call HASPRSCN.
7. Preset according to an input device initialization parameter (for example RDR(nn) ). If not set at initialization the parameter defaults to the job input source value (LOCAL or RMT(nnnn)). Can be modified by a /*ROUTE statement after the scan exit.
8. Can be modified by a /*JOBPARM statement after the scan exit.

Extending the JCT control block

You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (2, 3, 4, and 20) and the $QMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3K. You can have up to 8K of local extension (regardless of SPOOL buffer size).

Programming considerations

1. The accounting field resides in a 144-byte work area pointed to by X003ACCT in the XPL passed to the exit in register 0.
2. If you need to verify the existence of a JOB rather than a started task (STC) or TSO/E logon, this can be done by comparing the JCTJOBID field to a “J”. The presence of a “J” indicates the existence of a JOB.
3. If you need to change the scheduling environment, use the JCTSCHEN field in the JCT.
4. The ACCTFLD parameter on the JOBDEF statement indicates whether JES2 should scan the accounting field of a JOB statement. For further details concerning the use of the ACCTFLD parameter, see z/OS JES2 Initialization and Tuning Reference.

If the ACCTFLD parameter indicates that the scan is to be performed, and if this exit is implemented and enabled, input processing will call your exit routine to perform the scan. If your exit routine passes a return code of 0 or 4 to JES2, input processing will call the existing HASPRSCN accounting field scan subroutine after your routine has executed. Note that if both routines are to be called, your routine should not duplicate HASPRSCN processing. For example, your routine should not set the fields in the JCT that are set by HASPRSCN. However, if your routine passes a return code of 8 or 12 to JES2, JES2 suppresses execution of HASPRSCN. If the ACCTFLD parameter indicates that the scan is to be performed but the exit is disabled, JES2 calls only HASPRSCN; your exit routine is not called and is not given the opportunity to allow or suppress HASPRSCN execution. If the ACCTFLD parameter indicates that a scan is not to be performed, your exit routine is not called, even if this exit is enabled, and execution of HASPRSCN is also suppressed.

5. The ACCTFLD parameter on the JOBDEF statement indicates whether JES2 will cancel a job if the accounting field on the JOB statement is invalid or if a JCL syntax error has been detected during input processing. Note that your exit routine can affect this termination processing. For example, ACCTFLD=REQUIRED indicates that JES2 should scan the accounting field, that the job should be canceled if the accounting field is invalid, and that the job should be canceled if a JCL syntax error has been found. If you pass a return code of 8 to JES2, HASPRSCN is not called, therefore cannot terminate a job with an invalid accounting field, even though ACCTFLD=REQUIRED. Also note that HASPRSCN scans the accounting field passed in X003ACCT. Therefore, if your routine alters this field, you affect HASPRSCN processing.

6. The specification of the ACCTFLD parameter is stored in the HCT, in field $RJOBOPT. If your exit routine is meant to completely replace HASPRSCN, you may want to access this field for use by your algorithm.

7. Typically, use this exit, rather than Exit 2, to alter the JCT directly. If you use Exit 2 to alter the JCT, later processing might override your changes. The job exit mask and the spool partitioning mask are exceptions. See note 2 of Exit 2 for more information.

8. An 80-byte work area pointed to by X003JXWR in the XPL is available for use by your routine. If your routine requires additional work space, use the $GETMAIN macro to obtain storage (and the $FREEMAIN macro to return it to the system when your routine has completed).

9. When passing a return code of 12, your exit routine can pass an installation-defined error message to JES2 to be added to the JCL data set rather than the standard error message. To send an error message, generate the message text in your exit routine, move it to area pointed to by X003JXWR, and set the X003XSEM bit in X003RESP to one.

Note: The standard error message, $HASP110, still appears in SYSLOG on this path, in addition to the installation-defined message. However, only the installation message will be placed in the JCL data set and no WTO will be issued for the installation-defined message unless Exit 3 issues the WTO itself.

10. If there is no accounting field on a JOB statement, the length passed by JES2 to the exit routine in X003ACTL is zero. Your exit routine should take this possibility into account.
11. If you intend to use this exit to process nonstandard accounting field parameters, you should either suppress later execution of HASPRSCN or code your exit routine to delete nonstandard parameters before passing control to HASPRSCN. If you do neither, that is, if you allow HASPRSCN to receive the nonstandard parameters, it might cancel the job because of an illegal accounting field (depending on how the ACCTFLD parameter on the JOBDEF statement is specified).

If you change the length of the accounting field, you must reload the length into field JRWACCTL.

12. There are three job class fields (JCTJCLAS, JCTCLASS, and JCTAXCLS) in the JCT. JCTJCLAS is the initial job execution class as set during input processing and used when building the JQE during that processing. JCTCLASS is the actual execution class. After input processing it contains the same value as JCTJCLAS, but it might be updated when the job executes if a $T command was used to update the job’s class before execution. Therefore, JCTJCLAS and JCTCLASS could be different. JCTAXCLS is a copy of the actual execution class (JCTCLASS) that is propagated into the network JOB trailer. Do not use any exit routine to set the JCTAXCLS field.

If you intend to use an exit 3 routine to change the execution class of a job, be certain to set both the JCTJCLAS and JCTCLASS fields.

### Register contents when Exit 3 gets control

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level for base XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit ID number</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version number for exit</td>
</tr>
<tr>
<td>X003IND</td>
<td>Indicator byte</td>
</tr>
<tr>
<td>X003COND</td>
<td>Condition byte</td>
</tr>
<tr>
<td>X003RESP</td>
<td>Response byte</td>
</tr>
<tr>
<td>X003XSEM</td>
<td>Exit supplied error message in X003JXWR</td>
</tr>
<tr>
<td>X003SKIP</td>
<td>Skip default accounting field</td>
</tr>
<tr>
<td>X003KILL</td>
<td>Kill current job (queue job to OUTPUT processing)</td>
</tr>
<tr>
<td>XPLSIZE</td>
<td>Size of parameter list, including base section</td>
</tr>
<tr>
<td>X003ACCT</td>
<td>Address of accounting field</td>
</tr>
<tr>
<td>X003FLGX</td>
<td>Pointer to exit flags (same as JRWFLAGX)</td>
</tr>
<tr>
<td>X003JXWR</td>
<td>80-byte exit work area address (same as JCTXWRK)</td>
</tr>
<tr>
<td>X003JCT</td>
<td>JCT address</td>
</tr>
<tr>
<td>X003QJE</td>
<td>JQE address</td>
</tr>
<tr>
<td>X003AREA</td>
<td>JRW address</td>
</tr>
</tbody>
</table>

1 Address of a 3-fullword parameter list

| Word 1 (+0) | points to the accounting field (JCTWORK in the JCT) |
| Word 2 (+4) | points to the exit flag byte, JRWFLAGX in the JRW |
| Word 3 (+8) | points to the JCTXWRK field in the JCT |

2-10 Not applicable
Exit 3

Register contents when Exit 3 passes control back to JES2

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit, use the current setting of the ACCTFLD parameter on the JOBDEF statement to determine whether to execute the HASPRSCN subroutine.

4  Tells JES2 to ignore any other exit routines associated with this exit and to use the current setting of the ACCTFLD parameter on the JOBDEF statement to determine whether to execute HASPRSCN.

8  Tells JES2 to suppress execution of HASPRSCN and to complete job card processing.

12  Tells JES2 to cancel the job because an illegal accounting field has been detected. Tells JES2 to suppress execution of HASPRSCN and to queue the job for output; output (the incomplete JCL images listing) is produced.

Coded example

Module HASX03A in SYS1.SHASSAMP contains a sample of Exit 3.
Exit 4: JCL and JES2 control statement scan (JES2 main task)

Function

This exit allows you to provide an exit routine for scanning JCL and JES2 control statements for jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload. For jobs submitted through internal readers or TCP/IP NJE, exit 54 is called to process JCL and JES2 control statements (JECL). If this exit is implemented and enabled, it is taken whenever JES2 encounters a JCL or JES2 control statement. (Note: JOB statements are not included in the scan).

For JCL statements, your exit routine can interpret JCL parameters and, based on this interpretation, decide whether JES2 should cancel the job, purge the job, or allow the job to continue normally. Your routine can also alter JCL parameters and supply additional JCL parameters. If necessary, in supplying expanded JCL data, your routine can pass a JCL continuation statement back to JES2 or add statements before or after the current JCL statement.

For JES2 control statements, your routine can interpret the JES2 control parameters and sub-parameters and, based on this interpretation, decide whether JES2 should cancel the job, purge the job, or allow the job to continue normally. For any JES2 control statement, you can write your exit routine as a replacement for the standard JES2 control statement processing, suppressing execution of the standard JES2 scan, or you can perform your own (partial) processing and then allow JES2 to execute the standard control statement routine processing. Also, your routine can alter a JES2 control statement and then pass the modified statement back to JES2 for standard processing, or your routine can pass an entirely new JES2 control statement back to JES2, to be read (and processed) before or after the current control statement.

This exit also allows you to process your own installation-specific JES2 control statements or to implement new, installation-specific sub-parameters for existing JES2 control statements.

This exit gets control when JES2 detects a JES2 control statement or JCL statement within a job. JES2 also gives control to your exit routine when JES2 detects a JES2 control statement or JCL statement outside a job. JES2 also gives control to your exit routine when it detects a JCL continuation statement.

This exit allows you to input processing - JCL/JECL.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 4 in supervisor state and PSW key 1.
Exit 4

Restrictions

JES2 does not invoke this exit for JCL from cataloged procedures. See Appendix A, "JES2 exit usage limitations," on page 373 for other specific instances when this exit is invoked or not invoked.

Recovery

$ESTAE recovery is in effect. The recovery routine established by JES2 attempts to recover from program check errors, including program check errors in the exit routine itself. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine. Therefore, it can provide no more than minimal recovery. Provide your own recovery within your exit routine.

Job exit mask

Exit 4 is subject to suppression. You can suppress exit 4 by either implementing exit 2 to set the 4th bit in the job exit suppression mask (JCTXMASK) or disabling the exit in the JES2 initialization stream.

Mapping macros normally required

$HCT, $JCT, $JCTX, $MIT, $PC.E, $RDRWORK, $BUFFER, $HASPEQU, $JRW

Point of processing

This exit is taken from HASPRDR in the JES2 main task. The exit occurs in the main processing loop of HASPRDR, after HASPRDR has read an entire JES2 control statement or JCL statement (including JCL continuations) but before it has processed any keywords on the statement. The statement may be outside a valid job (that is, when there is no current job structure active on the reader).

This exit is invoked for jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload. It is not invoked for jobs submitted through the internal reader or TCP/IP NJE.

Programming considerations

1. This exit is taken once for each control statement (except for JOB statements) encountered by JES2. X004IND indicates whether the current statement is a JCL statement or a JES2 control statement. Your exit routine gets control for /* comment, /* (generated), and */ PRIORITY JES2 control statements.

2. During input processing, JES2 writes the JCL records to a JCL data set. If an error occurs during input processing, it is the JCL data set that is printed when the job goes through output processing. If the job is successfully processed by input processing, the JCL data set is the input for the converter. The converter produces a JCL images data set that is printed when the job goes to output processing after being successfully processed by input processing.

3. Exit 4 is called for each card in a JCL statement (the original card and all continuations) and for each JES2 control statement. Each time the exit is called, it is passed the current card image and the statement buffer. The statement buffer is all the operands for the JCL statement or JES2 control statement concatenated in a single buffer. For example:

   //OUTSET DD SYSOUT=H,OUTPUT=*.OUT1, COMMENT1
   // DCB=(LRECL=8000,RECFM=FB,BLKSIZE=8000) COMMENT2

   In this case the exit will be called 2 times, once for each card and will be
passed (on both calls) the following data in the statement buffer (pointed to by X004STMT):

SYSOUT=H,OUTPUT=*.OUT1,DCB=(LRECL=8000,RECFM=FB,BLKSIZE=8000)

To alter the processing of the JCL statement or JES2 control card, the exit can:

- Update the card image passed in X004CARD. This change shows up in the listing of the job.
- Update the statement buffer in X004STMT to add or modify the operands. This change does not show up in the listing of the job and is not passed to conversion processing (it only affects keywords input processing scans from the JOB card). If you update the statement buffer (X004STMT) in Exit 4 and change the length of the buffer, you must update the field X004STME to indicate the new end of buffer (one byte past the last meaningful character).
- Add additional card images to the JCL stream

You can add card images to the JCL stream by either queuing a single RJCB or a chain of RJCBs to the XPL, or by placing a card image to be placed after the current card into the area pointed to by X004JXWR and setting X004XSNC. In either case, when a card is added, the current card is re-scanned and the statement buffer is re-built. Exit 4 is driven again for the updated statement, with X004SEC set to indicate this card has been presented to the exit previously.

When adding cards using RJCBs, use the RGETRJCB service (located in HASCSRIP) to obtain a free RJCB; then add it to one of the three RJCB queues in the XPL. Use the $CALL macro to invoke the RGETRJCB service. Register 1 on entry must be the JRW address. The RJCB address is returned in register 1.

The 80-byte card image to be added is placed into the field RJCBCARD. RJCBs are chained together using the RJCBRJCB field in the $RJCB. They are added to the job stream in the order they exist in the chain. To add an element to the chain you would move the current RJCB queue head in the $XPL into the RJCBRJCB field of the last RJCB you are adding, and then set the address of the first RJCB element into the $XPL queue head. Be aware that multiple exit 4s might be using these queues, so ensure that you do not lose existing entries on the queue.

**X004RJCP**

Adds the card images before the first card in the current JCL statement or before the JES2 control card.

**X004RJCA**

Adds the card images after the last card in the current JCL statement. In this case, the card(s) are assumed to not be a continuation of the current JCL statement and the JCL cards are not re-scanned.

**X004RJCC**

Adds the card images after the current card. It is the callers' responsibility to ensure that the proper continuation processing will occur.

When processing the last card in a JCL statement or when processing a JES2 control statement, the difference between adding a card to the X004RJCA queue and the X004RJCC queue is that the first will not rescan the current statement and the second will.

Add the card images after the current card. It is the callers' responsibility to ensure that the proper continuation processing will occur.
Exit 4

a. Move a comma into the last byte of the operand on the JCL card image (X004CARD) that exit 4 is currently processing. The comma indicates additional information follows this JCL statement.

b. Move the information you want to add to the JCL statement to the area pointed to by X004JXWR and set the X004XSNC bit in the X004RESP byte to one. Setting X004RESP to X004XSNC indicates that the installation has supplied an additional JCL statement image.

c. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the statement.

You can also add an additional JCL statement to the job as follows:

a. Ensure that the JCL card image that exit 4 is currently processing is the last for the current statement (X004LOPR is on). Exit 4 is processing the last JCL statement image if a comma is not in the last byte of the JCL operand on the card image.

b. Place the JCL statement in the area pointed to by X004JXWR and set the X004XSNC bit in the X004RESP byte to one. Setting X004RESP to X004XSNC indicates that the installation has supplied an additional JCL statement image.

c. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the JCL or JECL card.

For JECL statements, because there are no formal rules for the format of the statement, the statement buffer will contain all the text after the VERB on the JECL statement. The following is an example of a JOBPARM JECL statement and the associated statement buffer:

/*JOBPARM SYSAFF=(IBM1),COPIES=2 This is a comment

The statement buffer for this statement would contain:

SYSAFF=(IBM1),COPIES=2 This is a comment

The statement buffer contains the comment in this case (and any trailing blanks) because there is no formal rule stating where a JECL statement ends.

4. Updating the statement buffer is only valid for parameters that have $STMTTABs in HASCSRIP.

5. Updates to the statement buffer are not passed to the converter and will not be seen by Exit 6.

6. The following indicators in the XPL can assist you in adding a card image to the current JCL statement:

X004LOPR
Current card has the last operand in the JCL statement. There can be additional continued comments after the current card.

X004QUOT
A quoted string is being continued from the current card to the next card. Pay attention if a card is being added after this card.

X004CCMT
The current card is a continued comment. Operand added to this card or after this card will not be processed.

X004LAST
This is the last card image in the JCL or JECL statement.
7. To assist you in processing the operands on a statement, you can use either of the following services to parse the statement buffer passed in X004STMT:
   - Use the $SCAN facility to parse the operands with the standard $SCAN rules for statements. This give you the flexibility of $SCAN but the parsing rules are not the same as normal JCL. See the $SCAN and $SCANTAB macros for additional information.
   - Use the RCARDSCN service and $STMTTAB macro to parse the operands with standard JCL rules. This is the service used by JES2 input processing to parse the statement buffer. However, the RCARDSCN service only parses the operands and calls a processing routine to do all the conversions and storing of data. Conversion of data to binary to store into data areas is the responsibility of the processing routines. See the $STMTTAB macro for more information.

8. To entirely replace standard JES2 control card processing (HASPRCCS) for a particular JES2 control statement, write your routine as a replacement version of the standard HASPRCCS routine; then pass a return code of 8 back to JES2 to suppress standard processing. Note that your routine becomes responsible for duplicating any HASPRCCS function you want to retain. If you merely want to supplement standard HASPRCCS processing, you can write your exit routine to perform the additional function and then, by passing a return code of 0 or 4, direct JES2 to execute the standard HASPRCCS routine.

9. To nullify a JES2 control statement, pass a return code of 8 to JES2 without using your exit routine to perform the function requested by the statement. Note that, based on what appears in the JCL images output data set, the user is not informed that the statement was nullified.

10. To modify a JES2 control statement, also use return code 8. Place the altered statement in the area pointed to by X004JXWR and set X004XSNC to one. If input processing is successful, the user will see in the output of the JCL images file the original statement, and the altered statement. Note, that if you modify a JES2 control statement and then pass a return code of 0 or 4, JES2 carries out normal input (HASPRCCS) processing, and the modified version of the statement will appear on the user’s output in the JCL images file, but the original statement will not appear unless you go directly to output phase (bypassing the converter); then, the user will see the original statement when the JCL data set is printed.

11. Also use return code 8 in processing your own installation-specific JES2 control statements. Write your exit routine to perform the function requested by the statement and then pass return code 8 to JES2 to suppress standard processing and thereby prevent JES2 from detecting the statement as “illegal.”

12. **Extending the JCT Control Block**
   You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (2, 3, 4, and 20) and the $QMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3k. You can have up to 8K of local extension regardless of SPOOL buffer size.

13. To process your own installation-specific JES2 control statement subparameters, you should generally write your exit routine to replace standard HASPRCCS processing entirely. That is, write your exit routine to perform the function(s) requested by the standard parameters and subparameters and those requested by any unique installation-defined subparameters on a statement. Then, from your exit pass a return code of 8 back to JES2.
Exit 4

Typically, because the parameters and subparameters on a JES2 control statement are interdependent, you will be limited to this method. However, if you have defined an installation-specific subparameter which can be processed independently of the rest of the control statement on which it appears, you can write your exit routine to process this subparameter alone, delete it, and pass a return code of 0 or 4 to JES2. JES2 can then process the remainder of the statement as a standard JES2 control statement.

14. When passing a return code of 12 or 16, it is also possible for your exit routine to pass an error message to JES2 for display at the operator’s console. To send an error message, generate the message text in your exit routine, move it to the area pointed to by X004JXWR, and set the X004XSEM bit in X004RESP to one.

15. If you intend to use this exit to affect the JCT, your exit routine must ensure the existence of the JCT on receiving control. If the JCT has not been created when your exit routine receives control, the pointer to X004JXWR is zero. For example, when your exit routine receives control for a /*PRIORITY statement, the JCT doesn’t exist yet. In this case, your routine must store any data to be placed in the JCT until JES2 creates the JCT.

16. Your exit routine does not have access to the previous control card image. You should take this into account when devising your algorithm.

17. An 80-byte work area, pointed to by X004JXWR, is available for use by your exit routine. If your routine requires additional work space, use the $GETMAIN macro to obtain storage (and the $FREMAIN macro to return it to the system when your routine has completed).

18. Exit 4 can use field JCTIPRIO to force a priority for a job subject to the limitations of the input device’s priority increment and priority limit values. When exit 4 receives control, a value of C’‘’ in JCTIPRIO indicates a priority has not been forced by an exit routine. If you want to force a priority in exit 4, set JCTIPRIO to a value between 0 and 15 in the low-order four bits on the field.

Note: Whether you may set field JCTIPRIO and the allowable values depend on the specific exit.

19. When this exit adds or modifies cards, whether the change is sent over NJE (including SPOOL offload) depends on the statement type and the setting of option flags in the $XPL or $RJCB. Modified JECL cards (original and modified card are both JECL) are not sent over NJE. By default, all other changes are sent over NJE. To limit changes to only the local node, you can set the X004RLOC in the XPL (affects the current card) or set the RJCB3LOC bit in any RJCBs that are added.

Register contents when Exit 4 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td><strong>Field Name</strong></td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>XPARLXLEV</td>
</tr>
<tr>
<td></td>
<td>X004IND</td>
</tr>
</tbody>
</table>
### Exit 4: JCL and JES2 control statement scan (JES2 main task)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>JCL card detected</td>
</tr>
<tr>
<td>04</td>
<td>JECL card detected</td>
</tr>
<tr>
<td><strong>X004COND</strong></td>
<td>Condition byte</td>
</tr>
<tr>
<td><strong>X004CONT</strong></td>
<td>Card is a continuation (not first card of statement)</td>
</tr>
<tr>
<td><strong>X004JOBP</strong></td>
<td>/*JOBPARM card detected</td>
</tr>
<tr>
<td><strong>X004CMND</strong></td>
<td>/*$ command card detected</td>
</tr>
<tr>
<td><strong>X004SEC</strong></td>
<td>This card has been passed to the exit previously for this job (set if cards added before this card)</td>
</tr>
<tr>
<td><strong>X004PREJ</strong></td>
<td>Card encountered outside a job structure</td>
</tr>
<tr>
<td><strong>X004RESP</strong></td>
<td>Response byte</td>
</tr>
<tr>
<td><strong>X004XSNC</strong></td>
<td>Exit supplied next card in X004JXWR</td>
</tr>
<tr>
<td><strong>X004XSEM</strong></td>
<td>Exit supplied error message in X004JXWR</td>
</tr>
<tr>
<td><strong>X004JCMT</strong></td>
<td>Skip processing card</td>
</tr>
<tr>
<td><strong>X004KILL</strong></td>
<td>Kill current job (queue job to OUTPUT processing)</td>
</tr>
<tr>
<td><strong>X004PURG</strong></td>
<td>Purge current job</td>
</tr>
<tr>
<td><strong>X004RLOC</strong></td>
<td>Changed or added cards are not sent through NJE (set RJC3LOC in current RJC3)</td>
</tr>
<tr>
<td><strong>XPLSIZE</strong></td>
<td>Size of parameter list, including base section</td>
</tr>
<tr>
<td><strong>X004CARD</strong></td>
<td>80-byte card image address</td>
</tr>
<tr>
<td><strong>X004FLGX</strong></td>
<td>Pointer to exit flags (same as JRWFLAGX)</td>
</tr>
<tr>
<td><strong>X004JXWR</strong></td>
<td>80-byte exit work area address (same as JCTXWRK)</td>
</tr>
<tr>
<td><strong>X004JCT</strong></td>
<td>JCT address</td>
</tr>
<tr>
<td><strong>X004JQE</strong></td>
<td>JQE address</td>
</tr>
<tr>
<td><strong>X004AREA</strong></td>
<td>JRW address</td>
</tr>
<tr>
<td><strong>X004STMT</strong></td>
<td>Concatenated statement buffer. This is all the operands on all continuations cards for this statement</td>
</tr>
<tr>
<td><strong>X004STME</strong></td>
<td>End of statement+1 pointer (in buffer)</td>
</tr>
<tr>
<td><strong>X004STML</strong></td>
<td>Statement label</td>
</tr>
<tr>
<td><strong>X004STMV</strong></td>
<td>Statement verb</td>
</tr>
<tr>
<td><strong>X004RJCP</strong></td>
<td>RJC3Bs to add before the current JCL/JECL statement</td>
</tr>
<tr>
<td><strong>X004RJCA</strong></td>
<td>RJC3Bs to add after the current JCL/JECL statement</td>
</tr>
<tr>
<td><strong>X004RJCC</strong></td>
<td>RJC3Bs to add after the current card</td>
</tr>
<tr>
<td><strong>X004FLG1</strong></td>
<td>Statement flag byte</td>
</tr>
<tr>
<td><strong>X004LOPR</strong></td>
<td>Last operand is on the current card</td>
</tr>
<tr>
<td><strong>X004QUOT</strong></td>
<td>Unfinished quote at end of current card</td>
</tr>
<tr>
<td><strong>X004CCMT</strong></td>
<td>Current card is a continued comment</td>
</tr>
</tbody>
</table>
Exit 4

X004LAST  Last card in JCL or JECL statement

1   Pointer to a 3-word parameter list with the following structure:
    Word 1  (+0) address of the control statement image buffer
    Word 2  (+4) address of the exit flag byte, JRWFLAGX, in
             the $JRW
    Word 3  (+8) address of the JCTXWRK field in the $JCT

2-10  Not applicable
11   Address of the HCT
12   Not applicable
13   Address of the PCE
14   Return address
15   Entry address

Register contents when Exit 4 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if any additional exit routines are associated with
    this exit, call the next consecutive exit routine. If there are no
    additional exit routines associated with this exit, perform standard
    input processing.

4  Tells JES2 to ignore any other exit routines associated with this exit
    and to perform standard input processing.

8  For JES2 control statements and JCL statements, tells JES2 not to
    perform standard processing and just write the statement to the
    JCL data set.

12  Tells JES2 to cancel the job because an illegal control statement
    has been detected; output (the incomplete JCL images listing) is
    produced.

16  Tells JES2 to purge the job because an illegal control statement
    has been detected; no output is produced.

Note: For all JES2 control statements preceding the JOB card (X004PREJ on), a
return code higher than 4 is ignored.

Coded example

Module HASX04A in SYS1.SHASSAMP contains a sample of Exit 4.
Exit 5: JES2 command preprocessor

Function

This exit allows you to preprocess most JES2 commands. If this exit is implemented and enabled, all but the following commands are available for preprocessing.

- $Mnn
- $Nnnnn
- SP JES2,ABEND,FORCE
- $T CKPTDEF,RECONFIG=YES
- Monitor commands –
  - $JD DETAILS
  - $JD HISTORY
  - $JD JES
  - $JD MONITOR
  - $JD STATUS
  - $J STOP

You can use your exit routine to perform your own command validation and, based on the checking performed by your validation algorithm, decide whether JES2 should terminate processing for the command or allow normal JES2 command processing to continue. If you use your exit routine to terminate processing for a command, the command subprocessor is bypassed and the requested action is not taken.

This exit also permits you to implement your own installation-specific JES2 command operands and suboperands, and nonstandard JES2 commands unique to your installation. Your exit routine must process nonstandard, installation-specific operands, suboperands, and commands itself, and then suppress standard JES2 command processing. Nonstandard command processing is considered in greater detail in the “Other Programming Considerations” below.

When suppressing standard JES2 command processing, you have the option of directing JES2 to send the standard “OK” return message to the operator, sending your own exit-generated message to the operator, or of suppressing standard JES2 command processing without operator notification.

Macro $CFSEL can help you process command operand strings.

The JES2 command translator migration aid:

JES2 provided a compatibility and migration aid in the form of an automatically invoked Exit 5 routine in OS/390 Version 2 Release 4 and up. However, this exit 5 command translation routine is no longer automatically loaded and enabled as of z/OS V1R2. The command translation module, HASX05C, is shipped (unchanged) in SYS1.SHASSAMP as of z/OS V1R2.

IBM suggests that you use the most current command syntax. However, if this is not possible, install the JES2 command translation exit (member HASX05C in SYS1.SHASSAMP). On the next JES2 restart, supply the following initialization statements:

```
LOAD(HASX05C)
EXIT(5) ROUTINES=(HASX5CTR)
```
If additional EXIT(5) statements are found in the initialization stream, they will override this default. To include the translation function in this case, HASX5CTR should be added to the list of routines on the EXIT(5) statement.

The following table lists those commands translated by the exit routine:

**Table 6. Old/New Comparison of JES2 Commands**

<table>
<thead>
<tr>
<th>Pre-HJE6604 Format</th>
<th>Translated Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D'name',...</td>
<td>$DJOBQ'name',CMDAUTH=',...</td>
</tr>
<tr>
<td>$T'name',...</td>
<td>$TJOBQ'name',...</td>
</tr>
<tr>
<td>NOTE: Similar for $A, $C, $E, $H, $L, $O, $P, $T, $TO</td>
<td></td>
</tr>
<tr>
<td>$DJ1,2,...</td>
<td>$DJ(1, 2),...</td>
</tr>
<tr>
<td>NOTE: J can be J, JOB, S, STC, T, TSU.</td>
<td></td>
</tr>
<tr>
<td>NOTE: Similar for $a, $C, $E, $H, $L, $O, $P, $TO</td>
<td></td>
</tr>
<tr>
<td>$DJ1–2, J3–4,...</td>
<td>$DJ(1–2, 3–4)...</td>
</tr>
<tr>
<td>NOTE: Similar for $A, $C, $E, $H, $L, $O, $P, $TO</td>
<td></td>
</tr>
<tr>
<td>$LJnnn,ALL</td>
<td>$DOJnnn</td>
</tr>
<tr>
<td>$LJnnn,H</td>
<td>$DOJnnn,HELD</td>
</tr>
<tr>
<td>$LJnnn,READY</td>
<td>$DOJnnn,READY</td>
</tr>
<tr>
<td>$LJnnn,OUTGRP=xxx</td>
<td>$DOJnnn,OUTGRP=xxx</td>
</tr>
<tr>
<td>$CJnnn,OUTGRP=xxx</td>
<td>$COJnnn,OUTGRP=xxx</td>
</tr>
<tr>
<td>$PJnnn,OUTGRP=xxx</td>
<td>$POJnnn,OUTGRP=xxx</td>
</tr>
<tr>
<td>$PJnnn,Q=x</td>
<td>$POJnnn, Q=x Unless Q= is a valid job queue (XEQ, PPU, etc.)</td>
</tr>
<tr>
<td>$vJnnnn,A=</td>
<td>DAYS=</td>
</tr>
<tr>
<td>$TJnnnn,S=(sid1, sid2,...)</td>
<td>$TJnnnn,S=(sid1, sid2,...)</td>
</tr>
<tr>
<td>$DSPL,JOBS=nn</td>
<td>$DJOBQ,SPOOL=(PERCENT&gt;=nn)</td>
</tr>
<tr>
<td>$DSPL,V=xxxxxx, JOBS=nn</td>
<td>$DJOBQ,SPOOL=(PERCENT&gt;=nn, VOLUME=xxxxxx)</td>
</tr>
<tr>
<td>$SSPL,V=xxxxxx,...</td>
<td>$SSPL(xxxxxxx),...</td>
</tr>
<tr>
<td>$vIxx</td>
<td>$vI(xx)</td>
</tr>
<tr>
<td>$TIxx,class-list</td>
<td>$TI(xx),C=class-list</td>
</tr>
<tr>
<td>$HQ,ALL</td>
<td>$TJOBCLASS(*),QHELD=Y</td>
</tr>
<tr>
<td>$HQ,C=xyz</td>
<td>$TJOBCLASS(x,y,z),QHELD=Y</td>
</tr>
<tr>
<td>$AQ,ALL</td>
<td>$TJOBCLASS(*),QHELD=N</td>
</tr>
<tr>
<td>$AQ,C=xyz</td>
<td>$TJOBCLASS(x,y,z),QHELD=N</td>
</tr>
<tr>
<td>$PQ,ALL,...</td>
<td>$POJOBQ,READY,...</td>
</tr>
<tr>
<td>$PQ,Q=xyz,....</td>
<td>$POJOBQ,READY,Q=XYZ,....</td>
</tr>
<tr>
<td>$QOQ,ALL,...</td>
<td>$QJOBQ,R=LOCAL,*,...</td>
</tr>
<tr>
<td>$QOQ,Q=xyz,....</td>
<td>$QJOBQ,R=LOCAL,*,/Q=xyz,....</td>
</tr>
<tr>
<td>$TALL,sid1,sid2,...</td>
<td>$TJOBQ(*),/S=(sid1),/Q=(sid2,...)</td>
</tr>
<tr>
<td>$LSYS</td>
<td>$DMEMBER</td>
</tr>
<tr>
<td>$ESYS,sid</td>
<td>$EMEMBER(sid)</td>
</tr>
<tr>
<td>$ESYS,RESET=sid</td>
<td>$ECKPLOCK,HELDBY=sid</td>
</tr>
</tbody>
</table>
Table 6. Old/New Comparison of JES2 Commands (continued)

<table>
<thead>
<tr>
<th>Pre-HJE6604 Format</th>
<th>Translated Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TSYS,IND=Y/N</td>
<td>$TMEMBER(local),IND=Y/N</td>
</tr>
</tbody>
</table>

Note: For ease of coding, some commands which work without translation may be translated to an equivalent form. For example, RDJ1 is translated to $DJ(1).

For further information about this pre-R4 to post-R4 migration aid, see the Exit 5 documentation in the z/OS Migration document for the release that you are migrating from.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 5 in supervisor state and PSW key 1.

Recovery

$ESTAE recovery is not in effect while an exit routine associated with this exit is being processed. However, you can implement $ESTAE recovery within your routine. As with all exits, you are responsible for your own recovery within your exit routine, whether you choose to implement $ESTAE recovery or other recovery procedures.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $MIT, $PCE, $COMWORK

Point of processing

This exit is taken from the JES2 main task, from the HASPCCOME command edit routine of HASPCOMM. The exit point occurs after the command has been edited but before lookup in the command selection tables (COMFASTR and COMTAB), before console authority checking, and before the call to the command subprocessor.

If your exit routine processes the command, the exit routine is responsible for performing any necessary security validation or auditing. Also, if your exit routine sets a return code of 8 or greater, auditing will not occur. If you want to audit commands that your exit routine would fail, you must call SAF in your exit routine to perform the auditing.
Programming considerations

1. For a multiple command, this exit is taken once for each command verb.

2. The same command can be presented to Exit 5 on multiple members of the MAS. If the command is operating on a job executing on a different member than where the command originated, JES2 will send the command to the target system where it will be reissued. Therefore, to distinguish between the original command and a reissued command your exit must check the contents of the COMFLAG3 field of the PCE pointed to by register 13. If the CMB3INTC bit is on, the command is a reissued command.

   It is recommended that one member be chosen to process the command, and ignore the command on the other members.

3. To preprocess a standard JES2 command, a typical exit routine would perform some type of validation checking. This validation checking would determine whether JES2 should terminate command processing or allow standard command processing to continue. You can base a validation algorithm on various factors. The fields of the command processor work area of the PCE contain extensive command-related information that can be used in validation checking. Note, however, that even if your exit routine validates a command, it is still possible for JES2 to reject the command based on its standard validation checking.

4. In processing your own installation-specific JES2 commands, your exit routine should perform its own validation checking to replace the functions normally performed by HASPCOME. Your routine should validate the command verb, contained in the COMVERB field of the PCE’s command processor work area, with the equivalent of the command table lookup performed by HASPCOME. This check should determine whether the command has a valid installation-specific command verb and what action your exit routine should take based on the verb. Your routine should also perform console authority checking by testing the COMAUTH field, of the PCE’s command processor work area, which contains the command’s restriction bits. COMAUTH has the following structure:

   | COMS          | (X’01’) when on indicates that the command should be rejected unless authorized for the system. |
   | COMD          | (X’02’) when on indicates that the command should be rejected unless authorized for the device. |
   | COMJ          | (X’04’) when on indicates that the command should be rejected unless authorized for the job. |
   | COMR          | (X’08’) when on indicates that the command should be rejected if it was entered from a remote work station. |

   If your routine validates the command, it can then perform the requested function, serving as the equivalent to a standard command subprocessor. If, however, your routine determines that the command is not valid, it must terminate processing for the command internally before returning control to JES2. Then, it should pass a return code (of 8, 12, or 16) to terminate standard HASPCOMM processing, with or without an accompanying message to the operator.

5. When issuing job-related messages, IBM suggests that you have a $CWTO for a control line if you also specify a console area (L=area). Issue job-related messages independently from any other messages in your exit; do not include JOB= or LAST=. Because JES2 inserts the message identifier and a time stamp, your message should not exceed 16 characters.
There is only one control line for a multi-line WTO, and the remaining lines (referred to as data lines) cannot exceed 70 characters in length.

When you have issued any job-related messages, you can then issue all remaining messages. Structure your logic to reduce dependencies on whether a console area is specified. Use the following guidelines:

- Assume JES2 issues each single-line and multi-line message independently, that is, as if no console area was specified.
  - Code LAST=YES on a $CWTO for a single-line message. Keep in mind the message isn't really a single line if a console area was specified and JES2 ignores LAST=YES.
  - Code LAST=NO on the first and middle lines and LAST=YES on the last line of multi-line messages.

- If you code JOB=YES on a multi-line message, code it for each line of that message. For a single or multi-line message with JOB=YES, place the 8-character JOBID followed by a blank in the first nine characters of the message text of the first or only message line. If a console area wasn't specified, JES2 removes the JOBID from the message text, shifts the remaining text to the left, and issues a WTO with the specified JOBID. If you are issuing a multi-line message, place nine blanks at the beginning of the text of all subsequent lines.

- Observe the following line length restrictions to reduce dependencies on whether an area was specified:
  - Place only the JOBID and job name on the first line of a job-related, multi-line message and not more than 25 characters on the first line of a non-job-related, multi-line message.
  - If JOB=YES, limit the length of subsequent message lines to 61 characters.
  - If JOB=NO, limit the length of subsequent message lines to 70 characters.

See z/OS JES2 Macros for more information about the use of the $CWTO macro.

6. Typically, to process nonstandard operands and suboperands, you must write your exit routine to replace standard JES2 processing entirely. That is, your exit routine must process both the nonstandard operands or suboperands and the standard portion of the command, by performing the function of the standard command subprocessor. This is typically because the command verb and the accompanying operands and suboperands are interdependent; the operands and suboperands modify the action of the command verb and cannot be processed independently.

7. When passing a return code of 16 and issuing an exit-generated message to the operator, move the text of the message to the COMMAND field of the command processor work area in the PCE. Place the length of the message in R0. Also, be certain to issue the $STORE (R0) macro after loading the message length in R0 but before issuing the $RETURN macro because $RETURN macro destroys the contents on register 0. (When passing a return code of 12, to cause JES2 to issue the standard “OK” return message, you do not have to supply the message length in R0.)

8. Use the $CWTO macro instruction in this exit to communicate to the operator. If you use the $CWTO macro, you must do all the processing required by the specified command within your exit routine and provide a return code indicating that JES2 should bypass any further processing of the specified command.
Exit 5

If the command being processed is a reissued command (the CMB3INTC bit in the COMFLAG3 field of the PCE pointed to by register 13 is on) the message issued by $CWTO will be displayed in the system log only. See [z/OS JES2 Macros](http://www.ibm.com/support/docview.wss?uid=swg21681009) for more information about the use of the $CWTO macro.

9. When this exit routine operates in a networking environment, your exit must check the contents of the COMGFLG1 flag byte of the PCE pointed to by register 13. If the COMG1SSI bit is on, the current command is in subsystem independent format, and registers 5, 6, and 7 do not contain pertinent information. *(Note: These subsystem-independent commands are also known as formatted commands and can be issued through $G commands.) The structure of the subsystem-independent commands is located at COSICMDA in the mapping macro $COMWORK.*

Register contents when Exit 5 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-4</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>Pointer to the address of the current operand*</td>
</tr>
<tr>
<td>6</td>
<td>Increment value of 4*</td>
</tr>
<tr>
<td>7</td>
<td>Pointer to the address of the last operand*</td>
</tr>
<tr>
<td>8-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of the HASPCOMM PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

*See "Programming Considerations" for use of these registers in a networking environment.

Register contents when Exit 5 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If an exit-generated message is to be passed, this register contains the length of the message; otherwise, it is not applicable.</td>
</tr>
<tr>
<td>1 - 13</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code:

- 0: Tells JES2 that if any additional exit routines are associated with this exit, execute the next consecutive exit routine. If there are no other exit routines associated with this exit, continue with normal command processing.
- 4: Tells JES2 to ignore any other exit routines associated with this exit point and to continue with normal command processing.
- 8: Tells JES2 to terminate standard processing for the command and to issue the $CRET macro to return control to the main command processor; the command subprocessors are bypassed.
Exit 5

12 Tells JES2 to terminate standard processing for the command and to issue the $CRET macro, specifying the standard $HASP000 "OK" message, to return control to the main command processor. The "OK" message is issued and the command subprocessors are bypassed.

16 Tells JES2 to terminate standard processing for the command and to issue the $CRET macro, specifying a message generated by your exit routine, to return control to the main command processor. The exit-generated message is issued and the command subprocessors are bypassed.

Coded example

Modules HASX05A and HASX05C in SYS1.SHASSAMP contain examples of Exit 5.
Exit 6: JES2 converter exit (subtask)

Function

This exit allows you to provide an exit routine for scanning resolved Converter/Interpreter (C/I) text. If this exit is implemented and enabled, it is taken after the converter has converted each JCL statement into C/I text and once after all of the JCL for a particular job has been converted to C/I text.

You can use your exit routine to:

- Interpret C/I text and, based on this interpretation, decide whether JES2 should either cancel the job at the end of conversion processing or allow it to continue with normal execution.
- Pass messages to the converter that it will write to the JCLMSG data set for the job.
- Modify the C/I text.

After the converter has processed the entire job, this exit again allows you to direct JES2 either to cancel the job or to allow it to continue with normal execution.

C/I text is represented by ‘keys’ that identify the various JCL parameters. These keys are documented in the JES2 assembly, HASPDODC, which calls macros IEFVKEYS and IEFTXTFT, which are distributed in SYS1.MODGEN. Specifying KEYS on $MODULE causes IEFVKEYS to be expanded; specifying TEXT on $MODULE causes IEFTXTFT to be expanded. IEFVKEYS contains the definition of the values for each key, and IEFTXTFT contains the definition of the format of the Converter/Interpreter text. For more information about C/I text, see z/OS MVS Installation Exits.

Related exits

Use exit 44 if you need to alter any fields in the job queue element ($JQE). Altering fields in the $JQE in Exit 6 will not be successful because you are in the subtask environment.

Recommendations for implementing Exit 6

It is important to remember that Exit 6 is invoked because either:

- The converter just completed converting a JCL statement to C/I text
- The converter completed processing the entire job.

You could implement Exit 6 to keep certain counters—for instance, the number of DD cards received. Then, when the JCL for the entire job has been processed, the second part of your routine, the part that receives control when the code in R0 is 4, can determine whether to allow the job to continue based on the contents of these counters.

You should use extreme caution when modifying C/I text. If any of your changes cause a job to fail (because of an interpreter error), there will be no correlation of the error with the resulting abend on the user’s output. To modify or examine the C/I text:

- Ensure register 0 contains a X’00’ to indicate the invocation of Exit 6 is to process a converted JCL statement.
Exit 6

- Use any information from the C/I text for any installation-written control blocks.
- Make any necessary modifications to the C/I text. \textit{z/OS MVS Installation Exits} describes the rules for changing C/I text to ensure the changes you make will not cause the other problems in your installation, such as loss of data, loss of integrity and performance.

\textbf{Note:}

- You may want to issue messages to the JCLMSG data set to track the changes you make to the C/I text since none of the changes you make will be reflected in the job’s output. However, the changes you make will be reflected in the jobs SWA control blocks.
- If you need to change the job class or the job priority, use the JCTJCLAS or JCTIPRIO fields in the JCT. When conversion and all Exit 6 processing is completed for a job, JES2 will use these fields to update the corresponding JQE fields, JQEJCLAS and JQEPRIIO. JES2 also ensures that these changes are checkpointed.
- If you need to change the scheduling environment you should update the internal text for the job card. The converter validates the scheduling environment after Exit 6 receives control. If the scheduling environment is not valid, JES2 fails the job with a JCL error. Alternatively, you can supply a scheduling environment directly in the JCTSCHEN field in the JCT. You should delete any scheduling environment text unit in the internal text to prevent the converter from validating it. You must supply a valid scheduling environment in JCTSCHEN or the system cannot schedule the job for execution.

- Set the appropriate return code in register 15 or perform additional processing.

If you decide to fail the job, you should issue error messages to the operator and to the user. You can fail the job in Exit 6 by either:

- Setting flag CNMBFJOB in byte CNMBOPTS of the CNMB. See \textit{z/OS MVS Installation Exits} for information about obtaining and initializing the CNMB. If you set this flag, the converter continues to convert the job’s JCL and will fail the job after it has completely processed the job. You can only fail the job in this manner when register 0 contains a ‘X’00’.
- Setting a return code of 8 in register 15 before returning to JES2.

If you want to issue messages to the:

- JCLMSG data set, you must obtain a CNMB and initialize it with the message text. You can not issue any messages to the JCLMSG data set, if this is the last invocation of the exit (register 0 contains a 4). See \textit{z/OS MVS Installation Exits} for additional information about how to initialize the CNMB.
- Operator or user, issue a $WTO macro.

\textbf{Environment}

\textbf{Task}

JES2 subtask. You must specify ENVIRON=SUBTASK on the $MODULE macro.

\textbf{Restrictions}

- Do not attempt to modify checkpointed data from this exit.
See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.

Exit 6 must be MVS reentrant. See “Reentrant Code Considerations” in Chapter 2 for more information.

Do not alter any fields in the $JQE. The changes will not be successful because you are in the subtask environment.

Do not attempt to control the processing of the MVS converter by changing the C/I text at Exit 6. The converter does not examine the C/I text returned from the exit to determine what changes have been made. For example, you cannot use this exit to execute a procedure other than the one initially named on the EXEC statement, nor can you use this exit to control the printing of JCL statement images by altering the MSGLEVEL parameter on the JOB statement.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 6 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

Exit 6 is subject to suppression. The installation can implement exit 2 to set the 6th bit in the job exit suppression mask (JCTXMASK) or the installation can indicate the exit is disabled in the JES2 initialization stream.

Storage recommendations

- Private subpool that resides below 16-megabytes
- Word 1 in register 1 contains the address of a 16-byte work area

Mapping macros normally required

$DTE, $DTECNV, $HASPEQU, $HCT, $JCT, $JCTX, $MIT, $XIT, CNMB, KEYS, TEXT

Point of processing

This exit is taken from HASCNVT, the JCL conversion processor subtask, from within HASPCNVS at the following two times:

1. JES2 first gives your exit control after the converter has successfully converted a complete JCL job into its equivalent C/I text. The exit receives control once for each complete JCL statement unless the converter determines that any JCL statement for this job is in error. A complete JCL statement is considered to be a single JCL statement with all of its continuations. When Exit 6 is invoked, the user’s JCL has been merged with the expanded JCL from PROCLIB, and all substitutions for symbolic parameters have been made. Therefore, all of the standard modifications that JES2 will make to the C/I text are complete when the exit receives control.

2. JES2 also gives your exit control after all of the JCL for a particular job has been converted to C/I text even if the converter did detect a JCL statement that was in error. It occurs at the return from the link to the converter, before JES2
creates the scheduler work area (SWA) control blocks. JES2 will not create the scheduler work area (SWA) control blocks until all the JCL for a particular job has been converted to C/I text.

**Programming considerations**

1. If you suspect that an exit routine associated with this exit is causing a problem, the most expedient method of debugging is to disable the exit to determine whether the problem still occurs when your exit routine is not executed. Then, if the problem seems to be within your exit routine, you can test the routine by turning on the tracing facility.

   The trace record serves as a valuable debugging aid because it contains two copies of each C/I text, one before the call to your exit routine and one after the call to your exit routine. However, **do not** turn on tracing in your normal production environment or you will seriously degrade the performance of your system.

2. **Extending the JCT Control Block**

   You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information.

3. If you need to change the scheduling environment, use the JCTSCHEN field in the JCT.

4. Be sure to take into account when you manage any resources for the exit that the final call to the exit cannot be made if the converter task abends.

**Register contents when Exit 6 gets control**

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code indicating the status of conversion processing</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 5-word parameter list</td>
</tr>
<tr>
<td>Word 1 (+0)</td>
<td>Address of a 16-byte work area available to the installation.</td>
</tr>
<tr>
<td>Word 2 (+4)</td>
<td>If the code passed in R0 is:</td>
</tr>
<tr>
<td></td>
<td>• 0, this word points to the address of a 8192 (2000 hex) byte buffer that contains the C/I text of the converted JCL statement.</td>
</tr>
<tr>
<td></td>
<td>• 4, this word contains the address of the converter’s return code.</td>
</tr>
<tr>
<td>Word 3 (+8)</td>
<td>Address of the $DTE</td>
</tr>
<tr>
<td>Word 4 (+12)</td>
<td>Address of the $JCT</td>
</tr>
<tr>
<td>Word 5 (+16)</td>
<td>JES2 sets this to 0 before it passes control to the exit routine.</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
Register contents when Exit 6 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable on return</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 5-word parameter list</td>
</tr>
<tr>
<td></td>
<td><strong>Word 5 (+16)</strong> Address of a CNMB to be processed by the converter. If you want to pass a message(s) that the C/I will include in the JCLMSG data set for the job, this must contain the address of the CNMB (see z/OS MVS Data Areas, Vol 1 (ABEP-DALT) for information about the IEFCNMB macro).</td>
</tr>
<tr>
<td>2-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

- **0** Tells JES2 that if any additional exit routines are associated with this exit, execute the next consecutive exit routine. If there are no more exit routines associated with this exit point, continue with normal JES2 processing. If the exit routine was called when register 0 contains a X’00″, normal processing is the conversion of the next JCL statement. If the exit routine was called when register 0 contains a X’04‘, normal processing is to queue the job for execution.

- **4** Tells JES2 to ignore any additional exit routines associated with this exit for this C/I text and continue with normal processing. If the exit routine was called when register 0 contains a X’00’ normal JES2 processing is the conversion of the next JCL statement. If the exit routine was called when register 0 contained a X’04‘, normal JES2 processing is to queue the job for execution.

- **8** Tells JES2 to bypass execution and cancel the job; the job is queued for output rather than for execution. Conversion will continue until all JCL has been converted.

Coded example

Module HASX06A contains a sample of Exit 6.
Exit 7: Control block I/O (JES2)

Function

This exit allows you to provide an exit routine to:

- Receive control whenever control block I/O is performed by the JES2 main task.
- Perform I/O for any installation-specific control blocks you may have created.

This exit uses JES2 main task control block I/O.

Related exits

Whenever control block I/O is performed by a JES2 subtask or by a routine running in the user environment, Exit 8 provides the same function. In the HASPFSSM address space, Exit 25 provides this function.

Recommendations for implementing Exit 7

If you are performing I/O for a $JCT, then you can use this exit to determine the queue on which a job resides at any point of processing at which JCT I/O is performed for the JES2 main task.

To determine which queue the job is currently on:

1. Ensure the control block is the $JCT by comparing the value in X007CBID with the characters ‘JCT’.
2. Take the offset in the JCTJQE field of the JCT and add the offset to $JOBQPTR to locate the JQE.
3. Access the JQE and locate the JQETYPE field. JQETYPE can then be tested to determine on which queue, out of ten general queues, the current job resides.

The following table lists the ten possible queues along with their corresponding hexadecimal representations in JQETYPE:

<table>
<thead>
<tr>
<th>Queue</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$XEQ</td>
<td>X'40'</td>
</tr>
<tr>
<td>$INPUT</td>
<td>X'20'</td>
</tr>
<tr>
<td>$XMIT</td>
<td>X'10'</td>
</tr>
<tr>
<td>$RECEIVE</td>
<td>X'04'</td>
</tr>
<tr>
<td>$OUTPUT</td>
<td>X'02'</td>
</tr>
<tr>
<td>$HARDCOPY</td>
<td>X'01'</td>
</tr>
<tr>
<td>$PURGE</td>
<td>X'00'</td>
</tr>
<tr>
<td>$FREE</td>
<td>X'FF'</td>
</tr>
<tr>
<td>$SPIN</td>
<td>X'80'</td>
</tr>
</tbody>
</table>

Note: The $XEQ queue is actually two general queues, the conversion queue (which is X'40') and the execution queues. The class of each execution queue is indicated by the low-order 6 bits. For example, execution class “A” is X'41'. The scheme is similar to the EBCDIC character conversion chart in the MVS Reference Summary.

Programming considerations

The following are processing considerations for Exit 7:

- Use the PCEID field to determine which processor is reading or writing the JCT; this avoids unnecessary processing.
Exit 7

- You can determine if Exit 7 is being invoked for a transaction program or a batch job by either:
  - Determining if a $DSCT is contained in the $IOT.
  - Determining if byte JCTFLAG3 is set to JCT3TPI to indicate the job is a transaction program.
- Bit X007CBIN in the parameter list indicates that the control block contains either an incorrect eyecatcher or job key. When this bit is on, the exit should not rely on the contents of the control block. After the exit returns, JES2 will issue a disastrous error.
- **Extending the JCT Control Block**
  If field X007CBID contains the 4-character string 'JCT ' (note the trailing blank), you can add, expand, locate, and remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service for all control block WRITEs.
  For control block READs you should neither add nor expand extensions, because JES2 might not write any modifications from control block READs to spool. For more information about using the $JCTX macro extension service, see [z/OS JES2 Macros](https://www.ibm.com/support/knowledgecenter/SANS2C_11.1.0/com.ibm.zos.v1r11.jes2.doc_11.1.0/profjct2450.html).

**Point of processing**

Exit 7 is taken from the JES2 main task in the HASPNUC module, just after the control block is read from or just before the control block is written out to spool.

**Environment**

**Task**

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

**AMODE/RMODE requirements**

AMODE 31, RMODE ANY

**Supervisor/problem program**

JES2 places Exit 7 in supervisor state and PSW key 1.

**Recovery**

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

**Job exit mask**

Exit 7 is subject to suppression. The installation can suppress the exit by either implementing exit 2 to set the 7th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

**Mapping macros normally required**

$HASPEQU, $HCT, $MIT, $PCE, $XPL

**Register contents on entry to Exit 7**

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
</table>

120  z/OS V1R11.0 JES2 Installation Exits
A pointer to a parameter list with the following structure, mapped by $XPL:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Maintenance level</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit number</td>
</tr>
<tr>
<td>XPLXLEV</td>
<td>Version number</td>
</tr>
<tr>
<td>XPLCOND</td>
<td>Condition byte JES2 sets the condition byte with one of the following bit settings:</td>
</tr>
<tr>
<td></td>
<td>X007CBWR Control block is to be written</td>
</tr>
<tr>
<td></td>
<td>X007CBUN Unknown control block read</td>
</tr>
<tr>
<td></td>
<td>X007CBIN Invalid control block read</td>
</tr>
<tr>
<td>X007RESP</td>
<td>Not applicable on entry to Exit 7</td>
</tr>
<tr>
<td>XPLSIZE</td>
<td>Length of parameter list</td>
</tr>
<tr>
<td>X007CBID</td>
<td>The 4-character EBCDIC control block identifier</td>
</tr>
</tbody>
</table>

1-13 Address of the buffer that contains the control block
14 Return address
15 The entry address

### Register contents when Exit 7 passes control back to JES2

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A pointer to a parameter list, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Field Name Description</td>
</tr>
<tr>
<td></td>
<td>XPLRESP Response byte. Turn the X007IOER bit setting on in the response byte if an I/O error occurred. Upon return to JES2, JES2 will issue message $HASP096. If there are any other exits associated with this exit, they are ignored, and normal processing continues.</td>
</tr>
<tr>
<td>1-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0 Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no other exit routines associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit was called.

4 Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

8 Tells JES2 that an I/O error was encountered. Message $HASP096 is issued. If there are any other exit routines associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.
Exit 7

Coded example

Module HASX07A in SYS1.SHASSAMP contains a sample of Exit 7.
Exit 8: Control block read/write (user, subtask, and FSS)

Function

This exit allows you to provide an exit routine to receive control whenever a JES2 subtask, FSS printer, or a routine running in the user environment performs control block I/O.

You can use this exit to perform I/O for any installation-specific control blocks you may have created.

This exit uses Non-JES2 main task control block I/O.

Related exits

Whenever control block I/O is performed by the JES2 main task, Exit 7 serves the purpose of this exit.

If you intend on updating information for a transaction program, you should consider implementing Exit 31.

Programming considerations

The following are programming considerations for Exit 8:

- You can determine if Exit 8 is being invoked to process a transaction program by either:
  - Determining if a $DSCT is contained in the $IOT
  - Determining if byte JCTFLAG3 is set to JCT3TPI
- If you need to alter information for a transaction program, you should make changes in the $DSCT rather than the $JCT. If you update the $JCT for a transaction program, the updates you make may not be applicable. You should consider implementing exit 31 if you will be updating the $DSCT for a transaction program.
- **Extending the JCT Control Block**
  
  If field X008CBID contains the 4-character string ‘JCT ’ (note the trailing blank), you can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For more information about using this service, see [z/OS JES2 Macros](https://www.ibm.com/support/knowledgecenter/SSEK75_2.4.0/com.ibm.jes2.message.doc/doc/}.${\text{Point of processing}}

This exit is taken from the user address space (HASCSRDS).

JES2 gives control to your exit routine:

- Before it writes a control block and it writes the $CHK, $JCT, $IOT, $OCT, or $SWBIT into storage.
- After it reads a control block and it reads the $CHK, $JCT, $IOT, $OCT or $SWBIT into storage.
Exit 8

Environment

Task

- User address space
- JES2 subtask
- FSS address space using $CBIO.

You must specify ENVIRON=SUBTASK or ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Restrictions

Exit 8 must be in common storage

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

Exit 8 is subject to job exit mask suppression unless $JCT unavailable.

Mapping macros normally required

$HASPEQU, $HCCT, $JCT, $JCTX, $MIT, $XPL

Register contents on entry to Exit 8

The registers contain the following on entry to Exit 8:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>XPLXLEV</td>
</tr>
<tr>
<td></td>
<td>XPLCOND</td>
</tr>
<tr>
<td></td>
<td>X008CBWR</td>
</tr>
<tr>
<td></td>
<td>X008CBUN</td>
</tr>
<tr>
<td></td>
<td>X008CBIN</td>
</tr>
<tr>
<td></td>
<td>X008FSSM</td>
</tr>
<tr>
<td></td>
<td>XPLRESP</td>
</tr>
<tr>
<td></td>
<td>XPLSIZE</td>
</tr>
</tbody>
</table>
X008CBID The 4-character EBCDIC control block identifier
1 Address of the control block
2-10 N/A
11 Address of the $HCCT
12 N/A
13 Address of an OS-style save area
14 Return address
15 Entry address

Register contents on return to JES2

Upon return to JES2, the contents of the registers must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A pointer to a parameter list, mapped by $XPL</td>
</tr>
</tbody>
</table>

Field Name Description

XPLCOND Condition byte.
X008RESP Response byte. Turn the X008IOER bit setting on in the response byte if an I/O error occurred. After returning to JES2, JES2 issues message $HASP370. If there are any other exits associated with this exit, they are ignored, and normal processing continues.

1-14 Unchanged
15 Return code

A return code of:

0 Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no other exit routines associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit was called.

4 Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

8 Tells JES2 that an I/O error was encountered. Message $HASP370 is issued. If there are any other exit routines associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX08A in SYS1.SHASSAMP contains a sample of Exit 8.
Exit 9: Output excession options

Function

This exit allows you to choose how JES2 will process jobs or transaction programs that have exceeded the estimates for either:
- Output records
- Lines of SYSOUT data
- Pages of SYSOUT data
- Bytes of SYSOUT data

A user submitting a job can specify the estimates on either the JES2 /*JOBPARM JECL statement or the JOB JCL statement. If a job submitter does not specify the estimates, JES2 obtains the estimates from the ESTLNCT, ESTPUN, ESTPAGE, or ESTBYTE JES2 initialization statements.

Transaction programs obtain the output limits for SYSOUT data sets from TP profiles.

Related exits

JES2 will not invoke Exit 9 for jobs that exceed the OUTLIM specification. You should implement SMF exit IEFUSO - SYSOUT Limit Excession to process any jobs that exceed the OUTLIM specification. See z/OS MVS Installation Exits for additional information on SMF exit IEFUSO.

Exit 9 is invoked for a transaction program if your installation has implemented exit 43 to set the excession limits for SYSOUT data set created by a transaction program.

Environment

Task

USER task:
- User's address space
- JES2 address space - converter subtask

You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 9 in supervisor state and PSW key 0.

Restrictions

Exit 9 should be in either common storage (CSA) or in the link pack area (LPA).

Recovery

$ESTAE is in effect and provides minimal recovery. JES2 will attempt to recover from any program check errors experienced by Exit 9. However, you should not depend on JES2 for recovery and should implement a recovery procedure.
Exit 9

Job exit mask

Exit 9 is subject to suppression.

Mapping macros normally required

$HASPEQU, $HCCT, $JCT, $JCTX, $MIT, $XPL

Point of processing

From the user’s address space, JES2 invokes Exit 9 if the output limits have been exceeded while writing records to a SYSOUT data set. The output limits for a job are specified either in the:
- JES2 initialization stream
- job’s JCL or JECL.

Programming considerations

The following are programming considerations for Exit 9:

- You can determine if JES2 invoked Exit 9 to process a transaction program by determining if byte JCTFLAG3 is set to JCT3TPI.
- If exit 9 is processing a multi-transaction program, Exit 9 is invoked for every transaction submitted under the multi-transaction program.
- If Exit 9 is invoked from the JES2 address space, you cannot change the output exception limits for any of the following JES2 system data sets:
  - JES2 job log
  - JES2 messages
  - JES2 images file

  JES2 ignores any action taken in Exit 9 for the data sets.

- Extending the JCT Control Block

  You can add, expand, locate, and remove extensions to the job control table ($JCT) control block through the $JCTX macro extension service.

- Exit 9 is entered for each PUT if the limit(s) have been exceeded. Ensure that any increment provided takes this into account.

Register contents on entry to Exit 9

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a 12-byte parameter list with the following structure:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Eyecatcher - $XPL</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit identifier number - 9</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version level of the exit</td>
</tr>
<tr>
<td>X009IND</td>
<td>Indicates the environment from which Exit 9 was invoked. A value of:</td>
</tr>
</tbody>
</table>

- **X009USER** indicates which address space invoked Exit 9. See Programming Considerations for additional information.
- **X009CNCL** indicates CANCEL was specified on the job's JOB statement.
- **X009DUMP** indicates DUMP was specified on the job's JOB JCL statement.
- **X009WARN** indicates WARNING was specified on the job's JOB JCL statement.

**X009COND**
Indicates which SYSOUT limit was exceeded. A value of:
- **X009CEXC** indicates the SYSOUT data set exceeded the cards limit.
- **X009LEXC** indicates the SYSOUT data set exceeded the lines limit.
- **X009PEXC** indicates the SYSOUT data set exceeded the pages limit.
- **X009BEXC** indicates the SYSOUT data set exceeded the bytes limit.

**X009RESP**
Response byte

**X009JCT**
Address of the $JCT.

**X009LVAL**
Number of lines specified for the job's output limit.

**X009PVAL**
Number of pages specified for the job's output limit.

**X009BVAL**
Number of bytes specified for the job's output limit.

**X009DLIN**
The print/punch record count (in packed decimal format) for the job.

**X009DPAG**
The page count (in packed decimal format) for the job.

**X009DBYT**
The byte count (in packed decimal format) for the job.

**XPLSIZE**
Length of $XPL including base section

<table>
<thead>
<tr>
<th>2-10</th>
<th>Not applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Address of the $HCCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of a save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

### Register contents when Exit 9 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unchanged from entry</td>
</tr>
<tr>
<td>1</td>
<td>Address of $XPL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Eyecatcher - $XPL</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit identifier number - 9</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version level of the exit</td>
</tr>
<tr>
<td>X009RESP</td>
<td>Indicates processing options for the job. To indicate</td>
</tr>
</tbody>
</table>
Exit 9 changed the processing options you must set X009USR and if you want to:

- Suppress error messages indicating the job has exceeded its specified output limits, you should set X009RESP to X009SDEM.
- Change how JES2 processes a job when a SYSOUT data set created by a job exceeds its output limits. If you want to:
  - Abend the job and produce a dump, set X009RESP to X009XOVR and X009722D.
  - Cancel the job, set X009RESP to X009XOVR and X009722N.
  - Issue a warning message, set X009RESP to X009XOVR.
- Specify new increments for the output limits by setting X009OLIR and increases in one or more of the following:
  - X009RINC
  - X009PINC
  - X009BINC

<table>
<thead>
<tr>
<th>XPLSIZE</th>
<th>Length of $XPL including base section</th>
</tr>
</thead>
<tbody>
<tr>
<td>X009RINC</td>
<td>Exit 9's increase for records</td>
</tr>
<tr>
<td>X009PINC</td>
<td>Exit 9's increase for pages</td>
</tr>
<tr>
<td>X009BINC</td>
<td>Exit 9's increase for bytes</td>
</tr>
<tr>
<td>2-14</td>
<td>Unchanged from entry registers</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

- **0**: Indicates JES2 should continue processing with the next exit routine if one exists.
- **4**: Indicates JES2 should continue processing but ignore any additional exit routines.

**Coded example**

Module HASX09B in SYS1.SHASSAMP contains a sample of Exit 9.
Exit 10: $WTO screen

Function

This exit allows you to provide an exit routine to receive control every time that JES2 is ready to queue a $WTO message for transmission. If this exit is implemented and enabled, it receives control for all messages destined for remote stations and for other systems, as well as for all messages with a destination of local.

However, this exit does not receive control for messages generated by the subsystem interface or functional subsystem modules.

You can use your exit routine to interrogate the message’s console message buffer (CMB) and, based on this interrogation, direct JES2 either to cancel the message or to queue it for normal transmission. You can also use your exit routine to change the text of the message or to alter its console routing.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMode/RMode requirements

RMode ANY, AMode 31

Supervisor/problem program

JES2 places exit 10 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$CMB, $HASPEQU, $HCT, $MIT, $PCE

Point of processing

This exit is taken from the JES2 main task, from the HASPWQUE (special purpose CMB queuing) routine of the HASPCON (console support services) module, for all JES2 main task $WTO messages. The exit occurs at the beginning of HASPWQUE, after the $WTOR routine has processed the $WTO macro and before HASPWQUE queues the CMB containing the message for transmission. If, by passing a return code of 0 or 4, your routine allows the message to continue, control returns to HASPWQUE, which then queues the message for transmission. If, however, your exit routine cancels the message by passing a return code of 8, the transmission queuing performed by HASPWQUE is bypassed and JES2 gives control to $FRECMBR, the $FRECMB service routine.
Programming considerations:

1. This exit is taken only for WTOs issued from the JES2 main task.

2. To cancel a message, pass a return code of 8 to JES2. This return code directs JES2 to bypass the HASPWQUE routine, which normally queues the CMB for the console service processor, and to give control directly to the $FRECMBR routine, which then discards the message by freeing its CMB.

3. To change the text of a message, your routine must access either the CMBTEXT field or the CMBJOBN field. If the message does not contain the job’s name and number, the message text starts in CMBJOBN. The length of the message is always in the CMBML field. Your routine can either retrieve the existing message text and modify it or else generate a completely new message and then write the new or modified message over the original message. **If the new or modified message is longer or shorter than the original message, your routine should alter the CMBML field accordingly.** After altering the text of the message, pass a return code of 0 or 4 to direct JES2 to queue the CMB for transmission. JES2 will then send the new or modified message.

**CAUTION:**
Altering or deleting an end-line of a multi-line WTO can put JES2 command processing in a Wait State and no more responses to commands will be received.

4. To alter a message’s console routing, your routine should first test the flag byte CMBFLAG to determine whether the CMBFLAGW, CMBFLAGT, and CMBFLAGU flags are off. If these three flags are off, the CMBROUT field contains the MVS console routings. After altering CMBROUT, pass a return code of 0 or 4 to direct JES2 to queue the CMB for transmission. JES2 will base its console routing on the new contents of CMBROUT.

5. If register 0 contains a value of 4 when this exit is invoked, do not take any action that will result in a wait. For example, do not issue a $WAIT or do not invoke another service, such as $QSUSE, that might issue a $WAIT. A $WAIT can cause problems such as line time-outs or cause JES2 to terminate.

---

Register contents when Exit 10 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates whether JES2 can tolerate a $WAIT:</td>
</tr>
<tr>
<td></td>
<td>• If register 0 contains a value of 0, JES2 can tolerate a $WAIT.</td>
</tr>
<tr>
<td></td>
<td>• If register 0 contains a value of 4, JES2 cannot tolerate a $WAIT.</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $CMB</td>
</tr>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of the $PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 10 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $CMB</td>
</tr>
</tbody>
</table>
A return code of:

0  Tells JES2 that if any more exit routines are associated with this exit, execute the next consecutive exit routine. If there are no more exit routines associated with this exit, continue with normal processing by queuing the CMB for transmission.

4  Tells JES2 to ignore any additional exit routines associated with this exit and to continue with normal processing by queuing the CMB for transmission.

8  Tells JES2 to discard the message by freeing the CMB; the message is not queued for transmission.

Coded example

Module HASX10A in SYS1.SHASSAMP contains a sample of exit 10.
Exit 11: Spool partitioning allocation ($TRACK)

Function

This exit allows you to provide an exit routine from the JES2 main task that selects the spool volumes from which a job should allocate additional spool space when JES2 determines that additional spool volumes should be added to the available volumes for the job.

Before implementing this exit, you must determine if your installation uses spool partitioning. Your installation uses spool partitioning if FENCE=ACTIVE=YES is specified on the SPOOLDEF initialization statement.

Related exits

If you implement spool partitioning in Exit 11, you must also implement its companion, Exit 12.

The following table identifies the similarities and differences between Exit 11 and Exit 12.

Table 7. Comparison of Exit 11 and Exit 12

<table>
<thead>
<tr>
<th>Exit 11</th>
<th>Exit 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spool Partitioning Mask</td>
<td>Can only reset bits in the mask to allow spool space to be allocated from additional spool volumes.</td>
</tr>
<tr>
<td>• Initializes and resets bits in the mask.</td>
<td></td>
</tr>
<tr>
<td>• Can be used to define spool partitioning for the job.</td>
<td></td>
</tr>
<tr>
<td>Invoked To</td>
<td>Allocate additional spool space when JES2 determines the spool-allowed mask of the job should be expanded.</td>
</tr>
<tr>
<td>Allocate spool space for the first time for the job.</td>
<td></td>
</tr>
</tbody>
</table>

Recommendations for implementing Exit 11

To allow a job or transaction program to allocate spool space from another spool volume:

1. Modifying a 32-byte work area passed in register 1. Each bit in the IOTSAMSK corresponds to a spool volume defined to your installation and represents an entry in the direct access spool data set DSECT ($DAS). When a bit in the work area is set to:

   0 It indicates the spool volume is not currently available to the job and is a candidate for use by Exit 11.

   1 It indicates the spool volume is already allocated to the job.

You **must** implement Exit 11 so that it sets at least one additional bit in the work area to allow the job to allocate spool space from at least one additional spool volume. If Exit 11 does not make at least one spool volume available, JES2 will allocate spool space by either:

- Resetting all the bits to ones to allow the job to obtain spool space from any spool volume defined to the system.
- Resetting a single bit as indicated by the FENCE=ACTIVE=YES parameter on the SPOOLDEF initialization statement.
Exit 11

2. Place a X’08’ in register 15 and return to JES2. If your routine passes a return code of 8 to JES2 but hasn’t actually expanded the mask through the new mask returned in the spool mask work area, JES2 sets the spool partitioning mask as indicated by the FENCE= parameter on the SPOOLDEF initialization statement and to reissue the $TRACK request.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

Exit 11 is placed in supervisor state and PSW key 1.

Restrictions

You should not change the definition of the spool space from which a multi-transaction program allocates spool space. If you alter the volumes from which the multi-transaction program can allocate spool space, you may experience unpredictable results.

Recovery

Because Exit 11 is called from every stage in JES2 processing, there are significant variations the recovery environments JES2 provides for Exit 11. For example, when $TRACK is called from HASPRDR, an error in your exit routine may cause only the current job to fail; however, when $TRACK is called from HASPNET, an error in your exit routine may cause JES2 itself to fail. As with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine, and therefore any standard JES2 recovery that happens to be in effect is, typically, minimal. You should provide your own recovery within your exit routine.

Job exit mask

Exit 11 is subject to suppression. Exit 11 can be suppressed by either implementing exit 2 to set the 11th bit in the exit suppression mask (JCTXMASK) or by disabling the exit in the JES2 initialization stream.

Mapping macros normally required

$BUFFER, $DAS, $HASPEQU, $HCCT, $HCT, $IOT, $JCT, $JCTX, $MIT, $PCE, $SCAT, $TAB, $XECB, RPL

Point of processing

This exit is taken from the JES2 main task, from the $TRACK subroutine in HASPTRAK, when JES2 determines that the spools allowed mask for the job (IOTSAMS_MSK) needs to be updated. The spools allowed mask will be updated in two different situations:

- The job is using the maximum number of volumes ($FNCCNT in HCT) and there is no space available for allocation (that is, the volume is full, the volume is not
available for allocation or the volume does not have affinity for the system) on
the spool volumes from which the job is permitted to allocate space.

- The job is not yet using the maximum number of spool volumes (SPOOLDEF
  FENCE=VOLUMES=nnnn) regardless of whether there is space available on the
spool volumes from which the job is permitted to allocate space.

Exit 11 is not invoked if any of the following are true:

- The job is permitted to allocate space from any spool volume, that is, the spool
  partitioning mask (IOTSAMSK/JCTSAMSK) for the job is set to all ones (X'FF').
- Spool partitioning is in effect, the job is using the maximum number of spool
  volumes and space is available on those spool volumes.

Initially when a job or transaction program is started, JES2:
1. Sets the JCTSAMSK to all zeros to prohibit the job from allocating space from
   any spool volume
2. Determines if you have implemented spool partitioning. If you have not
   implemented Exit 2, Exit 11, or Exit 12 and have specified the
   FENCE=ACTIVE=NO parameter on the SPOOLDEF initialization statement,
   JES2 automatically sets JCTSAMSK to all ones so that the job can allocate
   spool space from any spool volume.

Programming considerations

The following are programming considerations for Exit 11:

- If you intend to base your allocation algorithm on values contained in fields of the
  $JCT, you must consider that the $JCT is sometimes unavailable and write a
  section of your exit routine to take control in these instances.

- **Locating JCT Control Block Extensions**
  You can locate extensions to the job control table ($JCT) control block from this
  exit using the $JCTXGET macro.
  
  - You can determine if a job or transaction program is requesting additional spool
    space by either:
  
    - Determining if a $DSCT is contained in the $IOT
    - Determining if byte JCTFLAG3 is set to JCT3TPI.
  
  - Determining whether a job is at its fencing limit or not
    
    - Spool partitioning is active if $MVFENCE is on.
    
    - The field $FNCCNT contains the fencing limit (SPOOLDEF
      FENCE=VOLUMES=nnnn).
    
    - CCTSPLAF contains the mask of spool volumes with affinity for this member.
    
    - Only count the volumes that have affinity for this member and are in the IOT
      spools allowed mask when checking to see if the job has reached the fencing
      limit. To do this, 'and' CCTSPLAF with IOTSAMSK and then use the $CNTBIT
      macro to obtain the number of volumes to compare with $FNCCNT. The
      number of bits on in IOTSAMSK may be equal to or exceed $FNCCNT and
      another volume should still be added if the job obtained some of its spool
      space on another member which has affinity to different spool volumes.
    
    - CCTVBLOB is the mask of spool volumes with space in the BLOB. Adding a
      spool volume that is not in CCTVBLOB will do no good since there is no
      space for it in the BLOB and therefore the job will not be able to allocate
      space on the volume.
Register contents when Exit 11 gets control

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of the 3-word parameter list, having the following structure:</td>
</tr>
<tr>
<td></td>
<td>word 1 (+0) Address of $IOT.</td>
</tr>
<tr>
<td></td>
<td>word 2 (+4) Address of $JCT (if available); otherwise 0. For example, the $JCT is unavailable when JES2 is acquiring:</td>
</tr>
<tr>
<td></td>
<td>• Space for the spooled remote messages or multi-access spool messages.</td>
</tr>
<tr>
<td></td>
<td>• A record for the $IOT for the JESNEWS data set.</td>
</tr>
<tr>
<td></td>
<td>word 3 (+8) Address of a 32-byte spool partitioning mask work area which is copied from the IOTSAMSK field in the $IOT.</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of $HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of $PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 11 passes control back to JES2

Before returning to JES2, the contents of the registers must be:

<table>
<thead>
<tr>
<th>Registers</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0      Tells JES2 that if any additional exit routines are associated with this exit, execute the next consecutive exit routine. If there are no additional exit routines associated with this exit point, this return code tells JES2 to set the spool partitioning mask as indicated by the FENCE parameter on the SPOOLDEF initialization statement setting and to reissue the $TRACK request.

4      Tells JES2 that even if additional exit routines are associated with this exit, ignore them; instead, set the spool partitioning mask as indicated by the FENCE parameter on the SPOOLDEF initialization statement setting and reissue the $TRACK request.

8      Tells JES2 that an updated version of the spool partitioning mask—with at least one additional bit turned on—has been passed to JES2 in the spool mask work area and will now determine later spool allocation. It also tells JES2 to reissue the $TRACK request.

Coded example

None provided.
Exit 12: Spool partitioning allocation ($STRAK)

Function

This exit allows you to provide an exit routine from a users address space or JES2 subtask that selects the spool volumes that a job or transaction program should allocate additional spool space from when JES2 determines that additional spool volumes should be added to the available volumes for the job.

Before implementing this exit, you must determine if your installation uses spool partitioning. Your installation uses spool partitioning if FENCE=ACTIVE=YES is specified on the SPOOLDEF initialization statement.

Related exits

If you implement spool partitioning in Exit 12, you must also implement its companion, Exit 11.

The following table identifies the similarities and differences between Exit 12 and Exit 11.

<table>
<thead>
<tr>
<th>Exit 12</th>
<th>Exit 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spool Partitioning Mask</td>
<td>Can only reset bits in the mask to allow spool space to be allocated from additional spool volumes.</td>
</tr>
<tr>
<td>Invoked To</td>
<td>Allocate additional spool space when JES2 determines the spool-allowed mask of the job should be expanded.</td>
</tr>
</tbody>
</table>

Allocate spool space for the first time for the job.

Recommendations for implementing Exit 12

To allow a job or transaction program to allocate spool space from another spool volume:

1. Modifying a 32-byte work area passed in register 1. The first $SPOLNUM bits in the IOTSAMSK correspond to the number of spool volumes defined to your installation. Each bit represents an entry in the direct access spool data set dsect ($DAS). When a bit in the work area is set to:

0  It indicates the spool volume is not currently available to the job and is a candidate for use by Exit 12.

1  It indicates the spool volume is already allocated to the job.

You must implement Exit 12 so that it sets at least one bit in the work area to allow the job to allocate spool space from at least one additional spool volume. If Exit 12 does not make at least one spool volume available, JES2 will allocate spool space by either:

• Resetting all the bits to ones to allow the job to obtain spool space from any spool volume defined to the system.

• Resetting a single bit as indicated by the FENCE=ACTIVE=YES parameter on the SPOOLDEF initialization statement.
Exit 12

2. Place a X'08' in register 15 and return to JES2.
   If your routine passes a return code of 8 to JES2 but hasn't actually expanded
   the mask through the new mask returned in the spool mask work area, JES2
   sets the spool partitioning mask as indicated by the FENCE= parameter on the
   SPOOLDEF initialization statement and to reissue the $STRAK request.

Environment

Task

USER task:
- Users address space
- JES2 subtask

You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places Exit 12 in supervisor state and PSW key:

<table>
<thead>
<tr>
<th>Environment</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>0</td>
</tr>
<tr>
<td>Subtask</td>
<td>1</td>
</tr>
</tbody>
</table>

Restrictions

You should not change the definition of the spool space from which a
multi-transaction program allocates spool space. If you alter the volumes from which
the multi-transaction program can allocate spool space, you may experience
unpredictable results.

Recovery

Because Exit 12 is called from every stage in JES2 processing, there are significant
variations the recovery environments JES2 provides for Exit 12. For example, when
$STRAK is called from HASPRDR, an error in your exit routine may cause only the
current job to fail; however, when $STRAK is called from HASPNET, an error in
your exit routine may cause JES2 itself to fail. As with every exit, your exit routine
should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose
of your exit routine, and therefore any standard JES2 recovery that happens to be
in effect is, typically, minimal. You should provide your own recovery within your exit
routine.

Job exit mask

Exit 12 is subject to suppression. You can suppress Exit 12 by either implementing
exit 2 to turn off the 12th bit in the job exit suppression mask (JCTXMASK) or you
can disable the exit suppressed.

Mapping macros normally required

$BUFFER, $DAS, $HASPEQU, $HCCT, $HCT, $IOT, $JCT, $JCTX, $MIT, $PCE,
$SCAT, $TAB, $XECB, RPL
Point of processing

This exit is taken from the $STRAK subroutine when JES2 determines that the spools allowed mask for the job (IOTSAMSK) needs to be updated. The spools allowed mask will be updated in two different situations:

- The job is using the maximum number of volumes (CCTFNCNT in HCCT) and there is no space available for allocation (that is, the volume is full, the volume is not available for allocation or the volume does not have affinity for the system) on the spool volumes from which the job is permitted to allocate space.
- The job is not yet using the maximum number of spool volumes (SPOOLDEF FENCE=VOLUMES=nnnn) regardless of whether there is space available on the spool volumes from which the job is permitted to allocate space.

This exit will not be invoked if any of the following are true:

- The job is permitted to allocate space from any spool volume, that is, the spool partitioning mask (IOTSAMSK) for the job is set to all ones (X'FF').
- Spool partitioning is in effect, the job is using the maximum number of spool volumes and space is available on those spool volumes.

Programming considerations

The following are programming considerations for Exit 12:

- If you intend to base your allocation algorithm on values contained in fields of the $JCT, you must consider that the $JCT is sometimes unavailable and write a section of your exit routine to take control in these instances.
- Locating JCT Control Block Extensions
  You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro.
  - You can determine if a job or transaction program is requesting additional spool space by either:
    - Determining if a $DSCT is contained in the $IOT
    - Determining if byte JCTFLAG3 is set to JCT3TPI
  - Determining whether a job is at its fencing limit.
    - Spool partitioning is active if CCTSMVFN is on.
    - The field CCTFNCNT contains the fencing limit (SPOOLDEF FENCE=VOLUMES=nnnn).
    - CCTSPLAF contains the mask of spool volumes with affinity for this member.
    - Only count the volumes that have affinity for this member and are in the IOT spools allowed mask when checking to see if the job has reached the fencing limit. To do this, 'and' CCTSPLAF with IOTSAMSK and then use the $CNTBIT macro to obtain the number of volumes to compare with CCTFNCNT. The number of bits on in IOTSAMSK may be equal or exceed CCTFNCNT and another volume should still be added if the job obtained some of its spool space on another member which has affinity to different spool volumes.
    - CCTVBLOB is the mask of spool volumes with space in the BLOB. Adding a spool volume that is not in CCTVBLOB will do no good since there is no space for it in the BLOB and therefore the job will not be able to allocate space on the volume.

Register contents when Exit 12 gets control

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exit 12

0  Return Code:
RC = 0    Invoked from user address space.
RC = 1    Invoked by jes2 converter subtask.
RC = 2    Invoked by JES2 subtask.

1  Address of the 3-word parameter list, having the following structure:

word 1 (+0)  Address of $IOT
word 2 (+4)  Address of $JCT (if available); otherwise 0 For example, the $JCT is unavailable when JES2 is acquiring:
  • Space for the spooled remote messages or multi-access spool messages
  • A record for the $IOT for the JESNEWS data set.
word 3 (+8)  Address of a 32-byte spool partitioning mask work area which is copied from the IOTSAMSK field in the $IOT.

2-9  Not applicable
10  Address of SJB/SJOB.
11  Address of $HCCT.
12  N/A
13  Address of $PCE
14  Return address
15  Entry address

Register contents when Exit 12 passes control back to JES2

Before returning to JES2, the contents of the registers must be:

<table>
<thead>
<tr>
<th>Registers</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if any additional exit routines are associated with this exit, execute the next consecutive exit routine. If there are no additional exit routines associated with this exit point, this return code tells JES2 to set the spool partitioning mask as indicated by the FENCE parameter on the SPOOLDEF initialization statement setting and to reissue the $STRAK request.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; instead, set the spool partitioning mask as indicated by the FENCE parameter on the SPOOLDEF initialization statement setting and reissue the $STRAK request.

8  Tells JES2 that an updated version of the spool partitioning mask—with at least one additional bit turned on—has been passed to JES2 in the spool mask work area and will now determine later spool allocation. It also tells JES2 to reissue the $STRAK request.
Coded example

None provided.
Exit 14: Job queue work select – $QGET

Function

This exit allows you to provide an exit routine that incorporates your own search algorithms for finding work on the job queue. You use your exit routine to search for an appropriate JQE on the job queue and to indicate when normal JES2 JQE processing should resume.

Note:

This exit is not called for workload management (WLM) initiator work selection; rather, you must use Exit 49 for that purpose. Also, you will find it easier to implement because it does not require that you copy JES2 decision-making algorithms into your exit routine. See “Exit 49: Job queue work select - QGOT” on page 289.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

This exit is associated with the $QGET routine, in HASPJQS, which is entered to acquire control of a job queue element (JQE).

The $QGET routine scans the appropriate queue for an element that:

• is not held
• is not already acquired by a previous request to the job queue service routines
• has affinity to the selecting JES2 member
• has independent mode set in agreement with the current mode of the selecting member.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 14 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $JQE, $MIT, $PCE
Point of processing

This exit is taken from the JES2 main task, from the $QGET routine of HASPJQS, after $QGET first obtains control of the shared queues and verifies that the member is not draining but before it selects a JQE from the appropriate queue.

Programming considerations

- The $QSUSE control of the checkpoint record is not maintained if your exit routine issues a $WAIT or invokes a service that issues a $WAIT. You should ensure in your exit routine that you retain control of the checkpoint record before returning to JES2.
- You must ensure that the spool volumes, where this job allocated space, are online. Also, the JQE cannot be busy, held, or on an inappropriate queue (such as the hardcopy queue).

```assembly
LH R15,$JQEMSKL Get JQE spool
EX R15,EXJQEMVC Get spools used by this job
NC $SPMSKWA,$SPLSLCT 'AND' with qualifying spools
EX R15,EXJQECLC If all spool volumes are not
BNE NEXTJQE available, get next job
```
- Ensure the job affinity will allow the routine to run on this member.

```assembly
$SETAFF REQUEST=TEST, Test for our affinity
AFFIELD=JQESAF, in the JQE to
AFTOKEN=$AFFINTY, see if we can run it.
REGAREA=$GENWORK, No, go find next job
FAIL=NEXTJOB
```
- Ensure the job's independent mode status matches the member status. If the member is in independent mode then the job must be in independent mode.

```assembly
TM $STATUS,$INMODE Is this member in independent mode?
BO EXIND Yes, make sure job is too
TM JQEFLAG2,JQE2IND Is job in independent mode?
BO NEXTJQE Yes, get next job
B EXAFF No, check affinity
EXIND TM JQEFLAG2,JQE2IND Is job in independent mode?
BZ NEXTJQE No, get next job
```
- Ensure that if the job has a scheduling environment, that it is available on this member.

```assembly
TM JQASCHE,FF-C' ' Scheduling environment?
JZ EXSCHE No, select the job
$SETAFF REQUEST=TEST, Test for availability
AFFIELD=JQASCHAF, in the JQE to
AFTOKEN=$AFFINTY, see if we can run it.
FAIL=NEXTJQE No, get next job
```
- Ensure that the JQE1ARMH flag is not on. If JQE1ARMH is on, the job has ended execution and is awaiting a possible restart by the automatic restart manager; the job cannot be selected.

```assembly
TM JQETYPE,$XEQ If job is on execution
BNE QGTCONTA queue and is held for
TM JQEFLAG7,JQE7SPIN spin processing in CSA
BO QNEXT bypass the job
TM JQEFLAG1,JQE1ARMH Job held for ARM restart?
B0 QNEXT Yes, get next JQE
```
- The address returned in the QGET parameter list must be the address of a JQA in update mode. That is, it must have been retrieved through $DOGJQE ACTION=(FETCH,UPDATE), $DOGJQE ACTION=(FETCHNEXT,UPDATE), or at some point changed from read mode to update mode through $DOGJQE ACTION=(SETACCESS,UPDATE).
• If you use Exit 14 to replace the normal JES2 job selection for execution or conversion and intend to use SECLABEL by system, then your exit routine must take into account the new SECLABEL affinity field in selecting an eligible job to run. If the RACF® SETROPT option for SECLABEL by system is active, then JES2 honors any SECLABEL system affinity restrictions when selecting a job. A new field, JQASCLAF, contains an affinity mask of JES2 MAS members where the SECLABEL is available. SECLABEL affinity applies only to selection of job for conversion and execution.

```
TM JQEFILAG3,JQE3JOB JQE a TSU or STC?
JM EXSLBL Yes, bypass SECLABEL aff
L R14,CVTPT(,0) Get CVT address
ICM R14,'B''1111'',CVTRAC-CVT(R14) Get RACF CVT addr
JZ EXSLBL None, skip next
TM RCVTML2F-RCVT(R14),RCVTSBYS SECLABEL by sys?
JNO EXSLBL No, skip SECLABEL aff
SPACE 1
$SETAFF REQUEST=TEST, Test if SECLABEL
    AFFIELD=JQASCLAF, is active on
    AFTOKEN=$AFFINTY, this member?
    FAIL=NEXTJQE No, get next job
```

• Exit 14 can perform duplicate job name check and instruct JES2 to bypass the normal duplicate job checks it would perform. You can also use the exit to allow a duplicate jobname to execute under certain situations. Setting QGTFNDUP causes JES2 checking for selected job to be bypassed.

• In Exit 14, JES2 sets the QGTFNOPT bit to NO by default and the exit-specified selection criteria is used. If you want to use the class optimization as your selection criteria, turn off the QGTFNOPT bit.

---

Register contents when Exit 14 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a QGET parameter list having the following structure:</td>
</tr>
<tr>
<td></td>
<td>+0 (word 1) Address of the node table</td>
</tr>
<tr>
<td></td>
<td>+4 (word 2) Address of control block</td>
</tr>
<tr>
<td></td>
<td>• PIT – if INWS</td>
</tr>
<tr>
<td></td>
<td>• DCT – if OJTWSC or OJTWSC</td>
</tr>
<tr>
<td></td>
<td>+8 (word 3) Address of class list (if applicable)</td>
</tr>
<tr>
<td></td>
<td>+12 (word 4) Address of the JQE</td>
</tr>
<tr>
<td></td>
<td>+16 (word 5) each byte is set as follows:</td>
</tr>
<tr>
<td></td>
<td>+16 Length of the class list</td>
</tr>
<tr>
<td></td>
<td>+17 Queue type (see the $QGET macro description for a list of these) This byte is set to '00' for queue types INWS, OJTWSC, and OJTW. Byte 18 (the type flag) is used to differentiate between these three queue types.</td>
</tr>
<tr>
<td></td>
<td>+18 Work selection type flag</td>
</tr>
<tr>
<td></td>
<td>+19 This byte is not part of the interface</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of the PCE</td>
</tr>
<tr>
<td>14</td>
<td>The return address</td>
</tr>
<tr>
<td>15</td>
<td>The entry address</td>
</tr>
</tbody>
</table>
Register contents when Exit 14 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of a QGET parameter list having the following structure:</td>
</tr>
<tr>
<td></td>
<td>+0 (word 1) Address of the node table</td>
</tr>
<tr>
<td></td>
<td>+4 (word 2) Address of the control block</td>
</tr>
<tr>
<td></td>
<td>+8 Address of the class list</td>
</tr>
<tr>
<td></td>
<td>+12 (word 4) Address of the JQE</td>
</tr>
<tr>
<td></td>
<td>+16 (word 5) each byte is set as follows:</td>
</tr>
<tr>
<td></td>
<td>+16 Length of the class list</td>
</tr>
<tr>
<td></td>
<td>+17 Queue type (see the $QGET macro description for a list of these) This byte is set to '00' for queue types INWS, OJTWSC, and OJTWS. Byte 18 (the type flag) is used to differentiate between these three queue types.</td>
</tr>
<tr>
<td></td>
<td>+18 Work selection type flag</td>
</tr>
<tr>
<td></td>
<td>+19 Response byte flags: X'80' - Initiator class list optimization not allowed</td>
</tr>
<tr>
<td>2-14</td>
<td>Not applicable</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit continue normal queue scan processing.

4  Tells JES2 to ignore any other exit routines associated with this exit and to continue normal queue scan processing.

8  Tells JES2 to bypass normal queue scan processing because a JQE was found by the exit routine. The address of the JQE the exit routine found is provided in the fourth word of the QGET parameter list (the address of which is returned in register 1).

12 Tells JES2 to bypass normal processing because a JQE was not found.

Coded example

None provided.
Exit 15: Output data set/copy select

Function

JES2 calls Exit 15 twice to allow you to instruct JES2 to:

- **First:** Change the number of copies of the output data set or bypass processing the current data set when JES2 first selects that data set for output processing.

- **Second:** Print (or not print) a data set separator page for each copy of the output data set.

The data set separator page exit point allows the exit routine to place a separator page between data sets. This is similar to the function provided by Exit 1, the separator page exit. See [z/OS JES2 Initialization and Tuning Guide](#) for a sample standard separator page. If your security policy requires it, use this exit to create headers that include the security label for each output data set for JES2 managed printers.

You could also use your exit routine to reset the addresses of the PRTRANS table and the CCW translate tables. The parameter list passed to your exit routine contains the default addresses for both the PRTRANS table and the CCW translate tables. Change the defaults by changing the parameter list to point to your own PRTRANS table and to point to your own CCW command code translate tables.

When translation is to occur for a local 1403 or a remote printer, the PRTRANS table translates user data and changes each line to be printed. The default PRTRANS table changes lowercase letters to uppercase and any characters that are invalid on a specific universal character set (UCS) to blanks. To determine if translation will occur, see item 9 on page 150.

The CCW table translates user-specified channel commands into installation-defined channel commands.

**CAUTION:**

Translation of initialization, diagnostic, or control CCWs may cause unpredictable results.

Programming considerations

1. Change the following information by changing the values in the parameter list:
   a. Copies to be printed (255 maximum)
   b. Pointer to translate table
   c. CCW translate table

2. Do not produce separator pages if JES2 called this exit for data set select, because printer setup processing has not occurred yet.

3. To determine if Exit 15 is to produce a data set separator, test bit X015SEPP in condition byte X015COND of the $XPL. If X015SEPP is on, create a separator. If X015SEPP is off, do not create a separator.

   The SEPDS= parameter on the PRT(nnnn), PUN(nnnn), R(nnnn),PR(m), or R(nnnn),PU(m) initialization statements indicates whether the installation wants data set separators created. The operator has the option to change the SEPDS= value by issuing the command $T device with the SEPDS= parameter specified. Before invoking Exit 15, JES2 sets bit X015SEPP to correspond to the current value of the SEPDS= parameter.
Exit 15

- If SEPDS=YES, JES2 turns on bit X015SEPP.
- If SEPDS=NO, JES2 turns off X015SEPP.

4. The data set copy count and copy group count cannot be changed on the separator page call to Exit 15 because setup processing has already occurred. Make these changes during the data set select call to Exit 15.

5. The data set copy group count affects separator pages this exit produces. JES2 sends the copy to the AFP printer before the calling Exit 15. The printer repeats all pages, including separator pages, on the basis of the copy group count.

6. If Exit 15 returns a copy count or a copy group count greater than 255, JES2 writes a symptom record to the LOGREC data set to a job log and reset(s) the field(s) in error to 1.

7. If the spooling capabilities of a remote SNA device (such as the 3790) are operating, use the $SEPPDIR macro to send a peripheral data information record (PDIR) to the device. Use the $GETBUF macro to supply this routine with HASP-type buffers and the $FREEBUF macro to release the buffers after your routine creates the separator.

8. Use SWBTUREQ REQUEST=RETRIEVE to retrieve any parameters a user specifies on the OUTPUT JCL statement you need to build your separator page. See z/OS MVS Programming: Assembler Services Reference ABE-HSP for more details about using the scheduler JCL facility and the SWBTUREQ macro.

9. For local printers running in JES mode or for remote printers, the TRANS= parameter on the printer’s initialization statement (statement PRT(nnnn) for a local printer, and statement R(nnnn).PR(m) for a remote printer) affects data translation for that printer:
   - If the initialization statement specifies TRANS=YES, JES2 translates each line of output sent to the device regardless of the device type or the setting of the PRINTDEF TRANS= parameter.
   - If the initialization statement specifies TRANS=NO, JES2 does not translate output sent to the device regardless of the device type or the setting of the PRINTDEF TRANS= parameter.
   - If the initialization statement specifies TRANS=DEFAULT or omits TRANS=, and the PRINTDEF statement specifies TRANS=YES, and the device is either a remote printer or a local printer other than an IBM 3211, IBM 3800, or IBM 3203 printer, JES2 translates each line of output sent to the device. Otherwise, JES2 does not translate output sent to the device.

10. You can determine whether JES2 invoked Exit 15 to process SYSOUT created by a transaction program by:
    - Determining if field X015DSCT contains the address of a $DSCT
    - Determining if byte JCTFLAG3 is set to JCT3TPI

11. **Locating JCT Control Block Extensions**
    You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

---

**Environment**

**Task**

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.
AMODE/RMODE requirements

AMODE 31, RMODE ANY

Recovery

$ESTAE recovery is in effect. If a program check occurs in the exit, JES2 interrupts the output currently processing on the device. The recovery routine will not call Exit 15 to free allocated resources. JES2 places the interrupted output groups in system hold with an indication that a failure occurred during separator exit processing. As with every exit, you should supply your own recovery within your exit routine.

Job exit mask

Exit 15 is subject to job exit mask suppression.

Mapping macros normally required

$DCT, $HASPEQU, $HCT, $JCT, $JCTX, $JOE, $JQE, $PCE, $PDDB, $XPL

Point of processing

This exit is taken from the JES2 main task in HASPPRPU. The exit is taken once for each output data set where the $PDDB matches the job output element ($JOE) and once for each copy of the data set.

Contents of registers on entry to Exit 15

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number</td>
</tr>
<tr>
<td>X015IND</td>
<td>Indicator byte. This byte indicates data set selection or data set separator processing as follows:</td>
</tr>
</tbody>
</table>

| X015DSEL   | Bypass processing the current data set, or change the number of copies of the data set to be produced. (These functions are only available at data set selection time.) |
| X015DSEP   | Produce a data set separator, change the print translate table, and change the CCW translate table. (These functions are only available at data set copy time.) |
| X015COND   | Condition byte. |
| X015RFSW   | Identifies whether the current PDDB has output characteristics identical to characteristics pointed to by X015SWBT. |
| X015SEPP   | If X015SEPP is on, SEPDS=YES |
Exit 15

was specified for the device and a separator is to be created. Otherwise, SEPDS=NO was specified and no separator is to be created.

X015RESP Response byte. If the X015BYPS bit setting is on in the response byte, then the current PDDB will be bypassed. Otherwise, the current PDDB will be processed.

X015DCT Address of $DCT
X015JCT Address of $JCT
X015DSCT Address of $DSCT or zeros for a batch job
X015JQE Address of the JOE
X015JOA Address of the artificial JOE (JOA). The JOA contains both the Work-JOE and the Characteristics-JOE.

Note: If the exit must update JOE fields, it should obtain and return an update mode JOA. For more information, see “Checkpoint control blocks for JOEs” on page 384.

X015PDDDB Address of the PDDB
X015SWBT Address of the SWBTU pointer list mapped by the SJTRSBTL DSECT in the IEFSJTRP parameter list for the first PDDB in the JOE. This field is zero if there is no OUTPUT JCL statement associated with the first PDDB. JES2 uses the SWBTU associated with the first PDDB to retrieve the output identification and delivery information for the entire output group.

X015NSWB Number of SWBTUs JES2 despooled. [z/OS MVS Programming: Assembler Services Reference IAR-XCT] contains additional information about SWBTU, and the IEFSJTRP parameter list.

X015PRTR Address of the print translate table
X015CCWT Address of the CCW translate table
X015NCOP The number of copies of this data set originally requested
X015CPRT The number of copies currently printed
X015CPGP Address of the current copy group
X015CGCT Current copy group count

2-10 Not applicable
11 Address of $HCT
12 Not applicable
13 Address of $PCE
14 Return address
15 Entry address

Contents of register when Exit 15 returns to JES2

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unchanged</td>
</tr>
<tr>
<td>1</td>
<td>Address of a parameter list mapped by $XPL:</td>
</tr>
</tbody>
</table>
Exit 15

**XPLRESP**  This response byte must be set by the exit before returning to JES2. Set the response byte to X015BYPS to bypass processing of the current PDDB. If this byte is equal to some other value, the current PDDB will be processed.

<table>
<thead>
<tr>
<th>2-14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unchanged</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine.</td>
<td>Tells JES2 to ignore any other exit routines associated with this exit.</td>
</tr>
</tbody>
</table>

**Coded example**

Module HASX15A in SYS1.SHASSAMP contains a sample of Exit 15.
Exit 16: Notify

Function

This exit allows you to change notify message routing and to examine and modify $WTO messages before they are sent to the TSO/E user.

Use your exit routine and the CMB to access the intended message, change it in place, or replace it with a new message.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 16 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

Exit 16 is subject to suppression. If the installation sets the 16th bit in the job exit suppression mask, it should be done only once. All transactions submitted under this initiator will not invoke Exit 16.

Mapping macros normally required

$CMB, $JCT, $JCTX, $HASPEQU, $HCT, $MIT, $PCE

Point of processing

This exit is taken from the output processor in HASPHOPE before sending the $WTO notify message.

Programming considerations

1. The CMB maps the $WTO parameter list. You map the parameter list by performing a USING on CMBWTOPL.
2. CMBML in the $WTO parameter list is the length of the message that is intended to be sent. Whether your exit routine changes the messages in place or replaces it, you must update CMBML with the length of the new message. The intended message can be changed in place for up to a length of 86 bytes.
3. To change the node where the notify message is to be sent, move correct node number NITNUM (of the NIT) to CBMTONOD.
4. To change the TSO/E user that the notify message is to go to store the TSO/E user id (7-character id) in CMB user.
Exit 16

5. On return from the exit, JES2 uses the address of the message in the first word of the parameter list.
6. For a return of 8 from your exit routine, JES2 resumes processing at OPNOTX in HASPPRPU.
7. **Locating JCT Control Block Extensions**
   You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For example, you can use these extensions to store job-related information. For more information, see [z/OS JES2 Macros](https://www.ibm.com/support/documentation/)

### Register contents when Exit 16 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code indicating if this is the first or succeeding $HASP165 (JOB nnnnn ENDED – reason text) message</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td><strong>Note:</strong> There is now only one HASP165 notify message for the job. The indicator is always set to 0 for compatibility.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Address of a 3-word parameter list with the following structure:</td>
</tr>
<tr>
<td></td>
<td>Word 1 (+0)</td>
</tr>
<tr>
<td></td>
<td>Word 2 (+4)</td>
</tr>
<tr>
<td></td>
<td>Word 3 (+8)</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of the output processor $PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

### Register contents when Exit 16 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of the 3-word parameter list</td>
</tr>
<tr>
<td>2-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

| 0        | Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with exit continue normal notify processing. |
Exit 16

4 Tells JES2 to ignore any other exit routines associated with this exit and to continue normal notify processing.

8 Tells JES2 not to issue the notify $WTO.

Coded example

None provided.
Exit 17: BSC RJE SIGNON/SIGNOFF

Function

This exit allows you to exercise more control over your BSC RJE remote devices. With this exit you can implement exit routines to:

- Selectively perform additional security checks beyond the standard password processing of the signon card image.
- Selectively limit both the number and types of remote devices that can be on the system at any one time.
- Selectively bypass security checks.
- Implement installation-defined scanning of signon card images.
- Collect statistics concerning RJE operations on the BSC line and report the results of the activity.

See Appendix B, “Sample code for Exit 17 and Exit 18,” on page 375 for a sample code for Exit 17.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 17 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

This exit is not subject to job exit mask suppression.

Storage recommendations

Mapping macros normally required

$DCT, $HASPEQU, $HCT, $MIT, $PCE, $RAT

Point of processing

This exit is taken from the JES2 main task, during BSC RJE signon and signoff processing of HASPBSC. Three exit points are defined; two signon exit points for performing additional security or checks and one signoff exit point for gathering statistics about terminal usage.

The exit gets control during signon in the MSIGNON routines of HASPBSC, and after signon and password processing.
The exit is given control before signon and password processing, allowing your exit routine to scan the incoming signon card. Your exit routine may also bypass both the JES2 syntax checking of the signon and the remote and line password parameters on the signon card or just bypass only the signon syntax checking. JES2 also gives the exit control after signon and password processing, allowing your exit routine to provide additional setup of the remote terminal environment.

JES2 also gives the exit control at sign off, after writing the disconnect message at label MDSWTO.

Programming considerations

1. For exit point MSOXITA (R0=0) your exit routine has the option to return a return code that allows the user to specify that the signon should be rejected. A return code of 12 or 16 indicates that normal HASPBSC signon processing can be bypassed. In this case your installation exit routine is responsible for performing all the necessary syntax processing that HASPBSC does and for returning a valid RAT entry pointer in R0.

2. For the signoff exit point your exit routine should return a return code of 0 or 4 so that normal processing can continue.

3. To define and implement an installation-defined remote name, change the remote name to a standard JES2 remote name on the signon card and return with a return code of 0, or supply a valid RAT pointer (valid for the installation-defined remote name) and return with or return code of 12 or 16.

4. Your installation exit routine should not issue a $WAIT or invoke a service routine that issues a $WAIT.

5. For the syntax of the signon card, see z/OS JES2 Initialization and Tuning Guide.

6. The $RETURN macro destroys the contents of register 0. Therefore, if you return the RAT address in R0, be certain to have provided a $STORE R0 instruction before the $RETURN to place the contents of R0 in the current save area before return to JES2.

Register contents when Exit 17 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates whether signon or signoff processing is in effect. The following values apply: 0 indicates a signon before signon parameters are processed. 4 indicates a signon after the signon parameters have been processed. 8 indicates signoff processing.</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 5-word parameter list, having the following structure: Word 1 (+0) address of the remote attribute table (RAT) (for R0=0 only) address of the RAT entry (for R0=4 or 8) Word 2 (+4) address of the line DCT</td>
</tr>
</tbody>
</table>
**Register contents when Exit 17 passes control back to JES2**

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the remote's RAT entry when the return code in R15 is 12 or 16 and the signon indication in R0 is &quot;0&quot;&lt;br&gt;Otherwise not applicable</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit continue normal signon/signoff processing continues.</td>
</tr>
<tr>
<td>4</td>
<td>Tells JES2 to ignore any other exit routines associated with this exit and to continue normal signon/signoff processing.</td>
</tr>
<tr>
<td>8</td>
<td>Tells JES2 to terminate normal signon processing. No audit record is produced in this case. If you require an audit of this failure, your exit routine must issue a call to SAF to perform the audit.</td>
</tr>
<tr>
<td>12</td>
<td>Tells JES2 to call SAF with the remote id set in this exit and the password received on the /*SIGNON statement.</td>
</tr>
<tr>
<td>16</td>
<td>Tells JES2 to call SAF with the remote id from the /*SIGNON statement but do not verify the password.</td>
</tr>
</tbody>
</table>

**Note:** RC 8, 12, and 16 are only valid for the exit when called from label MSOXITA (that is, the first call to the exit, R0=0).

**Coded example**

See Appendix B, “Sample code for Exit 17 and Exit 18,” on page 375.
Exit 18: SNA RJE LOGON/LOGOFF

Function

This exit allows you to exercise more control over your SNA RJE remote devices. With this exit you can implement exit routines to:

- Selectively perform additional security checks beyond the standard password processing of the signon card image.
- Selectively limit both the number and types of remote devices that can be on the system at any one time.
- Selectively bypass security checks.
- Implement installation-defined scanning of signon card images.
- Collect statistics concerning RJE operations on the SNA line and report the results of the activity.

For a sample code of Exit 18, see Appendix B, “Sample code for Exit 17 and Exit 18,” on page 375.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 18 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$DCT, $HASPEQU, $HCT, $ICE, $MIT, $PCE, $RAT

Point of processing

This exit is taken from the JES2 main task during the SNA RJE logon and logoff processing of HASPSNA. Three exit points are defined for logon processing:

- At exit point MSNALXIT for a normal logon during REQ END processing after label MSNALPAR, your exit routine can be invoked to:
  - continue normal logon processing.
  - terminate normal logon processing.
  - perform password checking but not syntax checking.
  - bypass syntax and password checking.
Exit 18

When using multiple logical units, JES2 invokes Exit 18 from MSNALXIT for each logical unit on the remote when the logical unit logs on.

- At exit point MSNALXT2 your exit can get control when the remote terminal is logged on.
- Just before checkpointing the remote autologon at exit point MALGXIT, your exit can control autologon for the remote terminal.

One exit point (MICEXIT) is defined for logoff processing. This exit point is after label MICEDMSG in the session control subroutines of HASPSNA before the remote logoff message is issued. You can use this exit point for gathering statistics and reporting remote device activity.

Programming considerations

1. In logoff processing, JES2 does not expect a return code from your exit routine. Normal logoff processing proceeds.
2. Your installation exit routine should not issue a $WAIT or use a service routine that issues a $WAIT.
3. To define and implement a installation-defined remote name, change the remote name to a standard JES2 remote name on the remote logon card and return with a return code of 0, or supply a valid RAT pointer (valid for the installation-defined remote name) and return with a return code of 12 or 16.

Register contents when Exit 18 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A logon or logoff indication having the following meanings:</td>
</tr>
<tr>
<td>0</td>
<td>indicates syntax processing for a normal logon</td>
</tr>
<tr>
<td>4</td>
<td>indicates logon processing for a normal logon after logon parameters have been processed</td>
</tr>
<tr>
<td>8</td>
<td>indicates logoff processing</td>
</tr>
<tr>
<td>12</td>
<td>indicates autologon processing</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 5-word parameter list having the following structure:</td>
</tr>
<tr>
<td>Word 1 (+0)</td>
<td>address of the remote attribute table (RAT) when R0 indicates a normal logon process of “0”</td>
</tr>
<tr>
<td></td>
<td>address of a RAT entry when R0 indicates other than a normal logon process (that is, R0 contains a value of 4, 8, or 12).</td>
</tr>
<tr>
<td>Word 2 (+4)</td>
<td>0 during syntax processing (that is, R0=0)</td>
</tr>
<tr>
<td></td>
<td>address of the line DCT after logon is complete (that is, R0=0)</td>
</tr>
<tr>
<td>Word 3 (+8)</td>
<td>address of the ICE</td>
</tr>
<tr>
<td>Word 4 (+12)</td>
<td>address of the bind user data when R0 indicates normal logon processing (that is, R0=0). The format</td>
</tr>
</tbody>
</table>
of the bind user data is determined by installation VTAM® application programs that define the bind user data.

**Word 5 (+16)** length of the bind user data when R0 indicates normal logon processing (that is, R0=0).

- 2-10 N/A
- 11 Address of the HCT
- 12 N/A
- 13 Address of the line manager PCE
- 14 Return address
- 15 Entry address

### Register contents when Exit 18 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the RAT entry when R15 contains a return code of 12 or 16 and the logon indication in R0 is 0. Otherwise register 0 is ignored.</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

**A return code:**

- 0 Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit continue normal logon/logoff processing.
- 4 Tells JES2 to ignore any other exit routines associated with this exit and to continue normal logon/logoff processing.
- 8 Tells JES2 to terminate normal logon processing (R0=0 or 12 only). No audit record is produced in this case. If you require an audit of this failure, your exit routine must issue a call to SAF to perform the audit.
- 12 Tells JES2 to call SAF with the remote id set in this exit and the password received during logon processing (R0=0 only).
- 16 Tells JES2 to call SAF with the remote id received during logon processing but do not verify the password (R0=0 only).

### Coded example

See Appendix B, “Sample code for Exit 17 and Exit 18,” on page 375.
Exit 19: Initialization statement

Function

This exit allows you to process each JES2 initialization statement before JES2 processes the statement. You can use your exit routine to do any of the following functions:

- check or analyze each initialization statement.
- alter values supplied on an initialization statement.
- implement your own initialization statements.
- modify, replace, delete, or insert statements in the initialization statement stream.
- terminate JES2 initialization.
- tailor the initialization statement stream to provide for specific requirements of this start of JES2 (e.g., add or delete parameters based on the period within administrative cycles or the operator shift).

Environment

Task

JES2 main task (Initialization) – JES2 dispatcher disabled. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 19 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

Exit 19 is not subject to suppression.

Mapping macros normally required

$CIRWORK, $HASPEQU, $HCT, $MIT, $PCE

Point of processing

This exit is taken during JES2 initialization from the initialization routine (IR) that processes parameter input (IRPL) in HASPIRPL. IRPL is called out of the initialization routine processing loop (IRLOOP) in HASPIRA before most other IRs have been called. Previously executed IRs have processed the initialization options, analyzed the SSI status, and allocated a series of temporary and permanent control blocks. Exit 0 routines, called during initialization options processing, may have allocated installation control blocks that may be used now by Exit 19 routines.

HASPIRPL opens the initialization parameter data set (HASPPARM) and then begins a loop; get an initialization statement from HASPPARM or the operator
console or a previous insertion by Exit 19, pass it to Exit 19, log the statement, process the statement using the $SCAN facility if Exit 19 has not indicated it should be deleted. When all input is exhausted, IRPL closes the parameter and log data sets.

Programming considerations

1. Your EXIT(nn) and LOADmod(jxxxxxxx) initialization statements for this exit must be placed in the initialization deck ahead of those initialization statements that your exit routine is to scan. The EXIT(nn) statement must enable (STATUS=ENABLED) the exit; the $T EXIT(nn) command cannot be used to enable (STATUS=ENABLED) the exit at a later time since the point of processing for Exit 19 is before the time at which the command processor is made functional.

2. Tracing for this exit is disabled because of its sequence in the initialization process.

3. JES2 does not have a recovery environment established at the processing point for Exit 19 (the JES2 ESTAE will process termination, but not recover).

4. Because Exit 19 is called early in JES2 initialization, some main task services may not be functional and some control blocks and interfaces may not be established. The JES2 dispatcher is not yet functional, so MVS protocol should be used in Exit 19 routines (WAIT rather than $WAIT, ESTAE rather than $ESTAE, and so forth).

5. The CONSOLE statement simulated after all other parameter input is exhausted if the CONSOLE initialization option was specified is not presented to Exit 19 exit routines.

6. Exit 19 routines may change the initialization statement passed or replace it by changing the address and length in the exit parameter list. They may also indicate, through a return code, that JES2 should bypass processing of the statement (perhaps because the routine has processed the statement already). Note that JES2 writes the statement (and any later diagnostics) to the log data set and hardcopy console only after return from the exit. Therefore the exit routines may want to log the statement passed from JES2, for diagnostic purposes, before changing or replacing it. The $STMTLOG macro and service routine is provided to perform the logging function.

7. Independent of the actions of the exit routine that effect the status of the statement passed, a new initialization statement may be inserted into the parameter stream by the exit routine by returning a statement address and length in the exit parameter list. The inserted statement will be processed when the current statement is completely processed. Note that the current statement is not completely processed until either it is bypassed by exit 19, successfully scanned and processed by JES2, or found to be in error by JES2 and the resultant operator interaction by JES2 is complete. Since the operator interaction may involve input of multiple new initialization statements from the operator, the inserted statement may not be processed until after later calls to Exit 19. Also, when there are multiple exit 19 routines, only one routine can perform a statement insertion. For that reason, Exit 19 routines should verify that the insertion statement address and length in the exit parameter list are zero before using those fields to insert a statement.

8. The processing that JES2 does for each statement after calling Exit 19 is performed using the JES2 $SCAN facility and a collection of tables. The tables define the parameter input allowed and how to process it. The scan may involve multiple levels of scanning, that is, parameters which have sub-parameters, and
so on. At each level, a new table is used. Each table is actually composed of two tables, an installation-defined table followed by a JES2-defined table. By specifying installation-defined tables, an installation can implement its own initialization parameters on existing JES2 statements, or replace the JES2 definition for existing statements or parameters. Thus this function can be accomplished without implementing Exit 19, or with an implementation of Exit 19. Also, the $SCAN facility itself can be used from an Exit 19 routine to process initialization statements.

Register contents when Exit 19 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>An indication of how the initialization input was supplied. The following values in R0 are possible:</td>
</tr>
<tr>
<td></td>
<td>0: input came from the HASPARM parameter library file</td>
</tr>
<tr>
<td></td>
<td>4: input came from the console</td>
</tr>
<tr>
<td></td>
<td>8: input came from a previous insertion by an Exit 19 routine.</td>
</tr>
<tr>
<td>1</td>
<td>A 4-word parameter list having the following structure:</td>
</tr>
<tr>
<td></td>
<td><strong>Word 1 (+0)</strong>: address of the initialization statement about to be processed. You can modify the statement or replace the statement by altering this field.</td>
</tr>
<tr>
<td></td>
<td><strong>Word 2 (+4)</strong>: length of the complete initialization statement passed. If you alter the passed statement or replace it, you should reset this field to the correct new statement length.</td>
</tr>
<tr>
<td></td>
<td><strong>Word 3 (+8)</strong>: a word that can be used by Exit 19 to specify the address of an initialization statement you want to insert at the next possible statement insertion point. JES2 will log an information diagnostic indicating the statement was inserted by Exit 19.</td>
</tr>
<tr>
<td></td>
<td><strong>Word 4 (+12)</strong>: length of the initialization statement pointed to by word 3.</td>
</tr>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of initialization PCE – the PCE work area for this PCE is the common initialization routine work area, mapped by the $CIRWORK macro</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 19 passes control back to JES2

Upon return from this exit, the register contents must be:
Exit 19

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit, continue normal initialization statement processing. The exit routines might have changed or replaced the initialization statement passed.

4  Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit, continue normal initialization statement processing. The exit routines might have changed or replaced the initialization statement passed. However, JES2 should ignore any other exit routines associated with this exit.

8  Tells JES2 to bypass this initialization statement and continue with the next statement. JES2 will log the statement and a diagnostic information message indicating it was bypassed by Exit 19.

12  Tells JES2 to terminate all initialization processing and exit the system. HASPIRPL issues message $HASP864 and returns to the IRLOOP with return code 8.

16  Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit, continue normal initialization statement processing. The exit routines might have changed or replaced the initialization statement passed. However, the system is not to substitute text for system symbols that are specified in the initialization statement.

Coded example

None provided.
Exit 20: End of input

Function

This exit allows you to do the following:

• Selectively assign a job's priority, affinity, execution node, SCHENV, and job class, and influence next phase of job processing based on an installation's unique requirements and processing workload.

• Based on installation-defined criteria, terminate a job's normal processing and selectively print or not print its output.

• Exit 20 allows input processing - end of input.

Note: See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 20 in supervisor state and PSW key 1.

Recovery

$ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. You should provide your own recovery within your exit routine.

Job exit mask

Exit 20 is subject to suppression. You can suppress Exit 20 by either setting the 20th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the initialization stream.

Mapping macros normally required

$JCT, $JCTX, $HCT, $PCE, $HASPEQU, $MIT, $JRW, $HCCT, $BUFFER, RPL, $DCT

Point of processing

This exit is taken in the subroutine CJOBEND or in the subroutine CJOBKILL of HASCSRP in the JES2 main task.
Programming considerations

1. To change affinity, set the X020SAF field in the $XPL work area using the $SETAFF macro.
   To allow the job to run on any member:
   $SETAFF REQUEST=ANY,AFFIELD=X020SAF
   To allow the job to run on only this member:
   $SETAFF REQUEST=CLEAR,AFFIELD=X020SAF
   $SETAFF REQUEST=ADD,AFFIELD=X020SAF
   AFTOKEN=$AFFINTY

2. If MVS submits a job through an internal reader, it can force a job’s affinity to the local member. This can occur when the automatic restart manager restarts a job. The automatic restart manager expects the job to execute on a specific member, and will change the job’s affinity so the job can run on that specific member, if necessary. If the automatic restart manager has changed the job’s affinity, the X0201ARM flag in the XPL is on. You can test this flag and determine whether the affinity was changed. With that information, you can then decide whether to avoid changing the affinity.

3. To set independent mode for a job, the installation must turn on the bit X0201IND in X020FLG1.
   To put jobs that start with the characters 'IND' into independent mode:
   EXIT20 $ENTRY BASE=R12,SAVE=YES Set entry point
   LTR R10,R10 If JCT not present
   BZ RRET can’t check jobname
   CLC =C'IND',JCTJNAME Job want independent mode?
   BNE RRET No, leave flags alone
   OI X020FLG1, X0201IND Set independent mode
   RRET $RETURN RC=0 Return to caller

4. To change the priority, set X020PRIO in the XPL. The priority is contained in the 4 high-order bits of X020PRIO. For example, a value of ’C0’ indicates priority 12. (See z/OS JES2 Initialization and Tuning Reference for further details on setting and changing job priority.)
   • To change the execution node, update X020XNOD with the half word binary value of the node. Use the $DEST macro to convert an EBCDIC node name to the internal binary representation of the node number
   • To change the job class, place the new job class in X020JCLS. This is honored only if the job is a batch job, not if it is an STC or TSU job.
   • The exit can influence the next phase of the job in most circumstances. Place the next phase value in X020NEXT. X020NEXT is primed with the phase that JES2 believes is the correct next phase when the exit is called. The exit can place one of these values in X020NEXT:
     $OUTPUT Places the job in the OUTPUT queue unless JES2 has already determined that the job should be purged. In that case, X020NEXT is ignored.
     $PURGE Places the job in the PURGE queue.
     Any other phase JES2 honors the request unless it has already determined that the job should be placed in the OUTPUT or PURGE phase.
The next phase can also be set through the return code in R15. If one or both of the specifications specify PURGE; then PURGE will be the next phase. If neither specify PURGE, but one or both specify OUTPUT; then the next phase will be OUTPUT.

5. **Extending the JCT Control Block**
   You can add, expand, locate, and remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see [z/OS JES2 Macros](#).

6. This exit will not be taken under the following circumstances:
   • The JES2 input service processor fails the job because JES2 does not identify a JOB card within the input stream.

7. If you need to change the scheduling environment, use the X020SENV field in the XPL.

8. Setting the X020AVF response bit does NOT influence the next phase of the job. To influence the next phase of the job, you must use the documented methods.

---

### Register contents when Exit 20 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code indicating:</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher.</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level for base XPL.</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number.</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version number for exit</td>
</tr>
<tr>
<td>X020IND</td>
<td>Indicator byte.</td>
</tr>
<tr>
<td>X020COND</td>
<td>Condition byte.</td>
</tr>
<tr>
<td>X020GJOB</td>
<td>Condition bit that specifies a normal job.</td>
</tr>
<tr>
<td>X020JECL</td>
<td>Condition bit that specifies a JECL error.</td>
</tr>
<tr>
<td>X020BSAF</td>
<td>Condition bit that specifies an SAF failure.</td>
</tr>
<tr>
<td>X020WSEL</td>
<td>Condition bit that specifies the job failed to meet work selection criteria.</td>
</tr>
</tbody>
</table>
Exit 20

**X020RESP**  
Response byte.

**X020NORM**  
Response bit that specifies to do normal process.

**X020OUTP**  
Response bit that specifies to terminate with output.

**X020PURG**  
Response bit that specifies to terminate job without printing the output.

**X020AVF**  
Response bit that indicates the exit’s job verification failed.

**XPLSIZE**  
Size of parm list, including base section.

**X020JCT**  
Address of the JCT.

**X020JQE**  
Address of update mode JQA.

**X020DCT**  
Address of the DCT.

**X020AREA**  
Address of the JRW.

**X020PRIO**  
Job priority (Input/Output field)

**X020FLG1**  
Flags

**X020XNOD**  
Execution Node (Input/Output field)

**X020SAF**  
Full system affinity mask (Input/Output)

**X020SENV**  
Scheduling Environment (Input/Output field)

**X020JCLS**  
Job class (Input/Output field)

**X020NEXT**  
Next job phase (Input/Output field)

2-9  
Not applicable

10  
Address of the JCT.

11  
Address of the HCT.

12  
Not applicable

13  
Address of the HASPRDR PCE.

14  
Return address.

15  
Entry address

**Register contents when Exit 20 passes control back to JES2**

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Address of a parameter list mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td><strong>X020RESP</strong>  Response byte that may be set by the exit before returning to JES2.</td>
</tr>
<tr>
<td>15</td>
<td>Return code.</td>
</tr>
</tbody>
</table>
A return code of:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit continue normal processing.

4  Tells JES2 to ignore any other exit routines associated with this exit and to continue normal processing.

8  Tells JES2 to terminate normal processing and print the output.

12 Tells JES2 to terminate normal processing without printing the output.

Coded example

Module HASX20A in SYS1.SHASSAMP contains a sample of Exit 20.
Exit 21: SMF record

Function

This exit allows you to do the following:

- Selectively queue or not queue the SMF record of JES2 control blocks for processing by SMF.
- Obtain and create SMF control blocks before queuing.
- Alter content and length of SMF control block before queuing.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 21 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $MIT, $PCE, $SMF

Point of processing

This exit is taken in HASPNUC whenever a JES2 processor queues an SMF record for eventual processing by the JES2-SMF subtask. The $QUESMFB routine in HASPNUC places a JES2-SMF buffer on the queue of busy JES2-SMF buffers. (The $SMFBUSY cell in the HCT points to the busy queue.)

Programming considerations

1. When modifying the SMF record, your exit routine can increase the size of the SMF record up to a length of SMFLNG (bytes).
2. You can issue $GETSMFB and $QUESMFB in your exit routine.
3. The SMF record type is detected by examining the SMFHDRTY field, not the SMFTYPE field of the SMF DSECT.
   For more information about SMF, see z/OS MVS System Management Facilities (SMF).
4. You can determine if JES2 invoked exit 21 to record information for a transaction program by determining if byte JCTFLAG3 is set to JCT3TP.
Register contents when Exit 21 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Zero (0)</td>
</tr>
<tr>
<td>1</td>
<td>SMF buffer address.</td>
</tr>
</tbody>
</table>

This buffer will contain either an SMF record or a job management record (JMR) based on the value of field SMFTYPE.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Record Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X'00'</td>
<td>SMF record</td>
</tr>
<tr>
<td></td>
<td>X'40'</td>
<td>Large SMF record.</td>
</tr>
<tr>
<td></td>
<td>X'80'</td>
<td>JMR record.</td>
</tr>
</tbody>
</table>

2-9    N/A
10     Address of the JCT or 0
11     Address of the HCT
12     N/A
13     Address of the caller’s PCE
14     Return address
15     Entry address

Register contents when Exit 21 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0    Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit continue normal SMF queue processing.

4    Tells JES2 to ignore any other exit routines associated with this exit and to continue normal SMF queue processing.

8    Tells JES2 to terminate normal SMF queue processing.

Coded example

None provided.
Exit 22: Cancel/status

Function

This exit allows your installation to implement its own algorithms for job queue searching and for TSO/E CANCEL/STATUS. Your exit routine can perform its own search for a requested job or transaction program and indicate whether it has found the job, or it can let JES2 perform the standard search.

Environment

Task

JES2 main task. You must specify ENVRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 22 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $MIT, $PCE, $STAC, $XPL

Point of processing

This exit is taken just before searching the JES2 job queue for a “status” or “cancel” request in HASPSTAC of the JES2 main task. The exit is given control twice in HASPSTAC where HASPSTAC performs the cancel and status functions for the TSO/E user (STCSTART).

The cancel and status functions execute when a Status/Cancel block (STAC) is queued to the CCTCSHED FIFO queue in the HCCT. The cancel/status support routine performs this queuing. JES2 then issues a WAIT (against SJBSECBS) to wait for the completion of the cancel/status processing.

Programming considerations

1. The return code from your exit routine will cause HASPSTAC to pass back the proper return code to JES2. JES2 propagates that return code to TSO/E to issue the appropriate message.

2. For multiple cancel status requests, (your exit routine returned a return code of 12), HASPSTAC returns a 0 return code in the subsystem job block (SSJB). JES2 propagates that return code to TSO/E in SSOBRETN.

© Copyright IBM Corp. 1988, 2009
Exit 22

3. To end a multiple status request your exit routine must return a “0” JQE address in R1 and issue a return code of 12.

4. The $JCAN macro can be used in your exit routine.

5. Message IKJ56216I can be misleading. The second level message tells the user that the job queues were searched for job names consisting of the userid plus one character. You can code your exit so that the job queue is searched for all of the user’s jobs.

6. First level messages such as IKJ56190I, IKJ56192I, IJK56197I, and IJK56211I can also be misleading if the exit returned a JQE address in R1 and a return code of 12. The jobname in these messages is constructed by TSO/E using the TSO/E user’s userid and the last character of the job name in the JQE that was selected by this exit. Depending on the job(s) selected by the exit, the jobname(s) taken from the JQE may not begin with the userid; however, the jobid in the message(s) is correct for the job processed.

7. You can determine if JES2 invoked exit 22 to process a transaction program by determining if flag SJBFLGA is set to SJBATP. Otherwise, JES2 invoked exit 22 to process a batch job.

---

Register contents when Exit 22 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Eyecatcher</td>
</tr>
<tr>
<td></td>
<td>Maintenance Level</td>
</tr>
<tr>
<td></td>
<td>Exit Number</td>
</tr>
<tr>
<td></td>
<td>Version Number</td>
</tr>
<tr>
<td></td>
<td>Indicator byte</td>
</tr>
</tbody>
</table>

JES2 sets the indicator byte to one of the following bit settings:

- **X022FRST**  
  *First call to exit* Indicates a single cancel request or the first status request determined by examining the function bit (SACTFUNC) in the STAC.

- **X022MURE**  
  *Multiple recall* Indicates a multiple status recall request.

- **X022MUST**  
  *Multiple status overflow* Indicates a multiple status overflow condition. The buffer that holds the status information is too small.

<table>
<thead>
<tr>
<th>Register</th>
<th>Condition byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLCOND</td>
<td>XPLRESP</td>
</tr>
<tr>
<td></td>
<td>Response byte</td>
</tr>
</tbody>
</table>
The STAC, mapped by the $STAC macro, is in a data space. Perform $ARMODE ON before accessing the data and $ARMODE OFF after finishing the access.

X022STAC  Address of STAC
X022STAA  ALET of stack

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of the STATUS/CANCEL PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 22 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of the JQE for return codes of 8 and 12; otherwise not applicable</td>
</tr>
<tr>
<td>2-13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit, continue normal processing.</td>
</tr>
<tr>
<td>4</td>
<td>Tells JES2 to ignore any other exit routines associated with this exit and to continue normal processing.</td>
</tr>
<tr>
<td>8</td>
<td>Tells JES2 to process a single request.</td>
</tr>
<tr>
<td>12</td>
<td>Tells JES2 to process a multiple request.</td>
</tr>
<tr>
<td>16</td>
<td>Tells JES2 that the exit routine has done all the processing requested. HASPSTAC returns a code of 0.</td>
</tr>
<tr>
<td>20</td>
<td>Tells JES2 that the job is not found. HASPSTAC returns a code of 4.</td>
</tr>
<tr>
<td>24</td>
<td>Tells JES2 that an invalid combination was requested. HASPSTAC returns a code of 8.</td>
</tr>
<tr>
<td>28</td>
<td>Tells JES2 that jobs with the same job name were found. HASPSTAC returns a code of 12.</td>
</tr>
<tr>
<td>32</td>
<td>Tells JES2 that the status buffer is too small to hold all the data requested. HASPSTAC returns a code of 16.</td>
</tr>
<tr>
<td>36</td>
<td>Tells JES2 that the job was not queuing because it is on the output queue. HASPSTAC returns a code of 20.</td>
</tr>
</tbody>
</table>
Exit 22

40  Tells JES2 that an invalid cancel request was made. HASPSTAC returns a code of 28.

Note: RC 12 – 40 are only valid for this exit when called from label STCZEXIT (that is, R0=0 or 4 only).

44  Tells JES2 that the request should be failed for security reasons and SSCSAUTH should be returned to the SSI caller.

The returned code causes the correct message to be presented to the TSO/E interface. For multiple status requests (RC=12), register R1 must be returned with a zero to end the processing and cause the messages to be issued.

Coded example

None provided.
Exit 23: FSS job separator page (JSPA) processing

Function
This exit allows you to modify the user-dependent section of the job separator page data area (JSPA). When JES2 assigns an output group to a functional subsystem application (FSA), it also creates a JSPA to provide job- and data set-level information for that data set. The FSA uses this information to generate the job header, job trailer, and data set header for an output group.

The JSPA contains three sections. HASPFSSM fills in two of these sections, the JES-dependent section and common section, after this exit returns control to JES2. Therefore, HASPFSSM overwrites any modifications you make to these sections at that time. Use this exit to modify the user-dependent fields (JSPAUSR1 and JSPAUSR2) in the third section, only.

Recommendations for implementing Exit 23
You can use Exit 23 to suppress the assignment of a JESNEWS data set by:
1. Turning off the flag bit in the JOE information block (JIB) that indicates JESNEWS printing.
2. Setting a return code of 8 in register 15. This suppresses both the JESNEWS data set and the separator pages.

Environment

Task
Functional subsystem (HASPFSSM). You must specify ENVIRON=FSS on the $MODULE macro.

AMODE/RMODE requirements
RMODE ANY, AMODE 31

Supervisor/problem program
JES2 places Exit 23 in supervisor state and PSW key 1.

Recovery
No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine.

Job exit mask
Exit 23 is subject to suppression. You can suppress Exit 23 by either setting the 23rd bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the initialization stream.

Restrictions
You should ensure that your exit routine does not violate your installations security policy by:
- Overlapping the PSF-defined security label area
- Suppressing required separator pages.
Exit 23

Mapping macros normally required

$FSACB, $FSSCB, $HASPEQU, $HFCT, $JIB, JSPA, ETD, FSIP

Point of processing

This exit is invoked through the exit effector during GETDS processing. Whenever a new JIB is initialized during GETDS processing, Exit 23 is invoked in HASPFSSM. At this time, the associated $JCT, $IOT, and checkpoint records are read and the JSPA is built.

See “Programming Considerations” below for further coding requirements associated with this exit.

Programming considerations

1. A save-area type control block is obtained for use as the parameter list loaded into register 1 when control is passed to the exit routine.
2. The assignment of the JESNEWS data set can be checked in the $JOE information block ($JIB). The JIBFNEWS bit can be set or reset by the exit routine; however, if a return code of 8 is returned, the JESNEWS is not assigned; this is independent of the JIBFNEWS bit setting.
3. IAZFSIP maps the GETDS parameter list.
4. IAZJSPA maps the JSPA parameter list. Flag bit JSPA1UND, when on, indicates that the userid in field JSPCEUID is an undefined user.
5. Exit 23 routines should issue $SAVE after the $ENTRY macro and return to the exit effector using $RETURN. These routines also can call subroutines of their own which also use $SAVE/$RETURN logic.
6. This exit must be in common storage. Do not linkedit this exit to HASPFSSM.
7. Locating JCT Control Block Extensions
   If the $JCT address is contained in field JIBJCT, you can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 23 gets control

The contents of the register on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 5-word parameter list, having the following structure:</td>
</tr>
<tr>
<td></td>
<td>word1 (+0) JSPA address</td>
</tr>
<tr>
<td></td>
<td>word2 (+4) JIB address</td>
</tr>
<tr>
<td></td>
<td>word3 (+8) FSACB address</td>
</tr>
<tr>
<td></td>
<td>word4 (+12) FSSCB address</td>
</tr>
<tr>
<td></td>
<td>word5 (+16) GETDS parameter list address (IAZFSIP)</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HFCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>The address of an 18-word save area where the exit routine stores the exit effector’s registers</td>
</tr>
</tbody>
</table>
Register contents when Exit 23 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>Not applicable</td>
</tr>
<tr>
<td>2-14</td>
<td>Unchanged</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2, if additional exit routines are associated with this exit, to call the next consecutive exit routine. If no additional exit routines are associated with this exit a zero return code tells the FSA to produce any separator that has been defined by the installation based on the information contained in the JSPA.

4  Tells JES2 to ignore any additional exit routines associated with this exit. However, all other processing noted for return code 0 is accomplished.

8  Tells JES2 to unconditionally suppress production of the job separator page. The JESNEWS data set is not assigned.

12 Tells JES2 to unconditionally (that is, even if the printer has been set to S=N) produce any job separator page.

Coded example

Module HASX23A in SYS1.SHASSAMP contains a sample of Exit 23.
Exit 24: Post–initialization

Function

This exit allows you to make modifications to JES2 control blocks before JES2 initialization ends and to create and initialize control blocks that your installation defines for its own special purposes.

Environment

Task

JES2 Main Task (Initialization) – JES2 dispatcher disabled

The following JES2 initialization steps have been performed before your exit routine gets control. Essentially all JES2 initialization is done, but the JES2 warm start processor has not been dispatched yet to perform its initialization-like processing.

You must specify ENVIRON=JES2 on the $MODULE macro.

1. The JES2 initialization options are obtained from the operator or the PARM parameter on the EXEC statement and converted into status bits.
2. The JES2 initialization statement data set is read and processed.
3. The direct-access devices are scanned, and eligible spooling volumes are identified and allocated to JES2.
4. The spooling and checkpoint data sets are examined and initialized for JES2 processing.
5. The subsystem interface control blocks are constructed and initialized.
6. The unit-record devices, remote job entry lines, and network job entry lines are scanned; eligible and specified devices are located and allocated.
7. JES2 subtasks are attached, and exit routines are located.
8. SMF processing is started by generating a type 43 SMF record.
9. The JES2 control blocks, such as the HASP communications table (HCT), the device control tables (DCT), the data control blocks (DCB), the processor control elements (PCE), the data extent blocks (DEB), and the buffers (IOB), are constructed and initialized.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 24 in supervisor state and PSW key 1.

Recovery

JES2 does not have a recovery environment established at the processing point for Exit 24 (the JES2 ESTAE will process termination, but not recover).

Job exit mask

This exit is not subject to job exit mask suppression.
Exit 24

Mapping macros normally required
$CIRWORK, $HASPEQU, $HCT, $PCE

Point of processing
When Exit 24 is called, HASPIRA has called each JES2 initialization routine (IR) in turn to perform JES2 initialization. After all the IRs have successfully completed, HASPIRA calls the Exit 24 routine(s) before tracing the JES2 initialization and returning control to the HASJES20 load module (HASPNUC). On return from HASPINIT, HASPNUC deletes the HASPINIT load module (if not part of HASJES20) and passes control to the asynchronous input/output processor, $ASYNC, resulting in the dispatching of JES2 processors.

Creating an information string through Exit 24
This information string gives the installation the option of providing its own information to applications that request subsystem version information (through SSI code 54), and to override the information passed by JES2.

Information about defining keywords and values for information strings is provided in z/OS MVS Using the Subsystem Interface (in the discussion of SSI code 54).

Use the following steps to create an information string during JES2 initialization. (JES2 does not pass an information build area to Exit 24 during a hot start.)
1. Check the condition byte in field XPLCOND to ensure that the JES2 is warm starting, quick starting, cold starting, or restarting through a $E MEMBER RESTART command.
2. Check the information build area length in field X024SSWL to ensure that the area is large enough to accommodate the installation string. If the area is too small, ensure that Exit 24 bypasses the installation code that builds the string.
3. Obtain the pointer to the information build area from field X024SSIA, then move the installation string into the build area.
4. Initialize field X024SSIL with the length of the string.
5. Set flag X024RSSI in the XPL response byte to indicate that Exit 24 is supplying an information string before returning to JES2 initialization.

When JES2 processing validates the variable information string, the HASPIRA module obtains storage in ECSA. Then JES2 moves the variable information string from the build area pointed to by X024SSIA to extended common storage.

Programming considerations
1. The EXIT(nnn) statement for Exit 24 must specify STATUS=ENABLED for the exit; the $T EXIT(nnn) command cannot be used to enable (STATUS=ENABLED) the exit at a later time since the point of processing for Exit 24 is before the time at which the command processor is made functional.
2. Because Exit 24 is called from JES2 initialization, the JES2 dispatcher is not yet functional; so MVS protocol should be used in Exit 24 routines (for example, WAIT rather than $WAIT and ESTAE rather than $ESTAE).
3. If Exit 24 returns a return code of 8, HASPIRA issues message $HASP864 INITIALIZATION TERMINATED BY INSTALLATION EXIT 24. The $HASP428 message will also be issued before final termination.
4. Your exit routine can access JES2 control blocks through the HCT. Your exit routine can then access DCTs, PCEs, buffers, the UCT, and so on. for making modifications.

5. Your exit routine is responsible for establishing addressability to your own special control blocks. The HCT points to the optional user-defined UCT and other areas are provided in the HCT for various installation uses, identified by labels $USER1 through $USER5.

Register contents when Exit 24 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Parameter list eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level of $XPL parameter list</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit ID number</td>
</tr>
<tr>
<td>X024IND</td>
<td>Indicator byte: not applicable.</td>
</tr>
<tr>
<td>X024COND</td>
<td>Condition byte indicating the type of JES2 start in progress.</td>
</tr>
</tbody>
</table>

  | X024WARM   | Indicates single-system warm start. |
  | X024HOT    | Indicates hot start. |
  | X024QCK    | Indicates quick start. |
  | X024ALLS   | Indicates all-systems warm start. |
  | X024ESYS   | Indicates $E MEMBER restart. |
  | X024COLD   | Indicates cold start. |
  | X024IPL    | Indicates system has been IPLed. |
  | X024COFM   | Indicates cold start with format in progress. |
  | X024RESP   | Response byte |
  | X024SSIA   | Address of the information build area where the exit builds the SSI information string. The caller of EXIT 24 provides this area (set to zero during a JES2 hot start). |
  | X024SSWL   | Length of the information build area (the area pointed to by X024SSIA). The caller of Exit 24 provides this value. |

2-10      Not applicable

11        Address of $HCT

12        Not applicable

13        Address of $PCE: the PCE work area is the common initialization routine work area, mapped by the $CIRWORK macro.

14        Return address

15        Entry address
Register contents when Exit 24 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td><strong>XPLRESP</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X024RSSI</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X024SSIL</strong></td>
</tr>
<tr>
<td>2-13</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>Return Address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

| 0 | Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit continue the normal initialization process. |
| 4 | Tells JES2 to ignore any other exit routines associated with this exit and to continue normal initialization processing. |
| 8 | Tells JES2 to terminate normal initialization. This results in the $HASP864 error message to the operator. |

Coded example

Module HASX24A in SYS1.SHASSAMP contains a sample of Exit 24.
Exit 25: JCT read

Function

This exit allows you to provide an exit routine to receive control whenever a JES2 functional subsystem address space (HASPFS) performs JCT I/O. That is, your routine receives control just after the JCT is read into storage by the HASPFSSM module which executes as part of the FSS address space.

You can use this exit to perform I/O for any installation-specific control blocks you may have created.

Related exits

Whenever JCT I/O is performed by the JES2 main task, Exit 7 serves the purpose of this exit, and Exit 8 is used whenever a JES2 subtask or a routine running in the user environment performs JCT I/O.

Environment

Task

Functional subsystem (HASPFS). You must specify ENVIRON=FSS on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 25 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. As with every exit, you should provide your own recovery within your exit routine. The $ESTAE facility is inoperative within the FSS execution environment, rather the MVS ESTAE facility must be used to provide recovery. Also note that the FSS may have recovery routines in effect and that these depend on the FSS implementation.

Job exit mask

Exit 25 is subject to suppression. You can suppress Exit 25 by implementing exit 2 to set the 25th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$HASPEQU, $HFCT, $JCT, $JCTX, ETP, FSIP

Point of processing

This exit is taken from the functional subsystem address space (HASPFSM).
Exit 25

JES2 gives control to your exit routine after the $JCT has been read into storage, during $JIB initialization processing in the FSMGETDS routine of HASPFSSM if the $JCT read was successful and before initialization of the job separator page area (IAZJSPA) with fields from the $JCT. The $JCT read belongs to the job owning the JOE from which data set(s) will be selected for assignment to the FSA through the functional subsystem interface (FSI) GETDS function.

JES2 can also give control to your exit routine just after the FSMGETDS routine in HASPFSSM reads the JCT for the job owning the $JOE from which a data set will be selected (except if queuing on a setup request) for assignment to a functional subsystem application (FSA).

Programming considerations

1. Be sure your exit routines be in common storage. Do not linkedit this exit with HASPFSSM.
2. The $SAVE and $RETURN services are available in the FSS environment.
3. The service routines provided in the HASPFSSM module may be used within your exit routine. The cell pool services, $GETBLK and $RETBLK can be used to acquire save areas and other predefined storage cells dynamically. You are responsible for returning all storage cells explicitly acquired.
4. Locating JCT Control Block Extensions
   You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 25 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code passed to your routine by JES2</td>
</tr>
<tr>
<td>0</td>
<td>Indicates that the $JCT has been read from spool</td>
</tr>
<tr>
<td>4</td>
<td>Indicates that the $JCT will be written to spool</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $JCT</td>
</tr>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HFCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of an OS-style save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 25 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0 Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit...
routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4 Tells JES2 that even if there are additional exit routines associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

None provided.
Exit 26: Termination/resource release

Function

This exit allows you to free resources obtained during previous installation exit routine processing at any JES2 termination. At a JES2 termination (that is, $P JES2 command, JES2 initialization termination, or an abend), Exit 26 receives control to free whatever resources your exit routines continues to hold. To control the release of resources, this exit permits access to the termination recovery communication area (TRCA) and the HASP communications table (HCT). With such access available, your installation is provided sufficient flexibility to withdraw or free all services and resources you may have previously acquired. This exit can also be used to permit your installation to modify the termination options and edit operator responses to those options.

Environment

Task

JES2 main task (Termination) – JES2 dispatcher disabled. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 26 in supervisor state and PSW key 1.

Recovery

Exit 26 is protected by an ESTAE routine. If an error occurs during Exit 26 processing in your code, the ESTAE issues message $HASP082 INSTALLATION EXIT 26 ABEND to the operator. The ESTAE provides an SDUMP (if possible), returns control to JES2 termination processing ($HEXIT), and proceeds with normal termination. If this ESTAE does receive control, JES2 does not permit Exit 26 to receive control again.

Job exit mask

This exit point is not subject to job exit mask suppression.

Mapping macros normally required

$ERA, $HASPEQU, $HCCT, $HCT, $MIT, $PCE, $TRCA

Point of processing

This exit is taken from HASPTERM during JES2 termination processing ($HEXIT).

At JES2 termination, the operator receives the message $HASP098 ENTER TERMINATION OPTION. Following the operator response but before response processing, this exit gains control. At this time the exit has the option to change the operator’s reply to $HASP098. Exit processing completes, and on return from the exit, processing continues with the scanning of the operator response to the $HASP098 message.
Programming considerations

1. Be careful not to free private area storage (for example, the UCT) that might be needed by JES2 termination services after exit 26 processing. PCE tables and DTE tables, and so forth, may see UCT fields and might be needed later by HASPTERM.

2. The $CADDR (JES2 common storage address table) might not be available when Exit 26 is invoked.

Register contents when Exit 26 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code passed to your routine by JES2</td>
</tr>
<tr>
<td>0</td>
<td>Indicates that Exit 26 is invoked for the first time</td>
</tr>
<tr>
<td>4</td>
<td>Indicates that Exit 26 is invoked for other than the first time</td>
</tr>
<tr>
<td>1</td>
<td>Address of the JES2 main task $TRCA</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of the HASPTERM $PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 26 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code passed to your routine by JES2</td>
</tr>
<tr>
<td>0</td>
<td>Indicates that Exit 26 is invoked for the first time</td>
</tr>
<tr>
<td>4</td>
<td>Indicates that Exit 26 is invoked for other than the first time</td>
</tr>
<tr>
<td>1</td>
<td>Address of the JES2 main task TRCA</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of the HASPTERM $PCE – (this is a special PCE located in HASPTERM)</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code:

0        Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4        Tells JES2 that even if there are additional exit routines associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.
Coded example

None provided.
Exit 27: PCE attach/detach

Function

This exit allows resources to be allocated and deallocated. The exit also allows you to deny a PCE attach.

Environment

Task

JES2 main task. You must specify this task on the ENVIRON specification of the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 27 in supervisor state and PSW key 1.

Recovery

$ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

This exit point is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $MIT, $PCE

Point of processing

This exit is taken from HASPDYN either immediately after a PCE has been attached or immediately before a PCE is detached.

Programming considerations

None.

Register contents when Exit 27 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code passed to your routine by JES2</td>
</tr>
<tr>
<td>0</td>
<td>Indicates that Exit 27 is invoked after a PCE attach</td>
</tr>
<tr>
<td>4</td>
<td>Indicates that Exit 27 is invoked before a PCE is detached</td>
</tr>
</tbody>
</table>
Exit 27

1  Pointer to a 1-word parameter list that contains the address of the PCE to be processed.

2-10  N/A

11  Address of the HCT

12  N/A

13  Address of the PCE currently in control

14  The return address

15  The entry address

Register contents when Exit 27 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if there are additional exit routines associated with this exit, call the next consecutive exit routine. If there are no other exit routines associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if there are additional exit routines associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

8  Tells JES2 to detach the PCE that was attached immediately before invoking this exit.

Coded example

Module HASX27A in SYS1.SHASSAMP contains a sample of Exit 27.
Exit 28: subsystem interface (SSI) job termination

Function

This exit allows you to free resources (for example, storage for installation control blocks) that were obtained during Exit 32 (SSI Job Selection) processing. You can also use this exit (by changing the response byte) to either suppress the JES2 job termination-related message or replace them with your own installation-defined messages.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 28 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 28 is subject to suppression. You can suppress Exit 28 by either implementing exit 2 to set the 28th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$HASPEQU, $HCCT, $JCT, $JCTX, $MIT, $SJB

Point of processing

This exit is taken from HASCJBST before the freeing of job-related control blocks and the issuing of related messages.

Programming considerations

Changes of security information in the $JCT could cause a later security validation to fail. These changes could also be a violation of your installation’s security policy.

Expanding the JCT control block

You can add, expand, locate, and remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see [z/OS JES2 Macros](https://www.ibm.com/support/knowledgecenter/SSEK92_2.2.0/com.ibm.jes2.doc/).
Register contents when Exit 28 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a 12-byte parameter list with the following structure:</td>
</tr>
<tr>
<td>Byte 1 (+0)</td>
<td>A type-of-processing caller indicator, as follows:</td>
</tr>
<tr>
<td>0</td>
<td>job termination (JOB, STC, TSU, or XBM)</td>
</tr>
<tr>
<td>4</td>
<td>SYSLOG termination (return ID)</td>
</tr>
<tr>
<td>8</td>
<td>joblet termination</td>
</tr>
<tr>
<td>12</td>
<td>unsuccessful job selection (JOB, STC, TSU unable to obtain resources)</td>
</tr>
<tr>
<td>16</td>
<td>unsuccessful request ID JOB (request ID unable to obtain resources)</td>
</tr>
<tr>
<td>20</td>
<td>unsuccessful joblet selection (unable to obtain resources)</td>
</tr>
<tr>
<td>24</td>
<td>unsuccessful job restart (JOB RENQ unable to obtain resources)</td>
</tr>
<tr>
<td>Byte 2 (+1)</td>
<td>This byte is not part of the interface</td>
</tr>
<tr>
<td>Byte 3 (+2)</td>
<td>Response byte</td>
</tr>
<tr>
<td>Bits 0-6</td>
<td>These bits are not part of the interface</td>
</tr>
<tr>
<td>Bit 7</td>
<td>0 – indicates that JES2 will issue job termination message (default)</td>
</tr>
<tr>
<td></td>
<td>1 – indicates that JES2 will suppress job termination message</td>
</tr>
<tr>
<td>Byte 4 (+3)</td>
<td>This byte is not part of the interface</td>
</tr>
<tr>
<td>Byte 5 (+4)</td>
<td>Address of SJB or 0</td>
</tr>
<tr>
<td>Byte 9 (+8)</td>
<td>Address of JCT or 0</td>
</tr>
</tbody>
</table>

Register contents when Exit 28 passes control back to JES2

Upon return from this exit, the register contents must be:
Exit 28

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  
   Tells JES2 that if there are additional exit routines associated with this exit, call the next consecutive exit routine. If there are no other exit routines associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  
   Tells JES2 that even if there are additional exit routines associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASXJE in SYS1.SHASSAMP contains a sample of Exit 28.
Exit 29: Subsystem interface (SSI) end-of-memory

Function

This exit allows you to free resources in common storage (for example, installation control blocks that were obtained during Exit 32, SSI Job Selection, processing).

You can also use this exit to free resources on an address space level. Because this exit executes in the master scheduler address space, it can only process CSA-resident items.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 29 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

This exit point is not subject to job exit mask suppression.

Mapping macros normally required

$HASB, $HASPEQU, $HCCT, $MIT, $SJB

Point of processing

This exit is taken from HASCJBTR before the freeing of CSA job-related control blocks.

Programming considerations

None.

Register contents when Exit 29 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to an 8-byte parameter list with the following structure: Byte 1 (+0) This byte is not part of the interface</td>
</tr>
</tbody>
</table>
Exit 29

Byte 2 (+1)  Condition byte  

Bits 0-6   These bits are not part of the interface

Bit 7   0 – normal end-of-memory  
1 – abnormal end-of-memory

Byte 3 (+2)   This byte is not part of the interface
Byte 4 (+3)   This byte is not part of the interface
Byte 5 (+4)   This byte is not part of the interface
Byte 6 (+5)   This byte is not part of the interface
Byte 7 (+6)   Address space ID

2-10 Not applicable
11 Address of $HCCT
12 Not applicable
13 Address of an available save area
14 Return address
15 Entry address

Register contents when Exit 29 passes control back to JES2

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX29A in SYS1.SHASSAMP contains a sample of Exit 29.
Exit 30: Subsystem interface (SSI) data set OPEN and RESTART

Function

This exit allows you to get control during OPEN and RESTART processing of subsystem interface data sets. An indicator (passed to the exit in register 0) indicates either OPEN or RESTART processing; therefore, this exit can be used for either situation. Further, an indicator (passed in the parameter list pointed to by register 1) indicates the type of data set (SYSIN, SYSOUT, process SYSOUT, SPOOL BROWSE, or an internal reader type).

You can examine the data set characteristics and check them for validity, proper authority, or alter them.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 30 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect.

However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 30 is subject to suppression. You can suppress Exit 30 either by implementing exit 2 to set the 30th bit in the job exit suppression mask (JCTXMASK) or by including a statement in the initialization stream that disables Exit 30.

Mapping macros normally required

$HASPEQU, $HCCT, $IOT, $MIT, $PDDB, $SJB, DEB, JFCB

Point of processing

This exit is taken from HASCDSOC after the data set has been either OPENed or RESTARTed.

Programming considerations

1. Expanding the JCT Control Block
If the address of the $JCT is contained in field SJB, you can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

## Register contents when Exit 30 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Type of call indication</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a 28-byte parameter list with the following structure:</td>
</tr>
<tr>
<td>Byte 1 (+0)</td>
<td>Type of data set being processed</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Byte 2 (+1)</td>
<td>Condition byte</td>
</tr>
<tr>
<td>Bits 0-4</td>
<td>These bits are not part of the interface.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>0 – user authorization successful</td>
</tr>
<tr>
<td></td>
<td>1 – user authorization failed</td>
</tr>
<tr>
<td>Bit 6</td>
<td>0 – no error encountered</td>
</tr>
<tr>
<td></td>
<td>1 – error encountered</td>
</tr>
<tr>
<td>Bit 7</td>
<td>(applicable to data set OPEN for STC and TSU internal readers only)</td>
</tr>
<tr>
<td></td>
<td>0 – $P JES2 not in progress</td>
</tr>
<tr>
<td></td>
<td>1 – $P JES2 in progress</td>
</tr>
<tr>
<td>Byte 3 (+2)</td>
<td>Response byte</td>
</tr>
<tr>
<td>bits 0-5</td>
<td>These bits are not part of the interface.</td>
</tr>
<tr>
<td>bit 6</td>
<td>0 – open/restart the data set or reader. Default is 0 unless the data set type is unknown or if an error occurred while attempting to open the data set.</td>
</tr>
<tr>
<td></td>
<td>1 – fail the OPEN/RESTART processing</td>
</tr>
<tr>
<td>bit 7</td>
<td>0 – suppress unknown data set message ($HASP352). Zero is the default for this bit unless the type of data set is unknown.</td>
</tr>
<tr>
<td></td>
<td>1 – issue the unknown data set message ($HASP352)</td>
</tr>
<tr>
<td>Byte 4 (+3)</td>
<td>This byte is not part of the interface.</td>
</tr>
<tr>
<td>Byte 5 (+4)</td>
<td>Address of IRWD if internal reader data set (type 0, 4, 8 in byte 1 of parameter list)</td>
</tr>
<tr>
<td></td>
<td>Address of SDB if SYSIN, SYSOUT, PROCESS</td>
</tr>
</tbody>
</table>
SYSOUT, or SPOOL BROWSE data set (type 12, 16, 20, or 24 in byte 1 of parameter list)
0 if unknown data set file (type 28 in byte 1 of parameter list)

**Byte 9 (+8)** Address of SJB or 0

**Byte 13 (+12)** Address of JFCB

**Byte 17 (+16)** Address of DEB

**Byte 21 (+20)**
- 0 if internal reader data set (type 0, 4, 8 in byte 1 of parameter list) or if bits 6 and 7 of byte 2 (condition byte) are not 0
- Address of PDDB if SYSIN, SYSOUT, PROCESS SYSOUT, or SPOOL BROWSE data set (type 12, 16, 20, or 24 in byte 1 of parameter list)

**Byte 25 (+24)**
- 0 if internal reader data set (type 0, 4, 8 in byte 1 of parameter list) or if bits 6 and 7 of byte 2 (condition byte) are not 0
- Address of IOT if SYSIN, SYSOUT, PROCESS SYSOUT, or SPOOL BROWSE data set (type 12, 16, 20, or 24 in byte 1 of parameter list)

2-10 Not applicable
11 Address of HCCT
12 Not applicable
13 Address of an available save area
14 Return address
15 Entry address

---

### Register contents when Exit 30 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

- **0**
  - Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

- **4**
  - Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

---

### Coded example

Module HASXOCA in SYS1.SHASSAMP contains a sample of Exit 30.
Exit 31: Subsystem interface (SSI) allocation

Function

This exit allows you to receive control during allocation of subsystem interface data sets and internal readers. During allocation processing, JES2 can affect subsystem data set characteristics. This exit allows an installation to control how JES2 will process installation-specified statements and parameters during this processing phase.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 31 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery.

Job exit mask

Exit 31 is subject to suppression. You can suppress Exit 31 either by implementing exit 2 to set the 31st bit in the job exit suppression mask (JCTXMASK) or by indicating Exit 31 is disabled in the initialization stream.

Mapping macros normally required

$HASPEQU, $HCCT, $IOT, $MIT, $PDDB, $SJB, JFCB

Point of processing

This exit is taken from HASCDSAL after allocation processing but before return to the SSI caller.

Programming considerations

The following are programming considerations for Exit 31.

1. You can determine whether Exit 31 was invoked on behalf of a transaction program or batch job by either:
   • Determining if flag SJBLGFA is set to SJBLTG
   • Determining if the IOT contains a DSCT

2. Expanding the JCT Control Block
   If the address of the $JCT is contained in field SJBJCT, you can add, expand, locate, or remove extensions to the job control table ($JCT) control block from...
this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see [z/OS JES2 Macros](#).

## Register contents when Exit 31 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td><strong>Field Name</strong></td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXLEV</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>X031ID</td>
</tr>
<tr>
<td></td>
<td>X031COND</td>
</tr>
<tr>
<td></td>
<td>X031ERR</td>
</tr>
<tr>
<td></td>
<td>X031RESP</td>
</tr>
<tr>
<td></td>
<td>X031FAIL</td>
</tr>
<tr>
<td></td>
<td>X031DSTY</td>
</tr>
<tr>
<td></td>
<td>X031INTR</td>
</tr>
<tr>
<td></td>
<td>X031JSNW</td>
</tr>
<tr>
<td></td>
<td>X031SYIN</td>
</tr>
<tr>
<td></td>
<td>X031SYSO</td>
</tr>
<tr>
<td></td>
<td>X031PSPI</td>
</tr>
<tr>
<td></td>
<td>X031SDSB</td>
</tr>
<tr>
<td></td>
<td>X031UNK</td>
</tr>
<tr>
<td></td>
<td>X031SDB</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X031SJB</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X031JFCB</td>
</tr>
</tbody>
</table>
Exit 31

X031PDDB
Address of PDDB or zero

X031IOT
Address of IOT or zero

1  Pointer to type of data set being processed (X031DSTY)
2-10 N/A
11  Address of HCCT
12  N/A
13  Address of an available save area
14  The return address
15  The entry address

Register contents when Exit 31 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX31A in SYS1.SHASSAMP contains a sample Exit 31.
Exit 32: Subsystem interface (SSI) job selection

Function

This exit allows you to receive control during job selection processing. You can perform job-related processing such as allocating resources and I/O for installation-defined control blocks. Also, this exit can be used to suppress job selection related messages and replace them with installation-defined messages. Such messages can indicate, for example, that a job is “not to be selected for execution” and “the initiators were terminated”.

Related exits

Use Exit 28 (SSI Job Termination) and Exit 29 (SSI End-of-Memory) with Exit 32 to perform job termination processing.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places Exit 32 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 32 is subject to suppression. You can suppress Exit 32 by either implementing exit 2 to set the 32nd bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$HASPEQU, $HCCT, $JCT, $JCTX, $MIT, $SJB

Point of processing

This exit is taken from HASCJBST following job selection but before the issuing of the $HASP373 JOBID $HASP373 jobname STARTED message.

Programming considerations

1. Expanding the JCT Control Block
You can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 32 gets control

0 0  Pointer to an 12-byte parameter list with the following structure:

Byte 1 (+0)  Type of processing indicator
0  Reserved
4  Request for job by SYSLOG ID
8  Request for job by class
12  TSU
16  STC

Byte 2 (+1)  Condition byte
bits 0-6  These bits are not part of the interface
bit 7  0 – no error occurred during processing (job selectable for execution)

1 – error occurred during job select processing (job is to be restarted or terminated)

Byte 3 (+2)  Response byte
bits 0-3  These bits are not part of the interface
bit 4  0 – initiator is not abnormally ended (default)
1 – initiator is abnormally ended, then restarted automatically.

bit 5  0 – initiator is not abnormally ended (default)
1 – initiator is abnormally ended

Notes:
1. If you specify both bits 4 and 5, the initiator is not automatically ended and drained.
2. The initiator will stop after the job currently being processed has been terminated/queued for RESTART.
3. This bit is ignored unless the type of processing is a job request by class (R1, byte 1 = 8)

bit 6  0 – select this job (default)
1 – terminate this job

Note: This bit is ignored if the condition byte (byte 2) is nonzero
Exit 32

bit 7
0 – issue the JES2 job selection
($HASP373) message
1 – suppress the JES2 job
selection ($HASP373) message

Note: This bit is ignored if the
condition byte (byte 2) is
nonzero

Byte 4 (+3) This byte is not part of the interface
Byte 5 (+4) Address of SJB
Byte 9 (+8) Address of JCT or 0

2-10 N/A
11 Address of HCCT
12 N/A
13 Address of an available save area
14 Return address
15 Entry address

Register contents when Exit 32 passes control back to JES2

0-13 Unchanged
14 Return address
15 Return code

A return code of:
0
Tells JES2 that if additional exit routines are associated with this
exit, call the next consecutive exit routine. If no other exit routines
are associated with this exit, continue with normal processing,
which is determined by the particular exit point from which the exit
routine was called.

4
Tells JES2 that even if additional exit routines are associated with
this exit, ignore them; continue with normal processing, which is
determined by the particular exit point from which the exit routine
was called.

Coded example

Module HASX32A in SYS1.SHASSAMP contains a sample of Exit 32.
Exit 33: Subsystem interface (SSI) data set CLOSE

Function

This exit allows you to receive control during subsystem data set CLOSE processing. You can examine the data set characteristics and check them for validity, authority, or alter the characteristics. An indicator, passed to this exit in the parameter list pointed to by register 1, indicates the type of data set.

Related exits

Use Exit 30 (SSI Data Set OPEN and RESTART) in conjunction with Exit 33 to perform required data set OPEN and RESTART processing.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 33 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 33 is subject to suppression. You can suppress Exit 33 by setting the 33rd bit in the job exit suppression mask (JCTXMASK) or by indicating Exit 33 is disabled in the initialization stream.

Mapping macros normally required

$DCT, $HASPEQU, $HCCT, $IOT, $MIT, $PDDB, $SDB, $SJB, DEB, JFCB

Point of processing

This exit is taken from HASCDSOC before the CLOSE of the subsystem data set.

Programming considerations

1. Expanding the JCT Control Block
Exit 33

If the address of the $JCT is contained in field SJBJCT, you can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 33 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a 25-byte parameter list with the following structure:</td>
</tr>
<tr>
<td>Byte 1 (+0)</td>
<td>Type of data set indicator</td>
</tr>
<tr>
<td>0</td>
<td>JOB internal reader</td>
</tr>
<tr>
<td>4</td>
<td>STC internal reader</td>
</tr>
<tr>
<td>8</td>
<td>TSU internal reader</td>
</tr>
<tr>
<td>12</td>
<td>SYSIN data set</td>
</tr>
<tr>
<td>16</td>
<td>SYSOUT data set</td>
</tr>
<tr>
<td>20</td>
<td>PROCESS SYSOUT data set</td>
</tr>
<tr>
<td>24</td>
<td>SPOOL BROWSE data set</td>
</tr>
<tr>
<td>28</td>
<td>Unknown data set type</td>
</tr>
</tbody>
</table>

| Byte 2 (+1) | Condition byte |
| bits 0-6    | These bits are not part of the interface |
| bit 7       | 0 – no error occurred during CLOSE processing |
|             | 1 – error occurred during CLOSE processing |

| Byte 3 (+2) | Response byte |
| bits 0-5    | These bits are not part of the interface |
| bit 6       | 0 – CLOSE the data set or internal reader (default, unless data set type unknown, byte 1 = 28) |
|             | 1 – fail CLOSE processing |
| bit 7       | 0 – suppress the JES2 unknown data set type ($HASP3) message (default, unless data set type unknown, byte 1 = 28) |
|             | 1 – issue the JES2 unknown data set type ($HASP3) message |

| Byte 4 (+3) | This byte is not part of the interface |
| Byte 5 (+4) | |
| • Address of IRWD - if data set type is internal reader (byte 1 = 0, 4, or 8) |
| • Address of SDB - if data set type is SYSIN, SYSOUT, PROCESS SYSOUT, SPOOL BROWSE, unknown data set (byte 1 = 12, 16, 20, 24, or 28) or 0 |

| Byte 9 (+8) | Address of SJB or 0 |
| Byte 13 (+12) | Address of JFCB |
| Byte 17 (+16) | Address of DEB |
| Byte 21 (+20) | 0 if data set type is internal reader (byte 1 = 0, 4, or 8) or if byte 2 is nonzero |
|             | Address of PDDB if data set type is SYSIN,
Exit 33

SYSOUT, PROCESS SYSOUT, SPOOL BROWSE data set, or unknown (byte 1 = 12, 16, 20, 24, or 28)

**Byte 25 (+24)**
- 0 if data set type is internal reader (byte 1 = 0, 4, or 8) or if bit 7 of byte 2 is nonzero
- Address of IOT if data set type is SYSIN, SYSOUT, PROCESS SYSOUT, SPOOL BROWSE data set, or unknown (byte 1 = 12, 16, 20, 24, 28)

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of HCCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of an available save area</td>
</tr>
<tr>
<td>14</td>
<td>The return address</td>
</tr>
<tr>
<td>15</td>
<td>The entry address</td>
</tr>
</tbody>
</table>

**Register contents when Exit 33 passes back control to JES2**

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.</td>
</tr>
<tr>
<td>4</td>
<td>Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.</td>
</tr>
</tbody>
</table>

**Coded example**

Module HASXOCA in SYS1.SHASSAMP contains a sample of Exit 33.
Exit 34: Subsystem interface (SSI) data set unallocation

Function

This exit allows you to receive control during unallocation processing of subsystem interface data sets and internal readers.

Related exits

Use Exit 34 in conjunction with Exit 31 (SSI Data Set Allocation) to perform required data set unallocation processing.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 34 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 34 is subject to suppression. You can suppress Exit 34 by either implementing exit 2 to set the 34th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$DCT, $HASPEQU, $HCCT, $IOT, $MIT, $PDDB, $SDB, $SJB, JFCB

Point of processing

This exit is taken from HASCDSAL before the processing to unallocate the data set.

Programming considerations

When this exit routine returns control to JES2, JES2 updates certain characteristics of the data set being allocated with information in the SSOB extension, eliminating any changes you might have made to the PDDB in this exit. To have a permanent effect, you should make any changes to the data set characteristics in the SSOB extensions.

1. Expanding the JCT Control Block
If the address of the $JCT is contained in field SJBJCT, you can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS Macros.

Register contents when Exit 34 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a 24-byte parameter list with the following structure:</td>
</tr>
<tr>
<td></td>
<td>Byte 1 (+0) Type of data set indicator</td>
</tr>
<tr>
<td></td>
<td>0 - Internal reader</td>
</tr>
<tr>
<td></td>
<td>4 - JESNEWS data set</td>
</tr>
<tr>
<td></td>
<td>8 - SYSIN data set</td>
</tr>
<tr>
<td></td>
<td>12 - SYSOUT data set</td>
</tr>
<tr>
<td></td>
<td>16 - PROCESS SYSOUT or SYSOUT application program interface (SAPI) data set</td>
</tr>
<tr>
<td></td>
<td>20 - SPOOL BROWSE data set</td>
</tr>
<tr>
<td></td>
<td>24 - Unknown data set type</td>
</tr>
<tr>
<td></td>
<td>Byte 2 (+1) Condition byte</td>
</tr>
<tr>
<td></td>
<td>bits 0-5 These bits are not part of the interface</td>
</tr>
<tr>
<td></td>
<td>bit 6 0 – no error occurred during allocation processing</td>
</tr>
<tr>
<td></td>
<td>1 – error occurred during allocation processing</td>
</tr>
<tr>
<td></td>
<td>bit 7 0 – no error occurred during unallocation processing</td>
</tr>
<tr>
<td></td>
<td>1 – error occurred during unallocation processing</td>
</tr>
<tr>
<td></td>
<td>Byte 3 (+2) This byte is not part of the interface</td>
</tr>
<tr>
<td></td>
<td>Byte 4 (+3) This byte is not part of the interface</td>
</tr>
<tr>
<td></td>
<td>Byte 5 (+4) This byte is</td>
</tr>
<tr>
<td></td>
<td>• Address of IRWD - if data set type is internal reader (byte 1 = 0)</td>
</tr>
<tr>
<td></td>
<td>• Address of SDB - if data set type is SYSIN, SYSOUT, PROCESS SYSOUT, or SPOOL BROWSE data set (byte 1 = 8, 12, 16, or 20)</td>
</tr>
<tr>
<td></td>
<td>• 0- if unknown data set type (byte 1 = 24)</td>
</tr>
<tr>
<td></td>
<td>Byte 9 (+8) Address of SJB or 0. This value is 0:</td>
</tr>
<tr>
<td></td>
<td>• If error in obtaining SJB address,</td>
</tr>
<tr>
<td></td>
<td>• If data set is a started task or TSO/E internal reader, or</td>
</tr>
<tr>
<td></td>
<td>• When the automatic restart manager unallocates an internal reader.</td>
</tr>
<tr>
<td></td>
<td>Byte 13 (+12) Address of JFCB</td>
</tr>
<tr>
<td></td>
<td>Byte 17 (+16) Address of PDDB</td>
</tr>
<tr>
<td></td>
<td>0 – if data set type is a regular internal reader, an unknown data set type (byte 1 = 0 or 24), or if the PSO unallocation was performed after the JOB–step TCB ended.</td>
</tr>
</tbody>
</table>
Exit 34

Byte 21 (+20)  Address of IOT
               0 – if data set type is a regular internal reader or
               unknown data set type (byte 1 = 0 or 24)

2-10          N/A
11            Address of HCCT
12            N/A
13            Address of an available save area
14            The return address
15            The entry address

Register contents when Exit 34 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if additional exit routines are associated with this
    exit, call the next consecutive exit routine. If no other exit routines
    are associated with this exit, continue with normal processing,
    which is determined by the particular exit point from which the exit
    routine was called.

4  Tells JES2 that even if additional exit routines are associated with
    this exit, ignore them; continue with normal processing, which is
    determined by the particular exit point from which the exit routine
    was called.

Coded example

Module HASX34A in SYS1.SHASSAMP contains a sample of Exit 34.
Exit 35: Subsystem interface (SSI) end-of-task

Function

This exit allows you to free resources at the task level during end-of-task processing.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 35 in supervisor state and PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

This exit point is not subject to job exit mask suppression.

Mapping macros normally required

$HASB, $HASPEQU, $HCCT, $MIT, $SJB

Point of processing

This exit is taken from HASCJBTR after JES2 has located and locked the SJB (subsystem job block).

Programming considerations

1. Expanding the JCT Control Block

If the address of the $JCT is contained in field SJBJCT, you can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 35 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Exit 35

1  Pointer to a 20-byte parameter list with the following structure:

   Byte 1 (+0)  This byte is not part of the interface
   Byte 2 (+1)  Condition byte

      bits 0-6  These bits are not part of the interface
      bit 7    0 – task ended normally
               1 – task ended abnormally

   Byte 3 (+2)  This byte is not part of the interface
   Byte 4 (+3)  This byte is not part of the interface
   Byte 5 (+4)  This byte is not part of the interface
   Byte 6 (+5)  This byte is not part of the interface
   Byte 7 (+6)  Address space ID
   Byte 11 (+8) Address of SJB
   Byte 13 (+12) Address of primary IOT or 0
   Byte 17 (+16) Address of JCT or 0

2-10  N/A
11     Address of HCCT
12     N/A
13     Address of an available save area
14     The return address
15     The entry address

Register contents when Exit 35 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASXJEA in SYS1.SHASSAMP contains a sample of Exit 35.
Exit 36: Pre-security authorization call

Function

This exit allows you to modify information passed to the security authorization facility (SAF) of MVS. $SEAS invokes this exit just before passing control to SAF. You can:

- Bypass the default SAF call and perform your own security checking.
- Do additional security checking besides what SAF provides.
- Pass your own return and reason code to the invoker in place of the standard SAF return code.
- Pass information from JES2 to the security subsystem.
- Disable specific SAF security checking.

Environment

Task

USER environment. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 36 in supervisor state and PSW key 0.

Recovery

Recovery for this exit depends on the environment that invokes the exit:

Main task  If general purpose subtasks are attached then the subtask ESTAE is in effect. If no general purpose subtasks are attached and you specified UNCOND=YES, then the $SUBIT $ESTAE is in effect.

FSS  ESTAE recovery is in effect.

USER  JES2 fails the request and SSI $ESTAE recovery is in effect.

However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Table 9 on page 230 shows which function codes are subject to job mask suppression. (See the register one byte that is mapped by X036IND in “Register Contents when Exit 36 Gets Control”.)

Mapping macros normally required

$HASPEQU, $HCCT, $WAVE, $XPL
Point of processing

JES2 takes this exit before issuing the SAF call.

Programming considerations

- Use care when changing or restricting the functions that build, obtain, or extract information for tokens because you could cause later SAF calls to fail.
- If you need a finer level of control you will have to build more specific entity names in this exit. For example, if you want only certain operators to change the routing of a printer:
  - Define a more specific profile to RACF. For example, if you wanted to keep operators from changing the routing of jobs on JESC, you would define a profile named:
    ```
    JESC.MODIFY.JOBOUTROUTE
    ```
  - with only the operators you want to issue the command on the list of userids authorized to the command.
  - Intercept the command authorization call in Exit 36.
  - In Exit 36, scan the command and build the required profile name. The address of the command and the profile JES2 is requesting authorization for is in the $WAVE.
  - Replace the entity name (profile name) pointed to by the $WAVE with the more specific entity name.
  - If you code Exit 36 or Exit 37, you can pass a RACF request type to the exit. JES2 can request a branch entry extract to extract information from SECLABEL profiles (WAVREQUEST field set to WAVRXTRB). In addition, JES2 also uses the RACF extract (non-branch entry) to extract SECLABELs from various other profiles (WAVREQUEST field set to WAVRXTRT). New function codes (38 and 39) are defined for all these requests; see Table 9 on page 230.

- Locating Extensions to the JCT Control Block: You can use the $JCTXGET macro to locate extensions to the job control table ($JCT) control block from this exit.
- If you need to pass information from JES2 to the security subsystem, move the JCT pointer from the $SAFINFO parameter list (SFIJCT) to the SAF parameter list (ICHSAFP) in field SAFPUSRW to access the SAF router exit.
- If you include code (such as a branch table) based on the security function codes presented in Table 9 on page 230 be certain you also see the source of these function codes contained in macro $HASPEQU for their current and complete listing.

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Decimal Value</th>
<th>Symbolic Name</th>
<th>Meaning</th>
<th>Related Control Block</th>
<th>Job Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$SEANJES</td>
<td>Reserved for user code</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>$SEAINIT</td>
<td>Initialize security environment</td>
<td>SFI</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$SEAVERC</td>
<td>Security environment create</td>
<td>JCT</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$SEAVERD</td>
<td>Security environment delete</td>
<td>JCT</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Table 9. Security Function Codes
<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Symbolic Name</th>
<th>Meaning</th>
<th>Related Control Block*</th>
<th>Job Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$SEAXTRT</td>
<td>Extract security information for this environment</td>
<td>SJB</td>
<td>**</td>
</tr>
<tr>
<td>5</td>
<td>$SEASIC</td>
<td>SYSIN data set create</td>
<td>IOT</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>$SEASOC</td>
<td>SYSOUT data set create</td>
<td>IOT</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>$SEASIP</td>
<td>SYSIN data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>$SEASOP</td>
<td>SYSOUT data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>$SEAPSO</td>
<td>Process SYSOUT data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>$SEAPSS</td>
<td>Process SYSOUT data set select</td>
<td>PSO</td>
<td>No</td>
</tr>
<tr>
<td>11</td>
<td>$SEATCAN</td>
<td>TSO/E cancel</td>
<td>JCT</td>
<td>No</td>
</tr>
<tr>
<td>12</td>
<td>$SEACMD</td>
<td>Command authorization</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>$SEAPRT</td>
<td>Printer data set select</td>
<td>PDDB</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>$SEADEL</td>
<td>Data set purge</td>
<td>IOT</td>
<td>**</td>
</tr>
<tr>
<td>15</td>
<td>$SEANUSE</td>
<td>Notify user token extract</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>$SEATBLD</td>
<td>Token build</td>
<td>SWEL</td>
<td>No</td>
</tr>
<tr>
<td>17</td>
<td>$SEARJES</td>
<td>RJE signon, NJE source for command authorization</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>18</td>
<td>$SEADEVA</td>
<td>Device authorization</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>19</td>
<td>$SEANJEA</td>
<td>NJE SYSOUT data set create</td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>$SEAREXT</td>
<td>Re-verify token extract</td>
<td>JCT</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>---</td>
<td>Reserved</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>22</td>
<td>$SEANEWS</td>
<td>Update of JESNEWS</td>
<td>SJB</td>
<td>No</td>
</tr>
<tr>
<td>23</td>
<td>$SEANWBL</td>
<td>Build JESNEWS token</td>
<td>IOT</td>
<td>No</td>
</tr>
<tr>
<td>24</td>
<td>$SEAVERS</td>
<td>Subtask to create access control environment element (ACEE) for general subtasks</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>25</td>
<td>$SEAAUD</td>
<td>Audit for job in error</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>26</td>
<td>$SEADCHK</td>
<td>Authorization for $DESTCHK</td>
<td>DCW</td>
<td>No</td>
</tr>
<tr>
<td>27</td>
<td>$SEATSOC</td>
<td>SYSOUT data set create for trace</td>
<td>IOT</td>
<td>No</td>
</tr>
<tr>
<td>28</td>
<td>$SEASSOC</td>
<td>SYSOUT data set create for system job data sets (for example, JOBLOG)</td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>$SEANSOC</td>
<td>SYSOUT data set create for JESNEWS</td>
<td>IOT</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>$SEASOX</td>
<td>Transmit or offload of SYSOUT</td>
<td>PCE</td>
<td>Yes</td>
</tr>
<tr>
<td>31</td>
<td>$SEANJEV</td>
<td>VERIFYX for receive or reload of SYSOUT</td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>$SEAJOX</td>
<td>Transmit or offload of job</td>
<td>PCE</td>
<td>Yes</td>
</tr>
<tr>
<td>33</td>
<td>---</td>
<td>Reserved</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>34</td>
<td>$SEASPBO</td>
<td>Spool browse data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>35</td>
<td>$SEASEFS</td>
<td>Scheduler service, TOKNXTR</td>
<td>SSW</td>
<td>No</td>
</tr>
<tr>
<td>36</td>
<td>$SEASSWM</td>
<td>SWM modify ALTER AUTH</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 9. Security Function Codes (continued)

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Symbolic Name</th>
<th>Meaning</th>
<th>Related Control Block*</th>
<th>Job Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>$SEASAPI</td>
<td>SYSDOUT application programming interface</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>38</td>
<td>$SEASCLA</td>
<td>SECLABEL affinity extract</td>
<td>JQE</td>
<td>No</td>
</tr>
<tr>
<td>39</td>
<td>$SEASCLE</td>
<td>DCT SECLABEL extract</td>
<td>DCT or NIT</td>
<td>No</td>
</tr>
<tr>
<td>40</td>
<td>$SEANSON</td>
<td>Secure NJE signon SAF profiles for secure NJE signon</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>41</td>
<td>$SEADIRA</td>
<td>SECLABEL dominance</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>42</td>
<td>$SEASPLR</td>
<td>SPOOL I/O AUTH check</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>43-255</td>
<td>---</td>
<td>Not currently in use</td>
<td>Not in use</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. * Your exit routine should always check for the presence of the control block before using fields in the control block. Currently, the control block is not present when the $SEAXTRT function occurs during an open of TSU or STC internal readers.
2. ** Job exit mask suppression not in effect during selected processing.

Register contents when Exit 36 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

Field Name | Description |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number</td>
</tr>
<tr>
<td>X036IND</td>
<td>Indicator byte that contains the function code (value of FUNCODE=) passed by $SEAS. See Table 9 on page 230 for these function codes and their meanings.</td>
</tr>
<tr>
<td>X036COND</td>
<td>Condition byte showing the type of code that invoked the exit.</td>
</tr>
<tr>
<td>X036JES2</td>
<td>IBM-supplied code (CODER=JES2 on $SEAS).</td>
</tr>
<tr>
<td>X036USER</td>
<td>Customer-written code (CODER=USER on $SEAS).</td>
</tr>
<tr>
<td>X036RESP</td>
<td>Response byte you set to have the following meanings:</td>
</tr>
<tr>
<td>X036NORC</td>
<td>Setting this bit on in the response byte indicates that the exit-specified return and reason codes will be</td>
</tr>
</tbody>
</table>
Exit 36

2-10
11
12
13
14
15

N/A
Address of HCCT
N/A
Address of an available save area.
Return address
Entry address

Register contents when Exit 36 passes control back to JES2

Upon return from this exit, the register contents must be:

Register  | Contents             
-----------|----------------------
0-13       | N/A                  
14         | Return address       
15         | Return code          

A return code of:

0          | Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines
Exit 36

are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX36A in SYS1.SHASSAMP contains a sample of Exit 36.
Exit 37: Post-security authorization call

Function

This exit allows you to examine or modify return codes from the security authorization facility (SAF) of MVS. JES2 invokes this exit just before returning control to $SEAS. You can also perform additional security checking or other action based on the return code received. For example, you can:

- Notify the operator of the status of a request.
- Request confirmation of a request from the operator before continuing.
- Further restrict the criteria used to allow (or disallow) access.
- Call $SEAS again with new information.

Environment

Task

USER environment. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 37 in supervisor state and PSW key 0.

Recovery

Recovery for this exit depends on the environment that invokes the exit:

Main task

If general purpose subtasks are attached then the subtask ESTAE is in effect. If no general purpose subtasks are attached and you specified UNCOND=YES, then the $SUBIT $ESTAE is in effect.

FSS

ESTAE recovery is in effect.

USER

JES2 fails the request and SSI $ESTAE recovery is in effect.

However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 37 is subject to job exit mask suppression for function codes 5, 6, 7, 8, 9, 14, and 19. [Table 10 on page 237] shows which function codes are subject to job mask suppression. (See Byte 8 of 1 in “Register Contents when Exit 37 Gets Control”).

Mapping macros normally required

$HASPEQU, $HCCT, $WAVE, $XPL

Point of processing

This exit is taken from HASCSRIC after returning from the SAF call.
Exit 37

Programming considerations

- Use care when changing or restricting the functions that build, obtain, or extract information for tokens because you could cause later SAF calls to fail.
- **Locating Extensions to the JCT Control Block**: You can use the $JCTXGET macro to locate extensions to the job control table ($JCT) control block from this exit.
- If you include code (such as a branch table) based on the security function codes presented in Table 9 on page 230 be certain you also see the source of these function codes contained in macro $HASPEQU for their current and complete listing.
- If you code Exit 36 or Exit 37, you can pass a RACF request type to the exit. JES2 can request a branch entry extract to extract information from SECLABEL profiles (WAVREQUEST field set to WAVRXTRB). In addition, JES2 also uses the RACF extract (non-branch entry) to extract SECLABELs from various other profiles (WAVREQUEST field set to WAVRXTRT). Function codes 38 and 39 are defined for all these requests; see Table 9 on page 230.

Register contents when Exit 37 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number</td>
</tr>
<tr>
<td>X037IND</td>
<td>Indicator byte that contains the function code (value of FUNCODE=) passed by $SEAS. See Table 10 on page 237 for these function codes and their meanings.</td>
</tr>
<tr>
<td>X037COND</td>
<td>Condition byte showing the type of code that invoked the exit.</td>
</tr>
<tr>
<td>X037JES2</td>
<td>IBM-supplied code (CODER=JES2 on $SEAS).</td>
</tr>
<tr>
<td>X037USER</td>
<td>Customer-written code (CODER=USER on $SEAS).</td>
</tr>
<tr>
<td>X037RESP</td>
<td>Response byte you set to have the following meaning:</td>
</tr>
<tr>
<td>X037NORC</td>
<td>Setting this bit on in the response byte indicates that the exit-specified return and reason codes will be used. Otherwise, the SAF return code and reason code will be used.</td>
</tr>
<tr>
<td>X037PLUS</td>
<td>Exit 37 parameter list</td>
</tr>
<tr>
<td>X037PARM</td>
<td>Address of the parameter list, in the Work Access Verification Element ($WAVE), to pass to SAF. This address allows you to alter any parameters contained in the parameter list. However, do not...</td>
</tr>
</tbody>
</table>
change the address in this fullword field as SAF will not get the expected parameters.

**X037WAVE**  
Address of the $WAVE. This address allows you to alter any information contained in the $WAVE. However, do not change the address in this fullword field because you might not point to a valid $WAVE.

**X037RCBN**  
4-character identifier of related control block.

**X037RCBA**  
Address of related control block. If a control block is not related with this request, the address is zero.

**X037RETC**  
Fullword return code from exit routine. The exit passes the return code you set here to the caller in place of the SAF return code if bit 6 of byte 10 is a 1.

**X037RSNC**  
Fullword reason code from exit routine. The exit passes this reason code you set here to the caller in place of the SAF reason code if bit 6 of byte 10 is a 1.

**X037SIZE**  
Size of parameter list for Exit 37

| 2-10 | N/A                        |
| 11   | Address of HCCT            |
| 12   | N/A                        |
| 13   | Address of an available save area. |
| 14   | Return address             |
| 15   | Entry address              |

Table 10. Security Function Codes

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Symbolic Name</th>
<th>Meaning</th>
<th>Related Control Block*</th>
<th>Job Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$SEANJES</td>
<td>Reserved for user code</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>$SEAINIT</td>
<td>Initialize security environment</td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>$SEAVERC</td>
<td>Security environment create</td>
<td>JCT</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>$SEAVERD</td>
<td>Security environment delete</td>
<td>JCT</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>$SEAXTRT</td>
<td>Extract security information for this environment</td>
<td>SJB</td>
<td>**</td>
</tr>
<tr>
<td>5</td>
<td>$SEASIC</td>
<td>SYSIN data set create</td>
<td>IOT</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>$SEASOC</td>
<td>SYSOUT data set create</td>
<td>IOT</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>$SEASIP</td>
<td>SYSIN data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>$SEASOP</td>
<td>SYSOUT data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>$SEAPSO</td>
<td>Process SYSOUT data set open</td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>$SEAPSS</td>
<td>Process SYSOUT data set select</td>
<td>PSO</td>
<td>No</td>
</tr>
<tr>
<td>11</td>
<td>$SEATCAN</td>
<td>TSO/E cancel</td>
<td>JCT</td>
<td>No</td>
</tr>
<tr>
<td>12</td>
<td>$SEACMD</td>
<td>Command authorization</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>$SEAPRT</td>
<td>Printer data set select</td>
<td>PDDB</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>$SEADEL</td>
<td>Data set purge</td>
<td>IOT</td>
<td>**</td>
</tr>
<tr>
<td>15</td>
<td>$SEANUSE</td>
<td>Notify user token extract</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 10. Security Function Codes (continued)

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Decimal Value</th>
<th>Symbolic Name</th>
<th>Meaning</th>
<th>Related Control Block*</th>
<th>Job Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>$SEATBLD</td>
<td>Token build</td>
<td></td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>17</td>
<td>$SEARJES</td>
<td>RJE signon, NJE source for command authorization</td>
<td></td>
<td>SWEL</td>
<td>No</td>
</tr>
<tr>
<td>18</td>
<td>$SEADEVA</td>
<td>Device authorization</td>
<td></td>
<td>PCE</td>
<td>**</td>
</tr>
<tr>
<td>19</td>
<td>$SEANJEA</td>
<td>NJE SYSOUT data set create</td>
<td></td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>$SEAREXT</td>
<td>Re-verify token extract</td>
<td></td>
<td>JCT</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>---</td>
<td>Reserved</td>
<td></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>22</td>
<td>$SEANNEWS</td>
<td>Update of JESNEWS</td>
<td></td>
<td>SJB</td>
<td>No</td>
</tr>
<tr>
<td>23</td>
<td>$SEANWBL</td>
<td>Build JESNEWS token</td>
<td></td>
<td>IOT</td>
<td>No</td>
</tr>
<tr>
<td>24</td>
<td>$SEAVERS</td>
<td>Subtask to create access control environment element (ACEE) for general subtasks</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>25</td>
<td>$SEAAUD</td>
<td>Audit for job in error</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>26</td>
<td>$SEADCHK</td>
<td>Authorization for $DESTCHK</td>
<td></td>
<td>DCW</td>
<td>No</td>
</tr>
<tr>
<td>27</td>
<td>$SEATSOC</td>
<td>SYSOUT data set create for trace</td>
<td></td>
<td>IOT</td>
<td>No</td>
</tr>
<tr>
<td>28</td>
<td>$SEASSOC</td>
<td>SYSOUT data set create for system job data sets (for example, JOBLOG)</td>
<td></td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>$SEANSOC</td>
<td>SYSOUT data set create for JESNEWS</td>
<td></td>
<td>IOT</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>$SEASOX</td>
<td>Transmit or offload of SYSOUT</td>
<td></td>
<td>PCE</td>
<td>Yes</td>
</tr>
<tr>
<td>31</td>
<td>$SEANJEV</td>
<td>VERIFYX for receive or reload of SYSOUT</td>
<td></td>
<td>SFI</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>$SEAJOX</td>
<td>Transmit or offload of job</td>
<td></td>
<td>PCE</td>
<td>Yes</td>
</tr>
<tr>
<td>33</td>
<td>---</td>
<td>Reserved</td>
<td></td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>34</td>
<td>$SEASPBO</td>
<td>Spool browse data set open</td>
<td></td>
<td>SDB</td>
<td>Yes</td>
</tr>
<tr>
<td>35</td>
<td>$SEASEFS</td>
<td>Scheduler service, TOKNXTR</td>
<td></td>
<td>SSW</td>
<td>No</td>
</tr>
<tr>
<td>36</td>
<td>$SEASSWM</td>
<td>SWM modify ALTER AUTH</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>37</td>
<td>$SEASAPI</td>
<td>SYSOUT application programming interface</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>38-39</td>
<td>---</td>
<td>Not currently in use</td>
<td></td>
<td>Not in use</td>
<td>Not in use</td>
</tr>
<tr>
<td>40</td>
<td>$SEANSON</td>
<td>Secure NJE signon SAF profiles for secure NJE signon</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>41</td>
<td>$SEADIRA</td>
<td>Seclabel dominance</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>42</td>
<td>$SEASPLR</td>
<td>SPOOL I/O AUTH check</td>
<td></td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>43-255</td>
<td>---</td>
<td>Not currently in use</td>
<td></td>
<td>Not in use</td>
<td>Not in use</td>
</tr>
</tbody>
</table>

Notes:
1. * Your exit routine should always check for the presence of the control block before using fields in the control block. Currently, the control block is not present when the $SEAXTRT function occurs during an open of TSU or STC internal readers.
2. ** Job exit mask suppression not in effect during selected processing.
Register contents when Exit 37 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX37A in SYS1.SHASSAMP contains a sample of Exit 37.
Exit 38: TSO/E receive data set disposition

Function

During processing of a TSO/E RECEIVE command, SAF determines a user’s authority to receive a data set based on the SECLABELs listed in the user’s profile. Default actions JES2 takes when SAF returns control are:

- If the user can receive the data set with the current SECLABEL (the SECLABEL the user logged on with), RECEIVE processing continues normally and JES2 selects the data set.
- If the user cannot receive the data set with the current SECLABEL, but the user profile contains a SECLABEL that will allow the user to receive the data set, JES2 does not select the data set at this time. Use exit 37 to override this processing.
- If the user cannot receive the data set with the current SECLABEL or any of the SECLABELs in the user profile, JES2 deletes the data set. Use this exit to change this processing.

In this exit you set a response byte to have JES2:

- Continue normal processing, which deletes the data set.
- Bypass the data set. Bypassing the data set causes the data set to remain on spool. This could cause an undesirable accumulation of data on spool.

You can also supply extra information to the user about the final disposition of the data set. For more information about SECLABELs, see z/OS Security Server RACF Security Administrator’s Guide.

Environment

Task

JES2 address space. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 38 in supervisor state and PSW key 1.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery. If an abend does occur within the exit routine, JES2 assumes a response byte than indicates normal processing (delete the data set) should occur.

Job exit mask

This exit point is not subject to job exit mask suppression.
Exit 38

Mapping macros normally required

$HASPEQU, $HCT, $PSO, $XPL

Point of processing

This exit is taken from HASPPSO. JES2 passes control to this exit after obtaining a response from SAF for authorization to a data set during TSO/E RECEIVE processing.

Programming considerations

None.

Register contents when Exit 38 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>X038RESP</td>
</tr>
<tr>
<td></td>
<td>X038PSO</td>
</tr>
<tr>
<td></td>
<td>X038IND</td>
</tr>
<tr>
<td></td>
<td>X038COND</td>
</tr>
<tr>
<td></td>
<td>X038JOA</td>
</tr>
</tbody>
</table>

Note: If the exit must update JOE fields, it should obtain and return an update mode JOA. For more information, see the “Checkpoint control blocks for JOEs” on page 384.

| 2-10     | N/A      |
| 11       | Address of the HCT |
| 12       | N/A      |
| 13       | N/A      |
| 14       | Return address |
| 15       | Entry address |

Register contents when Exit 38 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
</tr>
<tr>
<td></td>
<td>X038IND</td>
</tr>
</tbody>
</table>
### Exit 38

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X038COND</td>
<td>Condition byte</td>
</tr>
<tr>
<td>X038RESP</td>
<td>Response byte. Set by the exit before returning to JES2:</td>
</tr>
<tr>
<td>X038KEEP</td>
<td>If you set this bit on, JES2 will bypass data set selection and will keep</td>
</tr>
<tr>
<td></td>
<td>the JOE. Otherwise, normal processing will continue and the data set will</td>
</tr>
<tr>
<td></td>
<td>be deleted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return Code</td>
</tr>
</tbody>
</table>

A return code of:

- **0**: Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

- **4**: Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

---

**Coded example**

Module HASX38A in SYS1.SHASSAMP contains a sample of exit 38.
Exit 39: NJE SYSOUT reception data set disposition

Function

This exit allows an installation to change the default processing (delete) for a data set that failed RACF verification upon entering this node for SNA and BSC NJE lines.

In this exit, you can:

- Continue default processing and delete the data set
- Accept the data set

Environment

Task

JES2 address space. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 39 in supervisor state and PSW key 1.

Recovery

No recovery is in effect when this exit is taken. Your exit routine must provide its own recovery.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $JCT, $JCTX, $NHD, $PDDB, $XPL

Point of processing

This exit is taken from HASPNET. JES2 passes control to this exit when RACF fails the verification for a SYSOUT data set received from another node.

Programming considerations

1. When rerouting the data set, your exit routine should ensure the data set has the proper authority for the target node.
2. If your routine accepts SYSOUT already rejected by RACF, there will not be an audit record for the subsequent data set create. The owner of the data set is the userid of the job that created the SYSOUT, even if that userid could not own the data on your system and RACF does not validate the assigned userid. If you are using security labels, RACF assigns a SECLABEL of SYSLOW to the data set created.
3. Expanding the JCT Control Block

© Copyright IBM Corp. 1988, 2009

245
Exit 39

You can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Note: If you code Exit 39, it may also be necessary for you to code a parallel Exit 55 to provide the same function for TCP/IP lines.

Register contents when Exit 39 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number</td>
</tr>
<tr>
<td>X039IND</td>
<td>Indicator byte</td>
</tr>
<tr>
<td>X039COND</td>
<td>Condition byte</td>
</tr>
<tr>
<td>X039RESP</td>
<td>Response byte.</td>
</tr>
<tr>
<td>X039PDBB</td>
<td>PDBB address</td>
</tr>
<tr>
<td>X039JCT</td>
<td>JCT address</td>
</tr>
<tr>
<td>X039NDH</td>
<td>Data set header address</td>
</tr>
<tr>
<td>X039AREA</td>
<td>SRW address</td>
</tr>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 39 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X039IND</td>
<td>Indicator byte</td>
</tr>
<tr>
<td>X039COND</td>
<td>Condition byte</td>
</tr>
<tr>
<td>X039RESP</td>
<td>Response byte. Set by exit before returning to JES2:</td>
</tr>
</tbody>
</table>

X039RECV Setting this bit on will allow JES2 to receive the data set. Otherwise, processing will continue and the data set will be deleted.

2-13 | N/A
Exit 39

14 Return address
15 Return Code

A return code of:

0 Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4 Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX39A in SYS1.SHASSAMP contains a sample of Exit 39.
Exit 40: Modifying SYSOUT characteristics

Function

Use Exit 40 to change the characteristics of a SYSOUT data set before JES2 gathers the attributes of the data set into an output group ($JOE). For example, you can change class, routing, or forms attributes of the data set. You can also affect the grouping of the PDDBs, or delete the data set by setting the PDB1NSOT bit in PDBFLAG1. Any logical attributes of the data can be changed with this exit. You can also use Exit 40 to influence the issuance of the $HASP549 notify messages.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places Exit 40 in supervisor state and PSW key 1.

Recovery

No recovery is in effect. Your exit routine should provide its own recovery.

Job exit mask

This exit is not subject to suppression.

Mapping macros normally required

$HASPEQU, $HCT, $DSCT, $JCT, $JCTX, $JQE, $PDDB, $PCE, $XPL

Point of processing

JES2 passes control to this exit just before it creates JOEs for the job. This exit can be taken:

- During spin processing, called from HASPSPIN before a JOE is created for a spin PDDB.
- During unspun processing, called from HASPSPIN before a JOE is created for a spin PDDB.
- During regular processing, called from HASPHOPE before the JOEs are created from the non-spin PDDBs.

JES2 gathers the non-spin data sets into groups after leaving this exit and the groups will reflect the changes your routine makes.

Programming considerations

- You can determine whether JES2 invoked Exit 40 for a transaction program by determining whether a $DSCT is available in field X040DSCT of the $XPL.
Exit 40

- You can **not** change the characteristics of SYSOUT data sets defined as OUTPUT=DUMMY; they are not passed to Exit 40. However, SYSOUT data sets defined as OUTDISP=PURGE are passed and available to this exit.

- **Expanding the JCT Control Block**

You can add, expand, locate, and remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro expansion service. For example, you can use these extensions to store job-related information. For more information, see [z/OS JES2 Macros](#).

Note that only the $JCTXGET macro can be used from this exit if any of the following indicator bytes (for non-spin and unspun PDDBs) have been marked on in the parameter list:
- X040NSPN
- X040UNSP

If these bytes are set on, JES2 will not write modifications of the extensions to spool.

### Contents of registers when Exit 40 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number</td>
</tr>
<tr>
<td>X040IND</td>
<td>Indicator byte.</td>
</tr>
<tr>
<td>X040SPIN</td>
<td>If this bit setting is on, it is a spin PDDB.</td>
</tr>
<tr>
<td>X040NSPN</td>
<td>If this bit setting is on, it is a non-spin PDDB.</td>
</tr>
<tr>
<td>X040UNSP</td>
<td>If this bit setting is on, it is an unspun PDDB.</td>
</tr>
<tr>
<td>X040COND</td>
<td>Condition byte</td>
</tr>
<tr>
<td>X040RESP</td>
<td>Response byte</td>
</tr>
<tr>
<td>X040PDBB</td>
<td>Address of $PDBB</td>
</tr>
<tr>
<td>X040JQE</td>
<td>Address of $JQE</td>
</tr>
<tr>
<td>X040JCT</td>
<td>Address of $JCT or 0. JES2 is unable to supply the address of a $JCT when processing spin PDDBs.</td>
</tr>
<tr>
<td>X040DSCT</td>
<td>Address of $DSCT or 0. JES2 only supplies the address of a $DSCT when processing a SYSOUT data set produced by a transaction program.</td>
</tr>
<tr>
<td>X040VTXT</td>
<td>A 20-byte EBCDIC field containing variable text to be placed in the $HASP548 message in place of &quot;INVALID USERID&quot; for NETMAIL output, if the PDB1NSOT flag is turned on by the exit.</td>
</tr>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of $HCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of $PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
</tbody>
</table>
Register contents when Exit 40 passes control back to JES2

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unchanged</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure:</td>
</tr>
</tbody>
</table>

**Field Name** | **Description** |
----------------|-----------------|
XPLID           | The eyecatcher  |
XPLLEVEL        | The version level of $XPL |
XPLXITID        | The exit ID number |
X040IND         | Indicator byte   |

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X040SPIN</td>
<td>If this bit setting is on, it is a spin PDB.</td>
</tr>
<tr>
<td>X040NSPN</td>
<td>If this bit setting is on, it is a non-spin PDB.</td>
</tr>
<tr>
<td>X040UNSP</td>
<td>If this bit setting is on, it is an unspun PDB.</td>
</tr>
</tbody>
</table>

X040COND   | Condition byte |
X040RESP   | Response byte  |

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X040RFNT</td>
<td>Enables JES2 to issue the $HASP549 notification message to the intended receiver of the transmitted file, if the PDB9ONOT flag of the PDBFLAG9 byte is set. If this return code is set, JES2 ignores the NJEDEF MAILMSG= parameter.</td>
</tr>
</tbody>
</table>

**Notes:**

1. If the exit turns on the PDB1NSOT bit in the PDBFLAG1 byte of the $PDDB, JES2 ignores this return code and suppresses the $HASP549 message.

2. If the exit routine alters the PDBUSER field of the $PDDB, JES2 routes the $HASP549 message to the user that the contents of PDBUSER indicate. So the sender’s intended receiver does not receive this notification message.

X040RNNT   | Disables JES2 to issue the $HASP549 notification message to the intended receiver of the transmitted file. If this return code is set, JES2 ignores the NJEDEF MAILMSG= parameter. |

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X040PDDB</td>
<td>Address of $PDDB</td>
</tr>
<tr>
<td>X040JQE</td>
<td>Address of $JQE</td>
</tr>
<tr>
<td>X040JCT</td>
<td>Address of $JCT, or 0. JES2 is unable to supply the address of a $JCT when processing spin PDDBs.</td>
</tr>
</tbody>
</table>
| X040DSCT   | Address of $DSCT or 0. JES2 only supplies the
Exit 40

address of a DSCT when processing a SYSOUT data set produced by a transaction program.

**X040VTXT**

A 20-byte EBCDIC field containing variable text to be placed in the $HASP548 message is in place of "INVALID USERID" for NETMAIL output, if the PDB1NSOT flag is turned on by the exit.

### 2-14

Unchanged

### 15

Return Code

A return code of:

- **0**
  
  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine.

- **4**
  
  Tells JES2 that even if additional exit routines are associated with this exit, ignore them.

### Coded example

Module HASX40A in SYS1.SHASSAMP contains a sample of Exit 40.
Exit 41: Modifying output grouping key selection

Function

Use exit 41 to affect which OUTPUT JCL keywords JES2 uses for generic grouping.

JES2 passes this exit a table that contains the SJF keys for the default generic grouping keywords. There is a one-to-one correspondence between the SJF keys and the OUTPUT JCL keywords. You can use this exit to add keys to or delete keys from this table. You can add up to 24 additional keys at the end of the table. Delete keys by compressing the table.

Generic grouping cannot perform special processing for keywords (such as handling defaults or overrides). A keyword should not be grouped generically if it has any of the following attributes:

- The keyword can be overridden by another source. CLASS, DEST, and WRITER can be overridden on the DD statement. The network SYSOUT receiver uses the group id in a data set header; the group id might have been generated by the execution node and thus not be present on the OUTPUT statement.
- The keyword can be specified at dynamic unallocation (for example, CLASS).
- The keyword has a default value that JES2 must provide. DEST, OUTDISP, and PRMODE, for example, have default values.
- The keyword can be specified in an alternate way (for example, HOLD=YES on the DD statement is equivalent to OUTDISP=HOLD).

Keywords that require special processing should be managed by the PDDB and be grouped upon by the output processor.

JES2 passes this exit the name of the JCL definition vector table (JDVT) that defines these keys. The table of OUTPUT grouping keys applies to all OUTPUT statements processed using this JDVT.

Environment

Task

User environment. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places exit 41 in supervisor state and PSW key 0 or 1.

Recovery

No recovery is in effect. Your exit routine must provide its own recovery.

Job exit mask

This exit is not subject to job exit mask suppression.
Mapping macros normally required
$HASPEQU, $HCCT, $XPL, SJTRP.

Point of processing
This exit is taken from HASCGGKY during JES2 initialization after the default OUTPUT grouping keywords have been selected, but before any grouping is done based on this JDVT name. The table of grouping keys, as modified by the exit, is used for all subsequent grouping for that JDVT name.

Programming considerations
None

Register contents when Exit 41 gets control
The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Zero</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>X041IND</td>
</tr>
<tr>
<td></td>
<td>X041COND</td>
</tr>
<tr>
<td></td>
<td>X041RESP</td>
</tr>
<tr>
<td></td>
<td>X041GGKT</td>
</tr>
<tr>
<td></td>
<td>X041DEFN</td>
</tr>
<tr>
<td></td>
<td>X041TOTN</td>
</tr>
<tr>
<td></td>
<td>X041RSVN</td>
</tr>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HCCT</td>
</tr>
<tr>
<td>12</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Address of an available save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry address</td>
</tr>
</tbody>
</table>

Register contents when Exit 41 passes control back to JES2
Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return Address</td>
</tr>
</tbody>
</table>
15

Return Code

A return code of:

0

Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. Otherwise, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4

Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX41A in SYS1.SHASSAMP contains a sample of exit 41.
Exit 42: Modifying a notify user message

Function

This exit allows you to affect how a notify user message will be handled. When a notify user message is to be issued, the notify user message SSI service routine is invoked. The routine validates the input and then invokes this installation exit, before the notify user message is built and issued. Use Exit 42 to:

- Cancel the message.
- Change the destination of the message. You can change the userid, node, or both to which the message is to be routed.
- Change the message text.
- Continue processing without changing the message or destination.

Environment

Task

User address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 42 in supervisor state and PSW key 0 or 1.

Recovery

$ESTAE recovery is in effect, under the $ESTAE established when the SSI was invoked. However, your exit routine should provide its own recovery, as with every exit.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HCCT, $XPL, SSNU, SSOB

Point of processing

JES2 takes this exit after the input for a message has been validated and authorization checking has been done for the receiving userid and node. If the exit routine changes the destination, it must provide its own authority and validity checks. Exit 42 will return to the SSI service for the message processing to be completed.

Programming considerations

1. Before this exit is invoked, the system does validity and authorization checking of the node and userid that is to receive the message. Therefore, if the exit changes the node or userid to which the message will be sent, the installation must check the validity and the authority of the new destination.
2. If errors were detected by the SSI service, the bit setting X042CANC will be on in the response byte, indicating that the notify message is to be canceled. If your exit routine corrects the error and turns X042CANC off, to issue the message, it should also zero out the exit-supplied reason and return codes in fields X042REAS and X042RC of the parameter list.

3. As the notify user SSI caller can be unauthorized, you must take special consideration if the SSNU extension is directly referenced in the exit routine. The SSI caller’s key is provided so that the exit can reference SSNU data appropriately. Additionally, the $XPL contains fields so that the exit can update the userid, message text, and message length.

Note: IBM suggests updating information in the XPL instead of the SSNU. When using the XPL fields, JES2 ensures the changes are appropriately handled. However, when changing the SSNU directly, the exit must understand how JES2 uses the SSNU fields in subsequent logic.

Register contents when Exit 42 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>The eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>The version level number of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number</td>
</tr>
<tr>
<td>X042IND</td>
<td>Indicator byte</td>
</tr>
<tr>
<td>X042COND</td>
<td>Condition byte. This byte might contain the following bit settings on entry, if an error exists:</td>
</tr>
<tr>
<td></td>
<td>X042EMSG</td>
</tr>
<tr>
<td></td>
<td>X042NOXT</td>
</tr>
<tr>
<td></td>
<td>X042EXTE</td>
</tr>
<tr>
<td></td>
<td>X042NOAU</td>
</tr>
<tr>
<td></td>
<td>X042UERR</td>
</tr>
<tr>
<td></td>
<td>X042DERR</td>
</tr>
<tr>
<td>X042RESP</td>
<td>Response byte.</td>
</tr>
<tr>
<td>X042SNUA</td>
<td>Address of the SSNU extension for the SSOB</td>
</tr>
<tr>
<td>X042NEWN</td>
<td>Current node identifier, in binary form</td>
</tr>
<tr>
<td>X042NEWR</td>
<td>Current remote identifier, in binary form</td>
</tr>
<tr>
<td>X042NWML</td>
<td>Current message length</td>
</tr>
<tr>
<td>X042REAS</td>
<td>Exit-supplied reason code</td>
</tr>
<tr>
<td>X042RC</td>
<td>Exit-supplied return code</td>
</tr>
<tr>
<td>X042NEWU</td>
<td>Current userid</td>
</tr>
<tr>
<td>X042NEWM</td>
<td>Pointer to current message</td>
</tr>
<tr>
<td>X042CKEY</td>
<td>SSI caller’s key</td>
</tr>
</tbody>
</table>
X042MEMB  The member number that the message should be routed to if the userid is not logged on and OUTDEF BRODCAST=NO.

2-10   N/A
11   Address of the HCCT
12   N/A
13   N/A
14   Return address
15   Entry address

Register contents when Exit 42 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list mapped by $XPL:</td>
</tr>
</tbody>
</table>

Field Name   Description

XPLRESP  This response byte must be set by the exit before returning to JES2. Set the response byte as follows:

X042CANC  This bit setting turned on in the response byte indicates that the notify message is to be canceled. Otherwise, the notify message is to be issued. This bit will be turned off on entry if no errors exist before the installation exit gets control, but will be turned on entry if errors are found before the installation exit gets control. If the exit corrects the errors detected, this bit setting should be reset to be off.

X042SETR  This bit setting turned on in the response byte indicates that both a return code and a reason code were specified in the parameter list. If this bit setting is not on, neither reason code nor return code are present.

X042NOCH  This bit setting turned on in the response byte indicates that the node has been changed. If this bit setting is not turned on, there has been no change to the destination node.

X042RMCH  This bit setting turned on in the response byte indicates that the remote has been changed. If this bit setting is not turned on, there
has been no change to the destination remote.

**X042USCH**  This bit setting turned on in the response byte indicates that the userid has been changed. If this bit setting is not turned on, there has been no change to the userid.

**X042MSGC**  This bit setting turned on in the response byte indicates that the message text and length have been changed. If this bit setting is not turned on, there has been no change to the message text and length.

**X042MEMC**  This bit setting turned on in the response byte indicates that the member number in X042MEMB was changed by the exit.

**X042MAIN**  This bit setting turned on in the response byte indicates that the notify request should be unconditionally queued to the JES2 main task for processing. This allows the message to be seen by $EXIT 10.

**X042NEWN**  New node identifier, in binary form, to be returned from exit, if there was a change in the node.

**X042NEWR**  New remote identifier, in binary form, to be returned from exit, if there was a change in the remote.

**X042NEWU**  New userid to be returned from the exit, if there was a change in the userid.

**X042NEWM**  New message text pointer. Note that if the text is updated, the message length in X042NWML must be updated.

**X042NWML**  New message text length to be returned, if there was a change to the message text.

**X042REAS**  Exit-supplied reason code

**X042RC**  Exit-supplied return code

<table>
<thead>
<tr>
<th>2-14</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Return Code</td>
</tr>
</tbody>
</table>

A return code:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them.

---

**Coded example**

Module HASX42A in SYS1.SHASSAMP contains a sample of exit 42.
Exit 43: APPC/MVS TP selection/change/termination

Function

When the system processes an APPC/MVS transaction program (TP) or a z/OS UNIX® application, this exit allows you to receive control during:

- TP selection processing, which means the TP initiator selected a TP to run.
- TP termination processing, which means the TP initiator completed processing a TP.
- TP change processing, which means the TP initiator was processing a multi-transaction TP. The APPC/MVS transaction initiator or z/OS UNIX BPXAS initiator started another TP as a result of completing another TP.

While JES2 is processing a TP selection request, you could implement Exit 43 to:

- Create installation-specific control blocks to be used by subsequent installation exits that are invoked for the TP after Exit 43.
- Modify the output limits maintained in the $SJB.
- Issue messages to the TP's message log.

While processing a multi-transaction TP, if JES2 is invoked for a change request, you could implement Exit 43 to:

- Reset the output limit counts associated with the TP’s SYSOUT data set
- Issue messages to the TP’s message log.

During TP termination processing, you could implement Exit 43 to:

- Release any control blocks Exit 43 previously obtained for the TP.
- Issue messages to the TP’s message log.

Related exits

IBM suggests that you use exit IEFUJI to terminate a TP instead of Exit 43. See [z/OS MVS Installation Exits](#) for additional information about exit IEFUJI.

If a SYSOUT data set created by a TP exceeded the output limits specified in Exit 43 or in the initialization stream, JES2 invokes Exit 9.

Recommendations for implementing Exit 43

It might be necessary for you to create control blocks that your installation will use while APPC/MVS is processing the transaction program. To create installation-specific control blocks:

1. Create a DSECT for your installation’s control block
2. In Exit 43:
   a. Include all the control blocks necessary for the exit. Mapping macros normally required in the Environment section identifies all the control blocks IBM suggests should be included. Be sure to include any installation-specific control blocks you have created for TPs.
   b. Issue a $GETMAIN macro to obtain storage for the control block.
   c. Initialize the control block with the required information.
   d. Use the information as required while JES2 processes the transaction program.
Exit 43

Your installation might want to issue installation-defined messages to the TP message log when either JES2 selects or terminates a transaction program. Code the following macro to issue a message in Exit 43:

\$WTO \ ROUTE=\$LOG

Environment

Task

User (APPC/MVS transaction initiator). You must specify ENVIRON=USER on the \$MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 43 in supervisor state and PSW key 0

Locks held before entry

\$SJB

Restrictions

- Exit 43 should not perform any I/O. If I/O is performed in Exit 43, your installation might experience a degradation in its performance.

Recovery

\$ESTAE is in effect and provides minimal recovery. JES2 will attempt to recover from any errors experienced by Exit 43. However, you should not depend on JES2 for recovery.

Job exit mask

Exit 43 is subject to suppression. You can suppress exit 43 by either implementing Exit 2 to set the 43rd bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream. All TPs submitted under the APPC/MVS transaction initiator will not invoke Exit 43.

Storage recommendations

Subpool 230

Mapping macros normally required

\$HASPEQU, \$SJB, \$JCT, \$JCTX, \$XPL

Point of processing

JES2 invokes Exit 43 during TP selection, change, or termination processing.

Programming considerations

You should consider the following when implementing installation exit 43:
• Any code implemented in this installation exit will be invoked for every transaction program submitted under this initiator.
• The output limits are found in the $SJB and the $SJXB.

• **Expanding the JCT Control Block**
  You can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see [z/OS JES2 Macros](#).

### Register contents when Exit 43 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address to a parameter list with the following structure:</td>
</tr>
</tbody>
</table>

#### Field Name

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Eyecatcher - $XPL</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level of $XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit identifier number - 43</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version level of the exit</td>
</tr>
<tr>
<td>X043IND</td>
<td>Indicator byte</td>
</tr>
</tbody>
</table>

| X043TPS    | Indicates Exit 43 was invoked for TP select processing. |
| X043TPT    | Indicates Exit 43 was invoked for TP terminate processing. |
| X043CHG    | Indicates Exit 43 was invoked for TP change processing. |
| X043COND   | Not applicable to Exit 43 |
| X043RESP   | Not applicable to Exit 43 |
| X043SJB    | Pointer to the $SJB |
| X043JCT    | Pointer to the $JCT |
| X043SIZE   | Length of $XPL for Exit 43 |
| 2-10       | Not applicable |
| 11         | Address of the $HCCT |
| 12         | Not applicable |
| 13         | Address of a save area |
| 14         | Return address |
| 15         | Entry address |

### Register contents when Exit 43 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-14</td>
<td>Unchanged from entry registers</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:
Exit 43

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the TP.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates JES2 should continue processing the TP but ignore any additional exits associated with the TP.</td>
</tr>
</tbody>
</table>

Coded example

Module HASX43A in SYS1.SHASSAMP contains a sample of Exit 43.
Exit 44: JES2 converter exit (JES2 main)

Function

This exit allows you to modify job-related control blocks after the converter running as a subtask in the JES2 address space has converted the job’s JCL into C/I text. After the system has converted the job’s JCL, your installation might want to:

- Change fields in the job’s job queue element ($JQE), such as:
  - Change the priority of the job
  - Release the job from hold
  - Route the job to print on a device other that what was specified on the job’s JCL
  - Reassign the system where the job should execute or print
- Perform spool I/O for installation-defined control blocks. You can supply a scheduling environment to the JQASCH field in the JQE. This will override any scheduling environment from the JOBCLASS(n) for this job. JES2 does not validate the scheduling environment; therefore, be careful to supply a valid scheduling environment or the system will not schedule the job for execution. If needed, use Exit 6 to provide scheduling environment validation.
- Exit 44 can be used to reject duplicate TSO logons.

Related exits

Exit 6 is invoked while the converter subtask is processing the job. Exit 6 is called earlier than Exit 44 during converter processing. Any changes your installation needs to make to the job control table ($JCT) can also be done in exit 6.

Recommendations for implementing Exit 44

If your installation implemented Exit 6 to extract information from the job’s JCL and created installation-specific control blocks, you can implement Exit 44 to write those installation-specific control blocks to spool by:

1. Issuing a $GETBUF macro to obtain a buffer. The information contained in the installation-specific control block should be moved into the buffer.
2. Issuing a $CBIO macro to write the buffer to spool.
3. Updating a user field in the $JCT with the address of the spool installation-specific control block.
4. If you intend to update the JQE passed in your exit, $DOGJQE should be used to obtain an update mode JQE and to return it when the updates are complete. You do not need to write the $JCT to spool since JES2 will write the $JCT to spool after returning from Exit 44.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31
Exit 44

Supervisor/problem program
JES2 places Exit 44 in supervisor state and PSW key 1

Recovery
$ESTAE is in effect and HASPCNVVT provides minimal recovery. JES2 attempts to recover from any abends experienced by the converter main task. However, you should not depend on JES2 for recovery.

Job exit mask
Exit 44 is subject to suppression. You can suppress Exit 44 by either implementing exit 2 to set the 44th bit in the job exit suppression mask (JCTXMASK) or by disabling the exit through the JES2 initialization stream.

Mapping macros normally required
$HASPEQU, $JQE, $JCT, $JCTX, $XPL

Point of processing
Exit 44 is invoked from the JES2 main task after the converter subtask has converted the job’s JCL. It is invoked before JES2 writes job-related control blocks to spool.

After Exit 44 returns to JES2, JES2 examines the response byte in the $XPL. If an error was encountered and Exit 44 set the response byte in Exit 44 to indicate the job should be placed on the:
• Purge queue or output queue, JES2 places the job on the specified queue.
• Purge queue and output queue, JES2 places the job on the purge queue.

If Exit 44 did not set the response byte, JES2 places the job on the execution queue.

Programming considerations
The following are programming considerations for Exit 44:
1. If Exit 44 sets an indicator in the response byte (XPLRESP) before returning to JES2, JES2 honors the setting over any specifications made in the job’s JCL.
2. Locating the JCT Control Block Extensions
   You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTGET macro. For more information, see $/OS JES2 Macros.
3. If you need to change the scheduling environment, use the JCTSCHEN field in the JCT.

Register contents when Exit 44 gets control
The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 44</td>
</tr>
<tr>
<td>1</td>
<td>Address of a parameter list with the following structure:</td>
</tr>
</tbody>
</table>

Field Name | Description
--- | ---
XPLID | Eyecatcher - $XPL
XPLLEVEL | Version level of $XPL
Exit 44: JES2 converter exit (JES2 main)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLXITID</td>
<td>Exit identifier number - 44</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version level of the exit</td>
</tr>
<tr>
<td>X044IND</td>
<td>Indicates the type of error, if any, while converting the job's JCL</td>
</tr>
<tr>
<td></td>
<td>- <strong>X044JCLO</strong> indicates the converter successfully converted the job's JCL</td>
</tr>
<tr>
<td></td>
<td>- <strong>X044JCLE</strong> indicates the converter encountered an error while converting</td>
</tr>
<tr>
<td></td>
<td>the job's JCL</td>
</tr>
<tr>
<td></td>
<td>- <strong>X044CPER</strong> indicates a system error occurred</td>
</tr>
<tr>
<td></td>
<td>while the converter was converting the job's JCL.</td>
</tr>
<tr>
<td></td>
<td>See X044COND for additional information.</td>
</tr>
<tr>
<td>X044COND</td>
<td>Indicates additional information about the type of error that was</td>
</tr>
<tr>
<td></td>
<td>encountered.</td>
</tr>
<tr>
<td></td>
<td>- <strong>X044DLGN</strong> a user is already logged onto the system with the same</td>
</tr>
<tr>
<td></td>
<td>TSU user id.</td>
</tr>
<tr>
<td></td>
<td>- <strong>X044FKOF</strong> JES2 was unable to open the system data sets for the</td>
</tr>
<tr>
<td></td>
<td>converter.</td>
</tr>
<tr>
<td></td>
<td>- <strong>X044CNWT</strong> JES2 could not convert the job</td>
</tr>
<tr>
<td></td>
<td>because the job's JCLIB data set was not available.</td>
</tr>
<tr>
<td>X044RESP</td>
<td>Response byte</td>
</tr>
<tr>
<td>X044CNVQ</td>
<td>JES2 requeues the job to conversion</td>
</tr>
<tr>
<td>X044JCT</td>
<td>Address of the $JCT</td>
</tr>
<tr>
<td>X044JQE</td>
<td>Address of the $JQE</td>
</tr>
<tr>
<td>X044SIZE</td>
<td>Length of $XPL for Exit 44</td>
</tr>
</tbody>
</table>

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of a parameter list with the following structure:</td>
</tr>
</tbody>
</table>

**Field Name**

- **X044IND** Indicator byte
- **X044COND** Condition byte
- **X044RESP** Response byte
  - **X044OUTQ** Indicates JES2 should place the job on the output queue
  - **X044PURQ** Indicates JES2 should place the job on the purge queue
Exit 44

<table>
<thead>
<tr>
<th>Exit Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the job.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates JES2 should continue processing the job but ignore any additional exits associated with the job.</td>
</tr>
</tbody>
</table>

**Coded example**

Module HASX44A in SYS1.SHASSAMP contains two samples of Exit 44.
Exit 45: Pre-SJF service request

Function

This exit allows you to process requests for the scheduler JCL facility prior to JES2’s processing of the request. A function code of 70 on a subsystem IEFSSREQ call with SSSFSWBM in field SSSFREQF invokes the exit. Exit 45 allows the installation to:

- Examine the request to determine if the system should continue to process the request for SJF services
- Redirect error messages for a request.

Environment

Task

User task. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places exit 45 in supervisor state and PSW key 1

Recovery

A $ESTAE recovery is in effect for exit 45. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide minimal recovery. You should provide recovery for errors that might be encountered by exit 45’s processing.

Job exit mask

Exit 45 is subject to suppression. The installation can suppress the exit either by implementing exit 2 to set bit 45 in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Storage recommendations

Subpool 241 or 231

Mapping macros normally required

$HASPEQU, $HCT, $XPL, $SFRB, IAZSSSF

Point of processing

Exit 45 is invoked by a subsystem issuing an IEFSSREQ macro with a function code of 70 and SSSFSWBM in field SSSFREQF. This is a request for scheduler JCL facility (SJF) SWB modify services. The request is routed through the subsystem interface and JES2, module HASCSJFS, receives control. HASCSJFS performs the following functions:

1. Establish a recovery environment.
2. Validate the SSFOB and its extension SSSF.
3. Issue a $SEAS request to obtain UTOKEN of the requester.
Programming considerations

Because the SJF Services SSI caller (SSI 70) can be unauthorized, the SSSF extension can be located in an unauthorized storage key. Therefore, you must take special consideration if the SSSF extension is directly referenced in the exit routine. The SSI caller's key is provided so that the exit can reference SSSF data appropriately. However, many fields in the SSSF extension are located in the $XPL, so no key considerations are necessary when using these fields. IBM suggests that the exit reference fields in the $XPL rather than the corresponding fields in the SSSF.

Register contents when Exit 45 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to exit 45</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list, which has the following structure:</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>X045VERN</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>XPLEXLEV</td>
</tr>
<tr>
<td></td>
<td>X045SIZE</td>
</tr>
<tr>
<td></td>
<td>X045IND</td>
</tr>
<tr>
<td>X045COND</td>
<td>If set, indicates the reason why JES2 is unable to process the SJF request. If XPLCOND is set to:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: If XPLCOND is set, JES2 has preset XPLRESP to X045CANC to cancel the request for SJF services.
Exit 45

X045RESP  Response byte
X045SSSA  Contains the address of IAZSSSF.
X045SFRB  Contains the address of the JES2 scheduler facilities request block (SFRB) to be given to the JES2 SJF PCE.
X045CKEY  Contains the SSI caller's key
X045FLG1  Indicates the intended type of security authorization checking to be done in order to ensure that the user has access to the target sysout dataset. A value of:

X045DEST  Indicates that a DEST (ISFAUTH) security check will be done.
X045SECL  Indicates that a SECLABEL dominance security check will be done.
X045JSSP  Indicates that a security check against the JESSPOOL resource class will be done.
X045JBNM  Contains the job name of the target sysout dataset.
X045JBID  Contains the job ID of the target sysout dataset.
X045GRPN  Contains the group name of the target sysout dataset.
X045GRP1  Contains the first group identifier of the target sysout dataset.
X045GRP2  Contains the second group identifier of the target sysout dataset.
X045CART  Contains the command and response token for WTO responses.
X045CNID  Contains the console ID for WTO responses.
X045MDAD  Contains the address of the output descriptor modify list in SWBTU format.
X045ERAD  Contains the address of the output descriptor erase list in TU format.
X045MDLN  Contains the length of the modify list (SWBTU).
X045ERLN  Contains the length of the erase list (TU).

2-10  Not applicable to exit 45
11  Address of the $HCCT
12-13  Not applicable to exit 45
14  Return address
15  Entry point address of exit 45
Exit 45

Register contents when Exit 45 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to exit 45</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list which has the following structure:</td>
</tr>
<tr>
<td></td>
<td>X045IND</td>
</tr>
<tr>
<td></td>
<td>X045COND</td>
</tr>
<tr>
<td></td>
<td>X045RESP</td>
</tr>
<tr>
<td></td>
<td>• X045CANC</td>
</tr>
<tr>
<td></td>
<td>• X045SETR</td>
</tr>
<tr>
<td></td>
<td>X045REAS</td>
</tr>
<tr>
<td></td>
<td>X045RC</td>
</tr>
<tr>
<td></td>
<td>X045FLG1</td>
</tr>
<tr>
<td></td>
<td>• X045DEST</td>
</tr>
<tr>
<td></td>
<td>• X045SECL</td>
</tr>
<tr>
<td></td>
<td>• X045JSSP</td>
</tr>
<tr>
<td>2-13</td>
<td>Not applicable to exit 45</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Exit effector return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the job.</td>
</tr>
</tbody>
</table>
Exit 45

4 Indicates JES2 should continue processing the job, but ignore any additional exits associated with the job.

Coded example

Module HASX45A in SYS1.SHASSAMP contains a sample of exit 45.
Exit 46: Modifying an NJE data area before its transmission

Function

This exit allows you to change an NJE data area before transmitting a job to another node through SNA or BSC NJE, or while offloading jobs to spool. (See Network Job Entry (NJE) Formats and Protocols for more information about the various NJE data areas that can be transmitted across a network.) Before transmitting the NJE job, your installation might need to add, remove or change information to one or more of the following NJE data areas:

- NJE job header
- NJE data set header
- NJE RCCS (Record Characteristics Change Section) header
- NJE job trailer

Your installation might want to:

- Remove any installation-defined sections your installation added to the NJE job when exit 47 was processing the NJE job. However, it might not be necessary to remove any installation-defined sections because installation-defined sections are ignored when they are received at other nodes.
- Add or change information, such as accounting, security or scheduling information, needed by another node in the network.
- Extract information from user fields in JES2 defined control blocks or installation defined control blocks and transfer them to the NJE data areas.
- Remove, modify, or add an RCCS header before sending the job stream into the network.

Related exits

Consider using:

- Exit 40 if you want to change the output characteristics associated with a SYSOUT data set before it prints at your node.
- Exit 2 or exit 47 to modify NJE job headers for jobs that are received for processing at your installation.
- Exit 46 to receive control for spool offload, and BSC and SNA NJE lines.
- Exit 56 to receive control for TCP/IP lines.

Recommendations for implementing Exit 46

If you want to remove an installation-defined section from the NJE data area passed to Exit 46, you should:

1. Use XPLIND to determine the type of NJE data area that JES2 passed to Exit 46 for processing.
2. Issue a $NDHREM macro to remove the installation-defined section from the NJE data area.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.
Exit 46

AMODE/RMODE requirements
RMODE ANY, AMODE 31

Supervisor/problem program
JES2 places Exit 46 in supervisor state and PSW key 0.

Recovery
Because different types of recovery are provided by the networking or spool offload PCE, your installation should provide its own recovery routine.

Job exit mask
Exit 46 is subject to suppression. Your installation can either implement exit 2 to set the 46th bit in the job exit suppression mask (JCTXMASK) or disable the exit in the JES2 initialization stream.

Mapping macros normally required
$HASPEQU, $PDDB, $SCR, $XPL, $HCT, $NHD, $HCCT, $DCT, $JQE, $JCT, $JCTX, $JOE, $PCE, $NJEWORK, $JTW, $STW

Point of processing
JES2 invokes Exit 46 before transmitting a job while performing spool offload processing or while transmitting an SNA or BSC NJE job across the network. Before invoking Exit 46, JES2:
1. Builds the NJE data area in a 32K buffer
2. Removes any JES2-specific sections from the NJE data area if JES2 is transmitting the NJE data area to another node in the network. The following NJE data areas contain a JES2 section:
   • Job Header
   • Job Trailer
   For spool offload processing, the transmission routine does not alter the NJE data area.
3. Initializes the $XPL parameter and invokes Exit 46.

After returning from Exit 46, JES2 examines the response byte (XPLRESP) in the $XPL parameter list. If in Exit 46 you set XPLRESP to:
• X046TERM, it indicates an error occurred, JES2 terminates the transmission of the NJE data area, and places the job in hold.
• X046BYP, JES2 continues processing the remainder of the NJE job because Exit 46 transmitted the buffer that contained the NJE data area.

If XPLRESP has not been set, JES2 transmits the NJE data area.

Programming considerations
The following are programming considerations for Exit 46:
• If your installation needs to process NJE data areas differently for spool offload processing and NJE processing, use field DCTDEVTP in the $DCT to determine the type of job JES2 is processing.
• Locating the JCT Control Block Extensions
  You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For example, you can use these extensions to
retrieve job-related information from the $JCTX control block to ship across the network in $NHD macro sections. For more information, see [z/OS JES2 Macros](#).

## Register contents when Exit 46 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 46</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list, which has the following structure:</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>X046VERN</td>
</tr>
<tr>
<td></td>
<td>XPLXTID</td>
</tr>
<tr>
<td></td>
<td>XPLEXLEV</td>
</tr>
<tr>
<td></td>
<td>X046IND</td>
</tr>
<tr>
<td></td>
<td>X046HDR</td>
</tr>
<tr>
<td></td>
<td>X046TRL</td>
</tr>
<tr>
<td></td>
<td>X046DSH</td>
</tr>
<tr>
<td></td>
<td>X046RCCS</td>
</tr>
<tr>
<td></td>
<td>X046COND</td>
</tr>
<tr>
<td></td>
<td>X046R1ST</td>
</tr>
<tr>
<td></td>
<td>X046RESP</td>
</tr>
<tr>
<td></td>
<td>On input, the response bit X046BYP may be set to indicate that default JES2 processing would suppress the sending of the header. This is the case when a SYSIN data set is being sent and JES2 decided not to send an RCCS header.</td>
</tr>
<tr>
<td></td>
<td>X046HADR</td>
</tr>
<tr>
<td></td>
<td>X046DCT</td>
</tr>
<tr>
<td></td>
<td>X046JQE</td>
</tr>
<tr>
<td></td>
<td>X046JCT</td>
</tr>
<tr>
<td></td>
<td>X046PDDB</td>
</tr>
<tr>
<td></td>
<td>X046JOA</td>
</tr>
</tbody>
</table>
Exit 46

- **Note:** If the exit must update JOE fields, it should obtain and return an update mode JOA. For more information, see “Checkpoint control blocks for JOEs” on page 384.

**X046AREA** Contains the address of the NJEWORK area (JTW or STW) for the transmitter device sending the header.

**X046SIZE** Indicates the length of the $XPL parameter list for Exit 46.

| 2-10 | Not applicable to Exit 46 |
| 11   | Address of the $HCT       |
| 12   | Not applicable to Exit 46 |
| 13   | Address of the spool offload or networking $PCE |
| 14   | Return address            |
| 15   | Entry point address of Exit 46 |

**Register contents when Exit 46 passes control back to JES2**

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 46</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list, which has the following structure:</td>
</tr>
<tr>
<td></td>
<td><strong>XPLID</strong> Eye-catcher for the $XPL - $XPL</td>
</tr>
<tr>
<td></td>
<td><strong>X046VERN</strong> Indicates the version number of Exit 46</td>
</tr>
<tr>
<td></td>
<td><strong>XPLXITID</strong> Exit identifier - 46</td>
</tr>
<tr>
<td></td>
<td><strong>XPLEXLEV</strong> Version level of the exit</td>
</tr>
<tr>
<td></td>
<td><strong>X046IND</strong> Indicator byte</td>
</tr>
<tr>
<td></td>
<td><strong>X046COND</strong> Condition byte</td>
</tr>
<tr>
<td></td>
<td><strong>X046RESP</strong> Indicates the processing Exit 46 determined JES2 should perform after processing the NJE data area. A value of:</td>
</tr>
<tr>
<td></td>
<td>• <strong>X046TERM</strong> indicates Exit 46 determined the NJE data area should not be transmitted. JES2 will discard the remainder of the NJE job.</td>
</tr>
<tr>
<td></td>
<td>• <strong>X046BYP</strong> indicates JES2 should not transmit the NJE data area. JES2 will continue to process the remainder of the NJE job.</td>
</tr>
<tr>
<td></td>
<td><strong>X046SIZE</strong> Indicates the length of the $XPL parameter list for Exit 46.</td>
</tr>
<tr>
<td>2-13</td>
<td>Not applicable to Exit 46</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Exit effector return code</td>
</tr>
</tbody>
</table>

A return code of:
Exit 46

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the job.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates JES2 should continue processing the job, but ignore any additional exits associated with Exit 46.</td>
</tr>
</tbody>
</table>

**Coded example**

Module HASX46A in SYS1.SHASSAMP contains a sample of Exit 46. Module HASXJECL in SYS1.SHASSAMP also contains an example.
Exit 47: Modifying an NJE data area before receiving the rest of the NJE job

Function

This exit allows you to:

- Examine and change an NJE data area before receiving the rest of the NJE job from another node through SNA or BSC NJE or before receiving jobs from spool.
- Add, expand, locate, or remove an extension to the $JCT control block where accounting information can be stored.

Before receiving an NJE job, your installation might need to add, remove or change information to one or more of the NJE data areas below. (See Network Job Entry [NJE] Formats and Protocols for more information about the various NJE data areas that can be transmitted across a network.)

- NJE job header
- NJE data set header
- NJE RCCS (Record Characteristics Change Section) header
- NJE job trailer

Your installation might want to:

- Remove any installation-defined sections your installation added to the NJE job when exit 46 was processing the NJE job.
- Add or change information, such as accounting or security information, needed by another node in the network.
- Extract information from the NJE data areas and transfer them to user fields in JES2 defined control blocks or installation defined control blocks.

Related exits

If you want to change the output characteristics associated with a SYSOUT data set, consider using exit 40. Exit 47 can be used to receive control for spool offload and SNA and BSC NJE. If you code exit 47, you also need Exit 57 to handle jobs received on TCP/IP lines.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 47 in supervisor state and PSW key 1.

Recovery

Because different types of recovery are provided by the networking or spool offload PCE, your installation should provide its own recovery routine.
Exit 47

Job exit mask
Exit 47 is subject to suppression. The installation can suppress the exit either by implementing exit 2 to set the 47th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required
$HASPEQU, $PDDB, $SCR, $XPL, $HCT, $NHD, $HCCT, $DCT, $JQE, $JCT, $JCTX, $JOE, $PCE, $NJEWORK, $JRW, $SRW

Point of processing
JES2 invokes Exit 47 before receiving a job while performing spool offload processing or while transmitting an NJE job across the network. Before invoking Exit 47 JES2:
1. Allocates a dummy $JCT and $JQE. JES2 initializes these data areas with minimal information.
2. Receives the NJE data area and invokes Exit 47 to perform installation-specific processing.

After returning from Exit 47, JES2 determines if exit 47 indicated whether the NJE data area should be received. If exit 47 indicated the NJE data area should not be received, JES2 places the NJE job in hold on the transmitting node. Otherwise, JES2 continues to process the NJE job. You cannot use this exit to update IBM-defined JCT or JQE fields in the dummy JCT and dummy JQE, respectively. You can, however, update user-defined fields (such as JCTUSERx) or any $JCTX extensions you have created. JES2 propagates changes to ‘user’ fields to the $JCT and $JQE.

Programming considerations
The following are programming considerations for Exit 47:
• If your installation needs to process NJE data areas differently for spool offload processing and NJE processing, use field DCTDEVTP in the $DCT to determine the type of job JES2 is processing.
• If exit is being invoked for a job header, then the JQE address passed points to a dummy JOE (as indicated by X047BJQE). This JOE is not valid as input to $DOGJQE. For other header types, use $DOGJQE to access the JQE passed. See “Checkpoint control blocks” on page 382 for more information.

• Expanding the JCT Control Block
You can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 47 gets control
The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 47</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list which has the following structure:</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
</tbody>
</table>
Exit 47

**X047VERN** Indicates the version number of Exit 47

**XPLXITID** Exit identifier - 47

**XPLEXLEV** Version level of the exit

**X047IND** Indicates the type of NJE data area JES2 passed to Exit 47 for processing. A value of:

- **X047HDR** indicates an NJE job header was passed to Exit 47 for processing.
- **X047TRL** indicates an NJE job trailer was passed to Exit 47 for processing.
- **X047DSH** indicates an NJE data set header was passed to Exit 47 for processing.
- **X047RCCS** indicates an NJE RCCS header was passed to Exit 47 for processing.
- **X047BJQE** indicates that the JQE address in field X047JQE points to a working copy of the JQE that has not yet been added to the job queue. The working copy should not be used in services that expect the address of a real JQE. For example, this JQE address should not be used as input to $DOGJQE.

**X047COND** Condition byte

**X047RESP** Response byte

**X047HADR** Contains the address of the NJE data area

**X047DCT** Contains the address of the $DCT

**X047JQE** Contains the address of either a working copy of the $JQE or the address of a real $JQE. See the X047BJQE bit to determine the type of $JQE that this address points to.

**X047JCT** Contains the address of the $JCT

**X047PDDB** Contains the address of the $PDDB if Exit 47 is processing an NJE data set header. If Exit 47 is processing an NJE job header or trailer, a 0 is passed as the address.

**X047AREA** Contains the address of the NJEWORK area (JRW or SRW) for the receiver.

**X047SIZE** Indicates the length of the $XPL parameter list for Exit 47.

<table>
<thead>
<tr>
<th>Register content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-10</td>
<td>Not applicable to Exit 47</td>
</tr>
<tr>
<td>11</td>
<td>Address of the $HCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable to Exit 47</td>
</tr>
<tr>
<td>13</td>
<td>Address of the spool offload or networking $PCE</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address of Exit 47</td>
</tr>
</tbody>
</table>

Register contents when Exit 47 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
</table>

Exit 47: Modifying an NJE data area before receiving the rest of the NJE job
Exit 47

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 47</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list which has the following structure:</td>
</tr>
<tr>
<td></td>
<td><strong>X047IND</strong> Condition byte</td>
</tr>
<tr>
<td></td>
<td><strong>X047COND</strong> Response byte</td>
</tr>
<tr>
<td></td>
<td><strong>X047RESP</strong> Indicates the processing Exit 47 determined JES2 should perform after processing the NJE data area. A value of:</td>
</tr>
<tr>
<td></td>
<td>• <strong>X047TERM</strong> indicates Exit 47 determined the NJE data area should not be received. JES2 will stop processing the rest of the NJE job.</td>
</tr>
<tr>
<td>2-13</td>
<td>Not applicable to Exit 47</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Exit effector return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the job.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates JES2 should continue processing the job, but ignore any additional exits associated with this exit.</td>
</tr>
</tbody>
</table>

Coded example

Module HASX47A in SYS1.SHASSAMP contains a sample of Exit 47. Module HASXJECL in SYS1.SHASSAMP also contains an example.
Exit 48: Subsystem interface (SSI) SYSOUT data set unallocation

Function

This exit gives control to installation exit routines during unallocation of sysout data sets. This exit is taken later in processing than exit 34. When this exit is taken, all the characteristics have been merged from the SSOB into the PDDB. Through this exit, an installation can control whether JES2 will spin the SYSOUT data set.

Unlike installation exit 34, which is taken once for an unallocation, installation Exit 48 is taken once for each PDDB associated with an unallocation.

Environment

Task

User address space. You must specify USER on the ENVIRON= parameter of the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

Exit 48 receives control in supervisor state with a PSW key 0.

Recovery

ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine and can therefore provide no more than minimal recovery. Your exit routine should provide its own recovery.

Job exit mask

Exit 48 is subject to suppression. You can suppress Exit 48 by either implementing exit 2 to set the 48th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$HASPEQU, $HCCT, $IOT, $MIT, $PDDB, $SDB, $SJB, $JCT, $JCTX, JFCB

Point of processing

This exit is taken from HASCDSAL after JES2 has merged the characteristics from the SSOB into the PDDB.

Programming considerations

1. Job mask suppression is in effect for this exit.
2. Bit 7 of the response byte is set based on the setting of SSALSPIN in the SSOB: If SSALSPIN is on, bit 7 is set on. If SSALSPIN is off, bit 7 is set off.
3. By examining the setting of bit 7 in the response byte and the setting of IOT1SPIN in IOTFLG1, you can determine if the data set was originally allocated as spin and how it was unallocated:

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>IOT1SPIN</th>
<th>JES2</th>
<th>DATA SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>on</td>
<td></td>
<td>Spins the data set The application allocated the data set as spin.</td>
</tr>
<tr>
<td>on</td>
<td>off</td>
<td></td>
<td>Spins the data set The application allocated the data set as non-spin (either DALLOC was not set in dynamic allocation or FREE=CLOSE was not specified on the DD statement). The application used dynamic allocation to unallocate the data set.</td>
</tr>
<tr>
<td>off</td>
<td>on</td>
<td></td>
<td>Does not spin the data set The application allocated the data set as spin but the task terminated before closing the data set.</td>
</tr>
<tr>
<td>off</td>
<td>off</td>
<td></td>
<td>Does not spin the data set The application allocated the data set as non-spin and the data set remains non-spin.</td>
</tr>
</tbody>
</table>

4. Expanding the JCT Control Block

If the $JCT address is contained in field SJB$JCT, you can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.

Register contents when Exit 48 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a 24-byte parameter list with the following structure:</td>
</tr>
<tr>
<td>Byte 1 (+0)</td>
<td>Type of data set indicator</td>
</tr>
<tr>
<td>0-6</td>
<td>These bits are not part of the programming interface</td>
</tr>
<tr>
<td>7</td>
<td>0 – Do not spin the data set</td>
</tr>
<tr>
<td>Byte 3 (+2)</td>
<td>Response byte</td>
</tr>
<tr>
<td>bit 7</td>
<td>1 – Spin the data set. For more information, see &quot;Programming considerations&quot; on page 285</td>
</tr>
<tr>
<td>Byte 4 (+3)</td>
<td>This byte is not part of the programming interface</td>
</tr>
<tr>
<td>Byte 5 (+4)</td>
<td>SDB address.</td>
</tr>
<tr>
<td>Byte 6 (+5)</td>
<td>SJB address.</td>
</tr>
<tr>
<td>Byte 7 (+6)</td>
<td>JFCB address.</td>
</tr>
<tr>
<td>Byte 8 (+7)</td>
<td>PDDB address.</td>
</tr>
<tr>
<td>Byte 9 (+8)</td>
<td>IOT address</td>
</tr>
<tr>
<td>2-10</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Address of HCCT</td>
</tr>
</tbody>
</table>
Exit 48

12   N/A
13   Address of the register save area
14   The return address
15   The entry point address

Register contents when Exit 48 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-14</td>
<td>Unchanged</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX48A in SYS1.SHASSAMP contains a sample of Exit 48.
Exit 49: Job queue work select - QGOT

Function

This exit allows you to gain control whenever JES2 work selection processing has located a pre-execution job for a device. This includes work selected for JES2 and workload management (WLM) initiators. Exit 49 also gets control when the start job ($S J) command is used to start a batch job.

Exit 14, Job Queue Work Select - $QGET is not called for workload management (WLM) initiator work selection. Use this exit to instruct JES2 to accept or not accept such work. Exit 49 is generally easier to implement because it does not require that you copy JES2 decision-making algorithms into your exit routine.

Your exit routine is called by the $QGET routine in HASPJQS, which JES2 uses to acquire control of a job queue element (JQE). This JQE is actually a JQA (an artificial JQE) in update mode; you do not need to verify its update-mode status for calls to $DOGJQE. This JQA represents a job that is “BERT locked” by the PCE calling Exit 49. You can update this JQA without using any $DOGxxx services and therefore avoid disallowed $WAITs for this exit.

The $QGET routine scans the appropriate queue for an element that:
- is not held
- is not already acquired by a previous request to the job queue service routines
- has affinity to the selecting JES2 member
- has independent mode set in agreement with the current mode of the selecting member.

If this exit rejects the selected job, the JES2 job queue search routine ($QGET) will continue to search for another job (JQE), which if found will cause this exit to again receive control.

Note: Exit 49 is not called if:
- JES2 does not find a job
- Exit 14 already selected a job.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 49 in supervisor state and PSW key 1.

Recovery

The recovery that is in effect when $QGET is called is the same environment your exit will assume. As with every exit, you should provide your own recovery within the exit routine.
Exit 49

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCT, $JQE, $MIT, $PCE, $XPL

Point of processing

HASPJQS calls your exit routine with the address of the JQE that represents the job selected by the $QGET routine. Your exit routine has opportunity to examine this JQE and return to JES2 with the indication to select it for further processing or reject it.

HASPXEQ also calls exit 49 when processing the $S Job command. The exit is called once when the command is issued, under the HASPCOMM PCE while processing the $S Job command. If a job is rejected at this point, a message will be returned to the operator that the job cannot be processed. The second point the exit gets control is when the job is selected for execution under the execution PCE on the member where the job will execute. If a job is rejected at this point, a message is issued to the console that the requested job is not found.

Programming considerations

1. $WAIT is not allowed in EXIT49.
2. Exit 49 can perform duplicate job name check and instruct JES2 to bypass the normal duplicate job checks it would perform. You can also use the exit to allow a duplicate jobname to execute under certain situations. Setting X049NDUP causes JES2 checking for selected job to be bypassed.

Register contents when Exit 49 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Parameter List Address having the following structure:</td>
</tr>
</tbody>
</table>

Field Name

- **XPLID**: Eyecatcher ('$XPL')
- **XPLLEVEL**: Maintenance Level
- **XPLXITID**: Version Number
- **X049VERN**: Parameter list version
- **X049XID**: Exit 49 ID
- **X049IND**: Indicator byte flag bits:
  - **X049NORM**: Normal job selection
  - **X049SJOB**: $S job command issued
  - **X049SJSE**: $S job selection
  - **X049COND**: Condition byte:
  - **X049RESP**: Response byte
  - **X049SKIP**: Do not select this JQE
  - **X049NDUP**: Bypass duplicate job name check for this job

z/OS V1R11.0 JES2 Installation Exits
Disallow initiator job selection optimization

Attention: Turning on this flag may cause performance degradation.

X049SIZE
Length of parameter list

X049JQE
Address of the JQE

X049QGT
Address of the QGET parameter list or zero if the is $S JOB processing. The QGET parameter list has the following structure:

+0 (word 1) Address of the node table
+4 (word 2) Address of control block
  • PIT – if INWS
  • DCT – if OJTWS or OJTWSC
+8 (word 3) Address of class list (if applicable)
+12 (word 4) Address of the JQE
+16 (word 5) each byte is set as follows:
  +16 Length of the class list
  +17 Queue type (see the SQGET macro description for a list of these) This byte is set to ‘00’ for queue types INWS, OJTWSC, and OJTWS. Byte 18 (the type flag) is used to differentiate between these three queue types.
  +18 Work selection type flag
  +19 This byte is not part of the interface

2-10 Not applicable
11 Address of the HCT
12 Not applicable
13 Current PCE address
14 The return address
15 The entry address

Register contents when Exit 49 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 14</td>
<td>Unchanged</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

0     Tells JES2 that if additional exit routines are associated with the exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing.

4     Tells JES2 that even if additional exit routines are associated with the exit, ignore them; continue with normal processing. Set bit X049SKIP in the response byte to cause JES2 to select another job.
Coded example

Modules HASX49A and HASX49B in SYS1.SHASSAMP contains samples of Exit 49.
Exit 50: End of input

Function

This exit allows you to do the following:

- Selectively assign a job’s priority, affinity, execution node, SCHENV, and job class, and influence next phase of job processing based on an installation’s unique requirements and processing workload.
- Based on installation-defined criteria, terminate a job’s normal processing and selectively print or not print its output.
- Exit 50 allows input processing - end of input.

Note: See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.

Recommendations for implementing Exit 50

To access the submitting information for a job on the internal reader, you can use the following code segment:

```
USING JRW,R2      Est JRW addressability
USING RIDCWKAR,JRW Est IRWD addressability
USING SJB,R3      Est SJB addressability
SPACE 1
L     R2,X05xAREA Get JRW address
L     R3,RIDSJB    Get submitters SJB address
L     R4,SJBJCT    Get submitters JCT address
```

For STC and TSU INTRDRs, RIDSJB is zero because there is no submitting job in these situations.

Environment

Do not attempt to access anything in the JES2 address space in this exit. The JQE provided is always a JQA. The real JQE address is not available. It is not valid and is not necessary to perform a $DOGJQE.

Task

JES2 user environment task. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 50 in supervisor state and PSW key 1.

Recovery

$ESTAE recovery is in effect. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine, therefore, it can provide no more than minimal recovery. You should provide your own recovery within your exit routine.
Exit 50

Job exit mask

Exit 50 is subject to suppression. You can suppress Exit 50 by either setting the 50th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the initialization stream.

Mapping macros normally required

$JCT, $JCTX, $SPCE, $HASPEQU, $MIT, $JRW, $HCCT, $BUFFER, $DCT

Point of processing

This exit is taken in the subroutine CJOBEND or in the subroutine CJOBKILL of HASCSRIP in the User environment.

Programming considerations

1. To change affinity, set the X050SAF field in the $XPL work area using the $SETAFF macro.
   To allow the job to run on any member:
   $SETAFF REQUEST=ANY,AFFIELD=X050SAF
   To allow the job to run on only this member:
   $SETAFF REQUEST=CLEAR,AFFIELD=X050SAF
   $SETAFF REQUEST=ADD,AFFIELD=X050SAF,
   ID=CCTTOQUL

2. If MVS submits a job through an internal reader, it can force a job’s affinity to the local member. This can occur when the automatic restart manager restarts a job. The automatic restart manager expects the job to execute on a specific member, and will change the job’s affinity so the job can run on that specific member, if necessary. If the automatic restart manager has changed the job’s affinity, the X0501ARM flag in the XPL is on. You can test this flag and determine whether the affinity was changed. With that information, you can then decide whether to avoid changing the affinity.

3. To set independent mode for a job, the installation must turn on the bit X0501IND in X050FLG1.
   To put jobs that start with the characters 'IND' into independent mode:
   EXIT50 $ENTRY BASE=R12,SAVE=YES Set entry point
   LTR R10,R10 If JCT not present
   BZ RRET can't check jobname
   CLC =C'IND',JCTJNAME Job want independent mode?
   BNE RRET No, leave flags alone
   OI X050FLG1, X0501IND Set independent mode
   RRET $RETURN RC=0 Return to caller

4. To change the priority set X050PRIO in the XPL. The priority is contained in the 4 high-order bits of X050PRIO. For example, a value of 'C0' would be a priority 12. (See z/OS JES2 Initialization and Tuning Reference for further details on setting and changing job priority.)
   • To change the execution node, update X050XNOD with the half word binary value of the node. Use the $DEST macro to convert an EBCDIC node name to the internal binary representation of the node number
   • To change the job class, place the new job class in X050JCLS. This is honored only if the job is a batch job, not if it is an STC or TSU job.
• The exit can influence the next phase of the job in most circumstances. Place the next phase value in X050NEXT. X050NEXT is primed with the phase that JES2 believes is the correct next phase when the exit is called. The exit can place one of these values in X050NEXT:

$OUTPUT
Job will be placed in the OUTPUT queue unless JES2 has already determined that the job should be purged. In that case, X050NEXT is ignored.

SPURGE
Job will be placed in the PURGE queue.

Any other phase
JES2 will honor the request unless it has already determined that the job should be placed in the OUTPUT or PURGE phase.

The next phase can also be set through the return code in R15. If one or both of the specifications specify PURGE; then PURGE will be the next phase. If neither specify PURGE, but one or both specify OUTPUT; then the next phase will be OUTPUT.

5. Extending the JCT Control Block
You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (52, 53, 54, and 50) and the $QMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3k. You can have up to 8k of local extension regardless of SPOOL buffer size. For more information, see z/OS JES2 Macros.

6. This exit will not be taken under the following circumstances:
• The JES2 input service processor fails the job because JES2 does not identify a JOB card within the input stream.

7. If you need to change the scheduling environment, use the X050SENV field in the XPL.

8. Setting the X050AVF response bit does NOT influence the next phase of the job. To influence the next phase of the job, you must use the documented methods.

9. Accessing $NITs
The $NIT macro defines the characteristics of NJE nodes. The $NITs are arranged in a table that is indexed by the node number. The table of $NITs is in JES2 private storage and shadowed in a data space for use outside the JES2 address space. Installation exits can use three fields in the $NJEWORK work area to access the $NIT table. Installation exits can use these fields to access a $NIT without regard for what address space they are in. Because these fields are in the $NJEWORK data area, you can address them using the ‘NJ’ prefix or the prefix for the device dependent work area in which the $NJEWORK is embedded. Therefore, you can address NJENITAD as JRWNITAD in the $JRW.

The following code accesses the origin node’s NIT in an NJE JOB receiver exit:

```
USING NIT,R1 Est NIT addressability
SPACE 1
$ARMODE ON,SYSSTATE=SET,INIT=CCTZEROS Enter AR mode
SPACE 1
```
### 10. Determining the device type

Most exits need to determine the type of device that they are being called under. The $NJEWORK area has copies of $DCT fields that can help identify the device. Which method you use depends on the condition that you are testing for.

The field NJEDEVTP (that corresponds to DCTDEVTP) is a one byte flag that can be used to test for classes of devices. A test of the DCTNET bit in NJEDEVTP indicates that the exit is being called under a networking device. A compare of the byte to DCTINR indicates that the exit is being called under an internal reader. See the $DCT for the meaning of the bits in DCTDEVTP.

NJEDEVID corresponds to DCTDEVID. This is a 3 byte value that can uniquely identify a device. This is more often used when knowing what specific device you are running under. See the $DCT for the meaning of the fields.

### Register contents when Exit 50 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A code indicating:</td>
</tr>
<tr>
<td></td>
<td>0: Normal end of input.</td>
</tr>
<tr>
<td></td>
<td>4: Job has a JES2 control statement error.</td>
</tr>
<tr>
<td></td>
<td>8: Job has an SAF (security) failure.</td>
</tr>
<tr>
<td></td>
<td>12: Job failed work selection criteria (OFFLOADER only)</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>XPLID</td>
<td>The eyecatcher.</td>
</tr>
<tr>
<td>XPLEVEL</td>
<td>Version level for base XPL.</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>The exit ID number.</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version number for exit</td>
</tr>
<tr>
<td>X050IND</td>
<td>Indicator byte.</td>
</tr>
<tr>
<td>X050COND</td>
<td>Condition byte.</td>
</tr>
<tr>
<td>X050GJOB</td>
<td>Condition bit that specifies a normal job.</td>
</tr>
<tr>
<td>X050JECL</td>
<td>Condition bit that specifies a JECL error.</td>
</tr>
<tr>
<td>X050BSAF</td>
<td>Condition bit that specifies an SAF failure.</td>
</tr>
<tr>
<td>X050WSEL</td>
<td>Condition bit that specifies the job failed to meet work selection criteria.</td>
</tr>
</tbody>
</table>
X050RESP: Response byte.

X050NORM: Response bit that specifies to do normal process.

X050OUTP: Response bit that specifies to terminate with output.

X050PURG: Response bit that specifies to terminate job without printing the output.

X050AVF: Response bit that indicates the exit’s job verification failed.

XPLSIZE: Size of parm list, including base section.

X050JCT: Address of the JCT.

X050JQE: Address of update mode JQA.

X050DCT: Always zero. This field exists so that the XPL for exit 50 will be compatible with the XPL passed in exit 20. Most DCT fields can be accessed using corresponding fields in the JRW (pointed to by X050AREA). For example, DCTDEVTP can be accessed using field JRDEVTP.

X050AREA: Address of the JRW

X050PRIO: Job priority (Input/Output field)

X050FLG1: Flags

X050XNOD: Execution Node (Input/Output field)

X050SAF: Full system affinity mask (Input/Output)

X050SENV: Scheduling Environment (Input/Output field)

X050JCLS: Job class (Input/Output field)

X050NEXT: Next job phase (Input/Output field)

---

Register contents when Exit 50 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of a parameter list mapped by $XPL:</td>
</tr>
</tbody>
</table>

Register 2-9 Not applicable

10 Address of the JCT.

11 Address of the HCCT.

12 Not applicable

13 Address of a save area

14 Return address.

15 Entry address
Exit 50

**X050RESP**
Response byte that may be set by the exit before returning to JES2.

- **15**  A return code

A return code of:

- **0**  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no additional exit routines are associated with this exit continue normal processing.

- **4**  Tells JES2 to ignore any other exit routines associated with this exit and to continue normal processing.

- **8**  Tells JES2 to terminate normal processing and print the output.

- **12** Tells JES2 to terminate normal processing without printing the output.

**Coded example**

Modules HASX50A and HASX50B in SYS1.SHASSAMP contain samples of Exit 50.
Exit 51: Job phase change exit ($QMOD)

Function

Exit 51 gets control when a job is moving from one phase to another or when a job completes execution phase and is being re-queued for execution. It is called from $QMOD processing when the new phase for the job is not the same as the current phase, while from $QPUT when a job has completed execution and is being re-queued to execute again.

The exit can alter the new queue for the job, prevent or cause the job to re-execute, or change the job class, scheduling environment, or affinity of the job. It can also be used as a point of control to track jobs as they move through the various phases of JES2 processing.

The exit will not get control when attributes of the job (such as the class, scheduling environment or service class) change even if those changes cause $QMOD to re-queue the job to a new job queue.

Environment

Task

JES2 main task. You must specify ENVIRON=JES2 on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places Exit 51 in supervisor state and PSW key 1.

Restrictions

See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.

Recovery

No specific recovery is in place for this exit; however, most callers of $QMOD have a general recovery routine in place to deal with ABENDs. Your exit routine for this exit should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine, therefore, it can provide no more than minimal recovery. Provide your own recovery within your exit routine.

Job exit mask

Exit 51 is subject to suppression if a JCT is available at the time the exit is taken. You can suppress Exit 51 by setting the 51st bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$PCE, $JCT, $JCTX, $HCT, $JQE
Exit 51

Point of processing

Exit 51 is called by $QMOD or $QPUT while the JQE is still on the original job queue. A update mode JQA has been obtained and the BERT lock is held. If the job is not being occupied by the call, the JQA passed to the exit has been updated. However, the busy bits (and device ID) of the real JQE have not been updated at the time of the call.

Programming considerations

1. Exit 51 can be used to alter the new phase for the job. However, the new phase must be a later phase than the current one. If the new phase is not later, the change will be ignored.

2. If a JCT address is passed to the exit, the job has completed the current phase of processing including writing out the JCT. After the exit completes, the JCT will not be written by JES2. Installations should avoid updating the JCT in exit 51. Instead, earlier exits (such as exits 20 and 50) should be used to alter the JCT.

3. JCT extensions can be used to pass information from earlier exits to exit 51. Input processing can create both local and SPOOLed JCT extension. These can be used to pass information from user environment exits (such as 52, 53, and 54) to process in the JES2 main task. Local extensions are also supported in exits 2, 3, 4, and 20 so that a common set of services can be used for all job input processing.

4. Code in exit 51 must check the X051NOCH bit in X051COND and not attempt to change the phase of the job if this bit is on. In addition, if the X051RBLD bit is on in X051COND, the job is on the rebuild queue (an error queue) and will be deleted when it is no longer busy. Jobs on the rebuild queue should not be processed, because errors have already been detected in the checkpointed data structures. They are passed to exit 51 to allow complete tracking of the job.

5. Internal reader and NJE over TCP/IP processing occurs outside the JES2 address space. However, the code must reach across into the JES2 address space to perform some key functions (like build JQEs and queue them to the next phase). This processing is accomplished using a new service call $JQESERV. There is also a set of PCEs (the JQE Request Processors) in the main task that handle these requests (10 of them in all). It is under these PCEs that the $QMOD is done and that exit 51 is called. The code is careful not to $WAIT for any extended length of time so that the JQE Request Processors can process as many requests as possible. Adding a $CBIO to write the JCT in exit 51 will limit the number of jobs that can be processed by a given JQE Request Processor to one per $JCT write. The design point for internal readers was a single reader submitting hundreds of jobs at once and completing input processing as fast as possible. If this is the environment you are in, the extra I/O will impact performance. If jobs are arriving at a more leisurely rate, you can wait for a $CBIO.

Register contents when Exit 51 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

Field Name | Description
-------------|-------------
XPLID       | Eyecatcher |
Exit 51: Job phase change exit (QMOD)

XPLLEVEL  Version level for base XPL
XPLXITID  Exit ID number
XPLEXLEV  Version number for exit
X051IND  Indicator byte
X051COND  Condition byte

- X051RBLD  Job is on the re-build queue and will be purged when no longer busy.
- X051NOCH  Phase change is not allowed (X051RXEQ and X051RQUE ignored).
- X051RESP  Response byte
  - X051RXEQ  Job is being/should be requeued for execution (only valid if X051OLDQ is X051QXEQ). This bit is set by JES2 if the job is being requeued for execution. Exit 51 can alter the setting of this bit to cause the job to be requeued or not.
  - X051RQUE  X051NEWQ has been updated with new phase (X051NEWT no longer matches X051NEWQ). To change the next phase of the job, set X051RQUE on and set the next phase in X051NEWQ. You cannot change phase if X051NOCH is on. The new phase must be a later phase than the current phase (X051OLDQ).

- XPLSIZE  Size of parameter list, including base section.
- X051JCT  Address of JCT (or zero). If a JCT is passed, it will not be written after this call. If updated, the exit must write the JCT and wait for the I/O to complete.
- X051JQA  Address of JQA
- X051OLDQ  Current queue job is in. See below for valid values.
- X051OLDT  Current JQE type. See JQETYPE field in the JQE for valid values.
- X051NEWQ  New queue job is moving to. See below for valid values.
- X051NEWT  Proposed new JQE type. See JQETYPE field in the JQE for valid values.
- X051JOBC  JOB class of the job
- X051SENV  SCHENV value
- X051SAF  Full sysaff mask
- X051FLG1  Flags

- X0511IND  Independent system affinity.

**Note:** X051JOBC, X051SENV, X051SAF, X051IND are only meaningful if NEWQ is X051QCNV, X051QSET, X051QXEQ.

Queue values for X051OLDQ and X051NEWQ (not same as JQETYPE field in JQE).
Exit 51

X051QINP
Input queue

X051QCNV
Conversion queue

X051QSET
Setup queue

X051QXEQ
Execution queue

X051QSPN
Spin queue

X051QXMT
XMIT queue

X051QRCV
X051QRCV

X051QOUT
X051QOUT

X051QHRD
Hardcopy queue

X051QPUR
Purge queue

2-10 Not applicable
11 Address of the HCT
12 Not applicable
13 Address of the current PCE
14 Return address
15 Entry address

Register contents when Exit 51 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 14</td>
<td>Unchanged</td>
</tr>
<tr>
<td>15</td>
<td>A return code</td>
</tr>
</tbody>
</table>

A return code of:

0  Tells JES2 that if additional exit routines are associated with the exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing.

4  Tells JES2 that even if additional exit routines are associated with the exit, ignore them; continue with normal processing.

Coded example

Modules HASX51A and HASX51B in SYS1.SHASSAMP contains samples of Exit 51.
Exit 52: JOB JCL statement scan (JES2 user environment)

Function

Exit 52 allows you to process information specified on the JOB JCL statement for jobs submitted through internal readers or TCP/IP NJE. (For jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload, exit 2 is called for JOB JCL statements.) Exit 52 is invoked for the initial JOB statement and each continuation of the JOB card. The initial JOB card and all continuations are read before invoking the exit.

Using Exit 52 you can:

- Add, delete, change information specified on the JOB statement. If you are adding information, such as accounting information, you can create an additional JOB continuation statements.
- Indicate which spool volumes from which a job or transaction program should allocate spool space, if the installation did not implement spool partitioning through the JES2 initialization stream.
- Add JCL statements or JES2 control statements (JECL) to the job.
- Cancel, purge, or continue processing the job.
- Indicate whether additional job-related exits should be invoked for the job.

Recommendations for implementing Exit 52

Exit 52 is called for each card in the job statement (the original card and all continuations). Each time the exit is called, it will pass the current card image and the statement buffer. The statement buffer includes all the operands for the JOB statement concatenated in a single buffer. For example:

```
//TEST JOB (ACCOUNT), 'PROGRAMMER', COMMENT 1
//   CLASS=A, MSGCLASS=A, COMMENT 2
//   USER=TEST, PASSWORD=TEST COMMENT 3
```

In this case the exit will be called 3 times, once for each card and will pass (on all 3 calls) the following data in the statement buffer (pointed to by X052STMT):

```
(Account), 'PROGRAMMER', CLASS=A, MSGCLASS=A, USER=TEST, PASSWORD=TEST
```

To alter the processing of the JOB card, the exit can:

- Update the card image passed in X052CARD. This change shows up in the listing of the job.
- Update the statement buffer in X052STMT to add or modify the operands. This change does not show up in the listing of the job and is not passed to conversion processing (it only affects keywords input processing scans from the JOB card). If you update the statement buffer (X052STMT) in exit 52 and change the length of the buffer, you must update the field X052STME to indicate the new end of buffer (one byte past the last meaningful character).
- Add additional card images to the JCL stream.

You can add card images to the JCL stream by either queuing a single RJCB or a chain of RJCBs to the XPL, or by placing a card image after the current card into the area pointed to by X052JXWR and setting X052XSNC. In either case, when a card is added, the current card is re-scanned and the statement buffer is re-built.

© Copyright IBM Corp. 1988, 2009
Exit 52

Exit 52 is driven again for the updated statement, with X052SEC set to indicate this card has been presented to the exit previously.

When adding cards using RJCBs, use the RGETRJCB service (located in HASCSRIP) to obtain a free RJCB; then add it to one of the three RJCB queues in the XPL. Use the $CALL macro to invoke the RGETRJCB service. Register 1 on entry must be the JRW address. The RJCB address is returned in register 1.

The 80-byte card image to be added is placed into the field RJCBCARD. RJCBs are chained together using the RJCBRJCB field in the $RJCB. They are added to the job stream in the order they exist in the chain. To add an element to the chain you would move the current RJCB queue head in the $XPL into the RJCBRJCB field of the last RJCB you are adding and then set the address of the first RJCB element into the $XPL queue head. Be aware that multiple exit 4s might be using these queues so ensure that you do not lose existing entries on the queue.

**X052RJCP**
- Adds the card images before the first card in the current JOB statement.

**X052RJCA**
- Adds the card images after the last card in the current JOB statement. In this case, the card(s) are assumed to not be a continuation of the current job statement and the job card is not re-scanned.

**X052RJCC**
- Adds the card images after the current card. It is the callers’ responsibility to ensure that the proper continuation processing will occur.

When processing the last card in a JOB statement, the difference between adding a card to the X052RJCA queue and the X052RJCC queue is that the first will not re-scan the job card and the second will. You can also add a single card image after the current card using the X052JXWR field. In this case, the job card will be re-scanned just as if the card was added to the X052RJCC queue. To add information to the job JCL statement:

1. Move a comma into the last byte of the job statement image exit 52 is currently processing. The comma indicates that additional information follows on the job statement.

2. Move the information you want to add to the job statement to the area pointed to by X052JXWR and set the X052XSNC bit in the X052RESP byte to one. Setting X052RESP to X052XSNC indicates that the installation has supplied an additional job statement image.

3. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the job.

You can also add an additional job level JCL statement to the job as follows:

1. Ensure that the job statement image that exit 52 is currently processing is the last. exit 52 is processing the last job statement image if a comma is not in the last byte of the job statement image.

2. Place the job-level JCL statement in the area pointed to by X052JXWR and set the X052XSNC bit in the X052RESP byte to one. Setting X052RESP to X052XSNC indicates that the installation has supplied an additional job statement image.

3. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the job.

If you want to issue messages when you cancel or purge the job:
1. Generate the message text in exit 52.
2. Move the message text to area pointed to by X052JXWR and set the X052XSEM bit in X052RESP to one. Setting X052RESP to X052XSEM indicates that the installation exit has supplied an error message that will be added to the JCL listing.
3. Set register 15 to ‘X’08’ to indicate JES2 should cancel or purge the job.

The following indicators in the XPL can assist you in adding a card image to the current job statement:

**X052LOPR**
Current card has the last operand in the job statement. There may be additional continued comments after the current card.

**X052QUOT**
A quoted string is being continued from the current card to the next card. Pay attention if a card is being added after this card.

**X052CCMT**
The current card is a continued comment. Operand added to this card or after this card will not be processed.

**X052LAST**
This is the last card image in the JOB statement.

To assist you in processing the operands on a statement, you can use either of the following services to parse the statement buffer passed in X004STMT:

- Use the $SCAN facility to parse the operands with the standard $SCAN rules for statements. This give you the flexibility of $SCAN, but the parsing rules are not the same as normal JCL. See the $SCAN and $SCANTAB macros for additional information.
- Use the RCARDSCN service and $STMTTAB macro to parse the operands with standard JCL rules. This is the service used by JES2 input processing to parse the statement buffer. However, the RCARDSCN service only parses the operands and calls a processing routine to do all the conversions and storing of data. Conversion of data to binary to store into data areas is the responsibility of the processing routines. See the $STMTTAB macro for more information.

To access the submitting information for a job on the internal reader, you can use the following code segment:

```assembly
USING JRW,R2          Est JRW addressability
USING RIDCWKAR,JRW     Est IRWD addressability
USING SJB,R3           Est SJBJ addressability
SPACE 1
L   R2,X05xAREA       Get JRW address
L   R3,RIDSJB          Get submitter SJBJ address
L   R4,SJBCT           Get submitter JCT address
```

For STC and TSU INTRDRs, RIDSJB is zero because there is no submitting job in these situations.

**Environment**

**Task**

JES2 user environment. You must specify ENVIRON=USER on the $MODULE macro.
Exit 52

AMODE/RMODE requirements
AMODE 31, RMODE ANY

Supervisor/problem program
JES2 places exit 52 in supervisor state and PSW key 0.

Restrictions
- See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.
- Installation Exit 52 is not invoked for jobs such as SYSLOG, $TRCLOG, or JESMSG.
- Do not use this exit to set fields in the JCT; they will likely be overwritten by future processing.
- Installation Exit 52 is not invoked for jobs submitted through card readers, RJE, SAN and BSC NJE and SPOOL reload.

Recovery
$ESTAE is in effect and provides minimal recovery. Input Services will attempt to recover from any program check errors experienced by exit 52. However, you should not depend on JES2 for recovery.

Job exit mask
Exit 52 and all subsequent job-related installation exits can be suppressed after Exit 2 processes the initial job statement image. You can set the 52nd bit in the job exit suppression mask (JCTXMASK) or you can indicate the exit is disabled in the JES2 initialization stream.

Storage recommendations
If exit 52 requires work areas or additional storage, you can:
- Use the 80-byte work area, JCTXWRK, in the JCT
- Issue $GETMAIN to obtain additional storage

Mapping macros normally required
$JCT, $JCTX, $HCCT, $BUFFER, $MIT, $HASPEQU, $JRW

Point of processing
Installation Exit 52 can be invoked when JES2 encounters either:
- the JOB statement, this is called the initial job statement image.
- or a continuation of the JOB statement, this is called an additional JOB continuation statement image.

Module HASPINJR invokes installation Exit 52 for initial JOB statement images. Input service has obtained and initialized the job control table (JCT) and the IOT before calling installation Exit 52. After performing the processing you coded in Exit 2, input services complete scanning the JOB statement and allocate spool space for the job.

Module HASPINJR invokes installation Exit 52 for continuation JOB statement images.
Extending the JCT control block

1. You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (52, 53, 54, and 50) and the $QMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3K. You can have up to 8K of local extension regardless of SPOOL buffer size.

2. If you need to change the scheduling environment, use the JCTSCHEN field in the JCT.

Programming considerations

1. Be aware that when a JOB card image is passed to Exit 52, any //* comment cards embedded within that statement are also passed to the exit. For example, all of the following are passed:

   //ABC JOB
   //* COMMENT CARD
   // CLASS=A

   If within a //* comment you embed valid JOB card parameters, there is potential to cause confusion in your scan routine and lead to unpredictable results. Consider the following:

   //* CHANGED CLASS FROM ORIGINAL CLASS=B

2. When this exit adds or modifies cards, whether the change is sent over NJE (including SPOOL offload) depends on the statement type and the setting of option flags in the $XPL or $RJCB. Modified JECL cards (original and modified card are both JECL) are not sent over NJE. By default, all other changes are sent over NJE. To limit changes to only the local node, you can set the X052RLOC in the XPL (affects the current card) or set the RJCB3LOC bit in any RJCBs that are added.

3. Updating the statement buffer is only valid for parameters that have $STMTTABs in HASCSPRP.

4. Updates to the statement buffer are not passed to the converter and will not be seen by Exit 6.

5. Accessing $NITs

   The $NIT macro defines the characteristics of NJE nodes. The $NITs are arranged in a table that is indexed by the node number. The table of $NITs is in JES2 private storage and shadowed in a data space for use outside the JES2 address space. Installation exits can use three fields in the $NJEWORK work area to access the $NIT table. Installation exits can use these fields to access a $NIT without regard for what address space they are in. Because these fields are in the $NJEWORK data area, you can address them using the ‘NJE’ prefix or the prefix for the device dependent work area in which the $NJEWORK is embedded. Therefore, you can address NJENITAD as JRWNITAD in the $JRW.

   The following code accesses the origin node’s NIT in an NJE JOB receiver exit:

   USING NIT,R1
   Est NIT addressability
   SPACE 1
   $ARMODE ON, SYSSTATE=SET, INIT=CCTZEROS
   Enter AR mode
   SPACE 1
Exit 52

- Get origin node number
- Get NIT offset
- Get NIT address
- Get NIT ALET

6. Determining the device type

Most exits need to determine the type of device that they are being called under. The $NJEWORK area has copies of $DCT fields that can help identify the device. Which method you use depends on the condition that you are testing for.

The field NJEDEVTP (that corresponds to DCTDEVTP) is a one byte flag that can be used to test for classes of devices. A test of the DCTNET bit in NJEDEVTP indicates that the exit is being called under a networking device. A compare of the byte to DCTINR indicates that the exit is being called under an internal reader. See the $DCT for the meaning of the bits in DCTDEVTP.

NJEDEVID corresponds to DCTDEVID. This is a 3 byte value that can uniquely identify a device. This is more often used when knowing what specific device you are running under. See the $DCT for the meaning of the fields.

Register contents on entry to Exit 52

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
</tr>
<tr>
<td></td>
<td>XPLID</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID</td>
</tr>
<tr>
<td></td>
<td>XPLEXLEV</td>
</tr>
<tr>
<td></td>
<td>X052IND</td>
</tr>
<tr>
<td></td>
<td><strong>X052JOB</strong></td>
</tr>
<tr>
<td></td>
<td>X052COND</td>
</tr>
<tr>
<td></td>
<td><strong>X052CONT</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052SEC</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052RESP</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052XSNC</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052XSEM</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052JCMT</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052KILL</strong></td>
</tr>
<tr>
<td></td>
<td><strong>X052PURG</strong></td>
</tr>
</tbody>
</table>
Exit 52

X052RLOC Changed or added cards are not sent through NJE (set RJCB3LOC in current RJCB)

XPLSIZE Size of parameter list, including base section
X052CARD 80-byte card image address
X052FLGX Pointer to exit flags (same as JRWFLAGX)
X052JXWR 80-byte exit work area address (same as JCTXWRK)
X052JCT JCT address
X052JQE Update mode JQA address
X052AREA JRW address
X052STMT Concatenated statement buffer. This is all the operands on all continuations cards for this statement
X052STME End of statement+1 pointer (in buffer)
X052STML Statement label (job name)
X052STMV Statement verb (JOB)
X052RJCP RJCBs to add before this JOB statement
X052RJCA RJCBs to add after this JOB statement
X052RJCC RJCBs to add after the current card
X052FLG1 Statement flag byte

X052LOPR Last operand is on the current card
X052QUOT Unfinished quote at end of current card
X052CCMT Current card is a continued comment

X052OCLS Override job class (batch jobs only)

1 Address of a 3-word parameter list with the following structure:
Word 1 (+0) points to the JOB statement image buffer
Word 2 (+4) points to the exit flag byte, JRWFLAGX, in the $JRW
Word 3 (+8) points to the JCTXWRK field in the $JCT

2-9 Not applicable
10 Address of the $JCT
11 Address of the HCCT
12 Not applicable
13 Address of an available save area
14 Return address
15 Entry address

Register contents when Exit 52 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 13</td>
<td>Not applicable</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>
Exit 52

A return code of:

0  Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit, continue with normal HASPINJR processing.

4  Tells JES2 to ignore any additional exit routines associated with this exit and to continue with normal HASPINJR processing.

8  Tells JES2 to cancel the job; output (the incomplete JCL images listing) is produced.

12 Tells JES2 to purge the job; no output is produced.

Note: If register 10 contains 0 (the JCT is unavailable), JES2 ignores any return code greater than 4.

Coded example

Module HASX52A in SYS1.SHASSAMP contains a sample of exit 52.
Exit 53: JOB statement accounting field scan (JES2 user environment)

Function

This exit allows you to provide an exit routine for scanning the JOB statement accounting field and for setting the corresponding fields in the appropriate JES2 control blocks. Exit 53 gets control for jobs submitted through internal readers or TCP/IP NJE. For jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload, exit 3 is called to process the JOB statement accounting field.

You can use your exit routine to interpret the variables in the accounting field and, based on this interpretation, decide whether to cancel the job.

Use this exit to record alterations to the accounting field; they will not appear on the user's output but are reflected in the JCT and the SMF type 6 record is written.

This exit is associated with the existing HASPRSCN accounting field scan sub-routine. You can write your exit routine as a replacement for HASPRSCN. Or, you can use a return code to direct input processing to call HASPRSCN after your exit routine has executed. In either case, when this exit is implemented and enabled, JES2 treats your exit routine as the functional equivalent of HASPRSCN. The specification of the ACCTFLD parameter on the JOBDEF initialization statement, which normally determines whether JES2 is to call HASPRSCN, becomes an additional factor in determining whether your exit routine is to be called. The exit is taken only if the ACCTFLD= parameter on the JOBDEF initialization statement is specified as either REQUIRED or OPTIONAL. The exit is not taken if ACCTFLD=IGNORE is specified. When it is called, your exit routine, rather than the ACCTFLD parameter, determines whether HASPRSCN is to be executed as an additional scan of the accounting field. For a complete explanation on how the ACCTFLD parameter is specified, see z/OS JES2 Initialization and Tuning Reference. The relationship of HASPRSCN to this exit is described in more detail in the “Other Programming Considerations” below.

Related exits

Use Exit 52 to alter the accounting information and supply new accounting information at the time the entire JOB statement is first scanned.

Recommendations for implementing Exit 53

To access the submitting information for a job on the internal reader, you can use the following code segment:

```
USING JRW,R2           Est JRW addressability
USING RIDCWKAR,JRW     Est IRWD addressability
USING SJB,R3           Est SJB addressability
SPACE 1
L  R2,X05XAREA         Get JRW address
L  R3,RIDSJB           Get submitters SJB address
L  R4,SJBJCT           Get submitters JCT address
```

For STC and TSU INTRDRs, RIDSJB is zero because there is no submitting job in these situations.
Exit 53

Environment

Task

JES2 user environment. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

AMODE 31, RMODE ANY

Supervisor/problem program

JES2 places Exit 53 in supervisor state and PSW key 0.

Restrictions

See Appendix A, “JES2 exit usage limitations,” on page 373 for a listing of specific instances when this exit will be invoked or not invoked.

Recovery

$ESTAE recovery is in effect. Input processing recovery will attempt to recover from program check errors, including program check errors in the exit routine. However, as with every exit, your exit routine for this exit should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine. Therefore, it can provide no more than minimal recovery. You should provide your own recovery within your exit routine.

Job exit mask

Exit 53 is subject to suppression. You can suppress Exit 53 by either implementing exit 52 to set the 53rd bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$JCT, $JCTX, $HCCT, $BUFFER, $HASPEQU, $JRW

Point of processing

This exit is taken from the JES2 user environment, the JOB statement processing routine of HASCINJR. If HASPRSCN is to be called, the exit occurs after JES2 has scanned the entire JOB statement, but before the execution of the HASPRSCN accounting field scan subroutine. The JCT has been initialized with the JES2 and installation defaults; in addition, those fields of the JCT that correspond to JOB statement parameters other than accounting field parameters have been set. The accounting field image is passed in X053ACCT and the length in X053ACTL.

Table 11 lists some of the fields in the JCT that you can modify.

Table 11. Selected JES2 Job Control Table Fields

<table>
<thead>
<tr>
<th>Field Name in JCT</th>
<th>Length (Bytes)</th>
<th>Field</th>
<th>Bit</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCTSMFLG</td>
<td>1</td>
<td>SMF Flags</td>
<td>0–1</td>
<td>These bits are not part of the interface</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>If set, IEFUSO exit not taken</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3–4</td>
<td>These bits are not part of the interface</td>
<td>–</td>
</tr>
<tr>
<td>Field Name in JCT</td>
<td>Length (Bytes)</td>
<td>Field</td>
<td>Bit</td>
<td>Meaning</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>---------------------</td>
<td>-----</td>
<td>-------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>JCTJOBFL</td>
<td>1</td>
<td>Job Flags</td>
<td>0</td>
<td>Background job</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>TSO/E (foreground) job</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Started task</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>No job journaling</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>No output</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>TYPRUN=SCAN</td>
<td>1,2,3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>TYPRUN=COPY</td>
<td>2,3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>Job restartable</td>
<td>1,2,8</td>
</tr>
<tr>
<td>JCTJBOPT</td>
<td>1</td>
<td>Job Options</td>
<td>0</td>
<td>/*PRIORITY card was read and value is in priority field (JCTIPRIO)</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>/*SETUP card was read</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>TYPRUN=HOLD was specified</td>
<td>1,2,4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>No job log for this job</td>
<td>1,2,6,8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Execution batch job</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>The job was read through an internal reader</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>The job was rerun</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>This bit is not part of the interface</td>
<td>–</td>
</tr>
<tr>
<td>JCTJOBID</td>
<td>8</td>
<td>JES2 JOB identifier</td>
<td>–</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>JCTJNAME</td>
<td>8</td>
<td>Job name</td>
<td>–</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>JCTPNAME</td>
<td>20</td>
<td>Programmer name</td>
<td>–</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>JCTMCLAS</td>
<td>1</td>
<td>Message class</td>
<td>–</td>
<td></td>
<td>1,4</td>
</tr>
<tr>
<td>JCTJCLAS</td>
<td>1</td>
<td>Job class</td>
<td>–</td>
<td></td>
<td>1,4</td>
</tr>
<tr>
<td>JCTIPRIO</td>
<td>1</td>
<td>Priority</td>
<td>–</td>
<td></td>
<td>1,5</td>
</tr>
<tr>
<td>JCTROUTE</td>
<td>4</td>
<td>Route code of input device (binary)</td>
<td>–</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>JCTINDEV</td>
<td>8</td>
<td>Input device name</td>
<td>–</td>
<td></td>
<td>–</td>
</tr>
<tr>
<td>JCTACCTN</td>
<td>4</td>
<td>Account number</td>
<td>–</td>
<td></td>
<td>1,6</td>
</tr>
<tr>
<td>JCTROOMN</td>
<td>4</td>
<td>Room number</td>
<td>–</td>
<td></td>
<td>1,6,8</td>
</tr>
<tr>
<td>JCTETIME</td>
<td>4</td>
<td>Estimated real-time job will run</td>
<td>–</td>
<td></td>
<td>1,6,8</td>
</tr>
<tr>
<td>JCTESTLN</td>
<td>4</td>
<td>Estimated count of output lines (in thousands)</td>
<td>–</td>
<td></td>
<td>1,6,8</td>
</tr>
<tr>
<td>JCTESTPU</td>
<td>4</td>
<td>Estimated number of output cards punched</td>
<td>–</td>
<td></td>
<td>1,6,8</td>
</tr>
<tr>
<td>JCTESTBY</td>
<td>4</td>
<td>Estimated number of SYSOUT bytes</td>
<td>–</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
Exit 53

Table 11. Selected JES2 Job Control Table Fields (continued)

<table>
<thead>
<tr>
<th>Field Name in JCT</th>
<th>Length (Bytes)</th>
<th>Field</th>
<th>Bit</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCTESTPG</td>
<td>4</td>
<td>Estimated number of output pages</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTFORMS</td>
<td>8</td>
<td>Job Forms</td>
<td>1,6,8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTCPYCT</td>
<td>1</td>
<td>Job copy count (binary)</td>
<td>1,6,8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTLINCT</td>
<td>1</td>
<td>Lines per page (binary)</td>
<td>1,6,8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTPROCOUT</td>
<td>4</td>
<td>Default print routing (binary)</td>
<td>1,7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTPUOUT</td>
<td>4</td>
<td>Default punch routing (binary)</td>
<td>1,7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTPROCN</td>
<td>8</td>
<td>Procedure DD name</td>
<td>1,2,8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

1. Can be modified by installation routine.
2. Preset from JOBCLASS(v) initialization statement according to job class
3. Preset from JOB statement
4. From JOB statement, if specified; otherwise according to input device as established at JES2 initialization (for example, in RDR(nn)).
5. Exit 53 can use field JCTIPRIO to force a priority for a job subject to the limitations of the input device’s priority increment and priority limit values. When exit 53 receives control, a value of C** in JCTIPRIO indicates a priority has not been forced by an exit routine. If you want to force a priority in exit 53, set JCTIPRIO to a value between 0 and 15 in the low-order four bits on the field.

**Note:** Whether you may set field JCTIPRIO and the allowable values depend on the specific exit.
6. Set by the routine (HASPRSCN) used by JES2 to scan the account field of the JOB statement. Exit 3 can specify that JES2 cannot call HASPRSCN.
7. Preset according to an input device initialization parameter (for example, RDR(nn)). If not set at initialization, the parameter defaults to the job input source value (LOCAL or RMT(nn)). Can be modified by a /*ROUTE statement after the scan exit.
8. Can be modified by a /*JOBPARM statement after the scan exit.

**Extending the JCT control block**

You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (52, 53, 54, and 50) and the $QMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3K. You can have up to 8K of local extension regardless of SPOOL buffer size.

**Programming considerations**

1. The accounting field resides in a 144-byte work area pointed to by X053ACCT in the XPL passed to the exit in register 0.
2. If you need to verify the existence of a JOB rather than a started task (STC) or TSO/E logon, this can be done by comparing the JCTJOBID field to a "J". The presence of a "J" indicates the existence of a JOB.

3. If you need to change the scheduling environment, use the JCTSCHEN field in the JCT.

4. The ACCTFLD parameter on the JOBDEF statement indicates whether JES2 should scan the accounting field of a JOB statement. For further details concerning the use of the ACCTFLD parameter, see z/OS JES2 Initialization and Tuning Reference.

If the ACCTFLD parameter indicates that the scan should be performed, and if this exit is implemented and enabled, input processing will call your exit routine to perform the scan. If your exit routine passes a return code of 0 or 4 to JES2, input processing will call the existing HASPRSCN accounting field scan subroutine after your routine has executed. Note that if both routines are to be called, your routine should not duplicate HASPRSCN processing. For example, your routine should not set the fields in the JCT that are set by HASPRSCN. However, if your routine passes a return code of 8 or 12 to JES2, it causes JES2 to suppress execution of HASPRSCN. If the ACCTFLD parameter indicates that the scan should be performed but this exit is disabled, only HASPRSCN will be called; your exit routine is not called and is not given the opportunity to allow or suppress HASPRSCN execution. If the ACCTFLD parameter indicates that a scan should not be performed, your exit routine will not be called, even if this exit is enabled, and execution of HASPRSCN is also suppressed.

5. The ACCTFLD parameter on the JOBDEF statement indicates whether JES2 should cancel a job if the accounting field on the JOB statement is invalid or if a JCL syntax error has been detected during input processing. Note that your exit routine can affect this termination processing. For example, ACCTFLD=REQUIRED indicates that JES2 should scan the accounting field, the job should be canceled if the accounting field is invalid, and the job should be canceled if a JCL syntax error has been found. If you pass a return code of 8 to JES2, HASPRSCN is not called. Therefore, it cannot terminate a job with an invalid accounting field, even though ACCTFLD=REQUIRED. Also note that HASPRSCN scans the accounting field passed in X053ACCT. Therefore, if your routine alters this field, you affect HASPRSCN processing.

6. The specification of the ACCTFLD parameter is stored in the HCCT, in field CCTJOPTS. If your exit routine is meant to completely replace HASPRSCN, you may want to access this field for use by your algorithm.

7. Typically, use this exit, rather than Exit 52, to alter the JCT directly. If you use Exit 52 to alter the JCT, later processing might override your changes. The job exit mask and the spool partitioning mask are exceptions. See note 2 of Exit 52 for more information.

8. An 80-byte work area pointed to by X053JXWR in the XPL is available for use by your routine. If your routine requires additional work space, use the $GETMAIN macro to obtain storage, and the $FREMAIN macro to return it to the system when your routine has completed.

9. When passing a return code of 12, your exit routine can pass an installation-defined error message to JES2 to be added to the JCL data set rather than the standard error message. To send an error message, generate the message text in your exit routine, move it to area pointed to by X053JXWR, and set the X053XSEM bit in X053RESP to one.

Note: The standard error message, $HASP110, still appears in SYSLOG on this path, in addition to the installation-defined message. However, only
the installation message will be placed in the JCL data set and no WTO
will be issued for the installation-defined message unless Exit 53 issues
the WTO itself.

10. If there is no accounting field on a JOB statement, the length passed by JES2
to the exit routine in R0 is zero. Your exit routine should take this possibility into
account.

11. If you intend to use this exit to process nonstandard accounting field
parameters, you should either suppress later execution of HASPRSCN or you
should code your exit routine to delete nonstandard parameters before passing
control to HASPRSCN. If you do neither, that is, if you allow HASPRSCN to
receive the nonstandard parameters, it might cancel the job because of an
illegal accounting field depending on how the ACCTFLD parameter on the
JOBDEF statement is specified.

If you change the length of the accounting field, you must reload the length
into field JRWACCTL.

12. There are three job class fields (JCTJCLAS, JCTCLASS, and JCTAXCLS) in
the JCT. JCTJCLAS is the initial job execution class as set during input
processing and used when building the JQE during that processing.
JCTCLASS is the actual execution class. After input processing it contains the
same value as JCTJCLAS, but it might be updated when the job executes if a
$T command was used to update the job’s class before execution. Therefore,
JCTJCLAS and JCTCLASS could be different. JCTAXCLS is a copy of the
actual execution class (JCTCLASS) that is propagated into the network JOB
trailer. Do not use any exit routine to set the JCTAXCLS field.

If you intend to use an exit 53 routine to change the execution class of a job,
be certain to set both the JCTJCLAS and JCTCLASS fields.

13. **Accessing $NITs**

The $NIT macro defines the characteristics of NJE nodes. The $NITs are
arranged in a table that is indexed by the node number. The table of $NITs is
in JES2 private storage and shadowed in a data space for use outside the
JES2 address space. Installation exits can use three fields in the $NJEWORK
work area to access the $NIT table. Installation exits can use these fields to
access a $NIT without regard for what address space they are in.

Because these fields are in the $NJEWORK data area, you can address them
using the ‘NJE’ prefix or the prefix for the device dependent work area in which
the $NJEWORK is embedded. Therefore, you can address NJENITAD as
JRWNITAD in the $JRW.

The following code accesses the origin node’s NIT in an NJE JOB receiver
exit:

```plaintext
USING NIT,R1  Est NIT addressability
SPACE 1
$ARMODE ON,SYSSTATE=SET,INIT=CCTZEROS Enter AR mode
SPACE 1
LLGH R1,JRRDNOD Get origin node number
MH R1,CCTNITSZ Get NIT offset
AL R1,JRNITBL Get NIT address
LAM AR1,AR1,JRNITAL Get NIT ALET
```

14. **Determining the device type**

Most exits need to determine the type of device that they are being called
under. The $NJEWORK area has copies of $DCT fields that can help identify
the device. Which method you use depends on the condition that you are
testing for.

The field NJEDEVTP (that corresponds to DCTDEVTP) is a one byte flag that
can be used to test for classes of devices. A test of the DCTNET bit in
NJEDEVTP indicates that the exit is being called under a networking device. A compare of the byte to DCTINR indicates that the exit is being called under an internal reader. See the $DCT for the meaning of the bits in DCTDEVTP.

NJEDEVID corresponds to DCTDEVID. This is a 3 byte value that can uniquely identify a device. This is more often used when knowing what specific device you are running under. See the $DCT for the meaning of the fields.

---

## Register contents when Exit 53 gets control

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
<tr>
<td></td>
<td>XPLID Eyecatcher</td>
</tr>
<tr>
<td></td>
<td>XPLLEVEL Version level for base XPL</td>
</tr>
<tr>
<td></td>
<td>XPLXITID Exit ID number</td>
</tr>
<tr>
<td></td>
<td>XPLEXLEV Version number for exit</td>
</tr>
<tr>
<td>X053IND</td>
<td>Indicator byte</td>
</tr>
<tr>
<td>X053COND</td>
<td>Condition byte</td>
</tr>
<tr>
<td>X053RESP</td>
<td>Response byte</td>
</tr>
<tr>
<td>X053XSEM</td>
<td>Exit supplied error message in X052JXWR</td>
</tr>
<tr>
<td>X053SKIP</td>
<td>Skip default accounting field scan</td>
</tr>
<tr>
<td>X053KILL</td>
<td>Kill current job (queue job to OUTPUT processing)</td>
</tr>
<tr>
<td>XPLSIZE</td>
<td>Size of parameter list, including base section</td>
</tr>
<tr>
<td>X053ACCT</td>
<td>Address of accounting field</td>
</tr>
<tr>
<td>X053FLGX</td>
<td>Pointer to exit flags (same as JRWFLAGX)</td>
</tr>
<tr>
<td>X053JXWR</td>
<td>80-byte exit work area address (same as JCTXWRK)</td>
</tr>
<tr>
<td>X053JCT</td>
<td>JCT address</td>
</tr>
<tr>
<td>X053JQE</td>
<td>Update mode JQA address</td>
</tr>
<tr>
<td>X053AREA</td>
<td>JRW address</td>
</tr>
<tr>
<td>1</td>
<td>Address of a 3-fullword parameter list</td>
</tr>
</tbody>
</table>

**Word 1 (+0)** points to the accounting field (JCTWORK in the JCT)
Exit 53

| Word 2 (+4) | pointers to the exit flag byte, JRWFLAGX in the JRW |
| Word 3 (+8) | points to the JCTXWRK field in the JCT |

| 2-10 | Not applicable |
| 11   | Address of the HCCT |
| 12   | Not applicable |
| 13   | Available save area address |
| 14   | Return address |
| 15   | Entry address |

**Register contents when Exit 53 passes control back to JES2**

| 0-13 | N/A |
| 14   | Return address |
| 15   | Return code |

A return code of:

- **0**: Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit, use the current setting of the ACCTFLD parameter on the JOBDEF statement to determine whether to execute the HASPRSCN subroutine.
- **4**: Tells JES2 to ignore any other exit routines associated with this exit and to use the current setting of the ACCTFLD parameter on the JOBDEF statement to determine whether to execute HASPRSCN.
- **8**: Tells JES2 to suppress execution of HASPRSCN and to complete job card processing.
- **12**: Tells JES2 to cancel the job because an illegal accounting field has been detected. Tells JES2 to suppress execution of HASPRSCN and to queue the job for output; output (the incomplete JCL images listing) is produced.

**Coded example**

Module HASX53A in SYS1.SHASSAMP contains a sample of Exit 53.
Exit 54: JCL and JES2 control statement scan (JES2 user environment)

Function

This exit allows you to provide an exit routine for scanning JCL and JES2 control statements for jobs submitted through internal readers (including TSO SUBMIT command) or TCP/IP NJE. For jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload, exit 4 is called to process JCL and JES2 control statements (JECL). If this exit is implemented and enabled, it is taken whenever JES2 encounters a JCL or JES2 control statement.

Note: JOB statements are not included in the scan.

For JCL statements, your exit routine can interpret JCL parameters and, based on this interpretation, decide whether JES2 should cancel the job, purge the job, or allow the job to continue normally. Your routine can also alter JCL parameters and supply additional JCL parameters. If necessary, in supplying expanded JCL data, your routine can pass a JCL continuation statement back to JES2 or add statements before or after the current JCL statement.

For JES2 control statements, your routine can interpret the JES2 control parameters and subparameters and, based on this interpretation, decide whether JES2 should cancel the job, purge the job, or allow the job to continue normally. For any JES2 control statement, you can write your exit routine as a replacement for the standard JES2 control statement processing, suppressing execution of the standard JES2 scan, or you can perform your own (partial) processing and then allow JES2 to execute the standard control statement processing. Also, your routine can alter a JES2 control statement and then pass the modified statement back to JES2 for standard processing, or your routine can pass an entirely new JES2 control statement back to JES2, to be read (and processed) before or after the current control statement.

This exit also allows you to process your own installation-specific JES2 control statements or to implement new, installation-specific subparameters for existing JES2 control statements.

This exit gets control when JES2 detects a JES2 control statement or JCL statement within a job. JES2 also gives control to your exit routine when JES2 detects a JES2 control statement or JCL statement outside a job. JES2 also gives control to your exit routine when it detects a JCL continuation statement.

Recommendations for implementing Exit 54

To access the submitting information for a job on the internal reader, you can use the following code segment:

```
USING JRW,R2          Est JRW addressability
USING RIDCWKAR,JRW     Est IRWD addressability
USING SJB,R3           Est SJB addressability
SPACE 1
L R2,X05XAREA          Get JRW address
L R3,RIDSJBJ           Get submitters SJB address
L R4,SJBJCT            Get submitters JCT address
```
Exit 54

For STC and TSU INTRDRs, RIDSJB is zero because there is no submitting job in these situations.

Environment

Task

JES2 user environment. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 54 in supervisor state and PSW key 0.

Restriction

JES2 does not invoke this exit for JCL from cataloged procedures. See Appendix A, JES2 exit usage limitations for other specific instances when this exit will be invoked or not invoked.

Recovery

$ESTAE recovery is in effect. The recovery routine established by JES2 attempts to recover from program check errors, including program check errors in the exit routine itself. However, as with every exit, your exit routine should not depend on JES2 for recovery. JES2 cannot anticipate the exact purpose of your exit routine. Therefore, it can provide no more than minimal recovery. You should provide your own recovery within your exit routine.

Job exit mask

Exit 54 is subject to suppression. You can suppress Exit 54 by either implementing exit 52 to set the 54th bit in the job exit suppression mask (JCTXMASK) or disabling the exit in the JES2 initialization stream.

Mapping macros normally required

$HCCT, $JCT, $JCTX, $BUFFER, $HASPEQU, $JRW

Point of processing

This exit is taken from HASCINJR in the user environment. The exit occurs in input processing’s main processing loop, after the entire JES2 control statement or JCL statement (including JCL continuations) has been read but before it has processed any keywords on the statement. The statement may be outside a valid job (that is, when there is no current job structure active on the reader).

This exit is invoked for jobs submitted through the internal reader or TCP/IP NJE. It is not invoked for jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload.
Programming considerations

1. This exit is taken once for each control statement (except for JOB statements) encountered by JES2. X054IND indicates whether the current statement is a JCL statement or a JES2 control statement. Your exit routine gets control for //* comment, /* (generated), and /* PRIORITY JES2 control statements.

2. During input processing, JES2 writes the JCL records to a JCL data set. If an error occurs during input processing, it is the JCL data set that is printed when the job goes through output processing. If the job is successfully processed by input processing, the JCL data set is the input for the converter. The converter produces a JCL images data set that is printed when the job goes to output processing after being successfully processed by input processing.

3. Exit 54 is called for each card in a JCL statement (the original card and all continuations) and for each JES2 control statement. Each time the exit is called, it is passed the current card image and the statement buffer. The statement buffer is all the operands for the JCL statement or JES2 control statement concatenated in a single buffer. For example:

```
//OUTSET DD SYSOUT=H,OUTPUT=*.OUT1,
// DCB=(LRECL=8000,RECFM=FB,BLKSIZE=8000)
```

In this case the exit will be called 2 times, once for each card and will be passed (on both calls) the following data in the statement buffer (pointed to by X054STMT):

```
SYSOUT=H,OUTPUT=*.OUT1,DCB=(LRECL=8000,RECFM=FB,BLKSIZE=8000)
```

To alter the processing of the JCL statement or JES2 control card, the exit can either:

- Update the card image passed in X054CARD. This change will show up in the listing of the job.

- Update the statement buffer in X054STMT to add or modify the operands. This change does not show up in the listing of the job and is not passed to conversion processing (it only affects keywords input processing scans from the JCL/JECL card). If you update the statement buffer (X054STMT) in Exit 54 and change the length of the buffer, you must update the field X054STME to indicate the new end of buffer (one byte past the last meaningful character).

- Add additional card images to the JCL stream. Adding card images to the JCL stream can be accomplished by either queuing a single RJCB or a chain of RJCBs to the XPL or by placing a card image to be placed after the current card into the area pointed to by X054JXWR and setting X054XSNC. In either case, when a card is added, the current card is re-scanned and the statement buffer is re-built. Exit 54 will be driven again for the updated statement, with X054SEC set to indicate this card has been presented to the exit previously.

When adding cards using RJCBs, use the RGETRJCB service (located in HASCSRIP) to obtain a free RJCB; then add it to one of the three RJCB queues in the XPL. Use the $CALL macro to invoke the RGETRJCB service. Register 1 on entry must be the JRW address. The RJCB address is returned in register 1.

The 80-byte card image to be added is placed into the field RJCBCARD. RJCBs are chained together using the RJCBRJCB field in the $RJCB. They are added to the job stream in the order they exist in the chain. To add an element to the chain you would move the current RJCB queue head in the $XPL into the RJCBRJCB field of the last RJCB you are adding and then set...
the address of the first RJCB element into the $XPL queue head. Be aware that multiple exit 4s might be using these queues so ensure that you do not lose existing entries on the queue.

**X054RJCP**  
Adds the card images before the first card in the current JCL statement or before the JES2 control card.

**X054RJCA**  
Adds the card images after the last card in the current JCL statement. In this case, the cards are assumed not to be a continuation of the current JCL statement, and the JCL cards are not re-scanned.

**X054RJCC**  
Adds the card images after the current card. It is the callers’ responsibility to ensure that the proper continuation processing will occur.

When processing the last card in a JCL statement or when processing a JES2 control statement, the difference between adding a card to the X054RJCA queue and the X054RJCC queue is that the first will not rescan the current statement and the second will do.

You can also add a single card image after the current card using the X054JXWR field. In this case, the JCL statement will be re-scanned just as if the card was added to the X054RJCC queue. To add information to a JCL statement:

a. Move a comma into the last byte of the operand on the JCL card image (X054CARD) that exit 54 is currently processing. The comma indicates additional information follows this JCL statement.

b. Move the information you want to add to the JCL statement to the area pointed to by X054JXWR and set the X054XSNC bit in the X054RESP byte to one. Setting X054RESP to X054XSNC indicates that the installation has supplied an additional JCL statement image.

c. Set register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the statement.

You can also add an additional JCL statement to the job by:

a. Ensuring that the JCL card image that exit 54 is currently processing is the last for the current statement (X054LOPR is on). Exit 54 is processing the last JCL statement image if a comma is not in the last byte of the JCL operand on the card image.

b. Placing the JCL statement in the area pointed to by X054JXWR and set the X054XSNC bit in the X054RESP byte to one. Setting X054RESP to X054XSNC indicates that the installation has supplied an additional JCL statement image.

b. Setting register 15 to X’00’ or X’04’ depending on whether you want to invoke additional installation exits to process the JCL or JECL card.

For JECL statements, because there are no formal rules for the format of the statement, the statement buffer will contain all the text after the VERB on the JECL statement. The following is an example of a JOBPARM JECL statement and the associated statement buffer:

```
/*JOBPARM SYSAFF=(IBM1),COPIES=2 This is a comment

The statement buffer for this statement would contain:

SYSAFF=(IBM1),COPIES=2 This is a comment
```
The statement buffer contains the comment in this case (and any trailing blanks) because there is no formal rule stating where a JECL statement ends.

4. Updating the statement buffer is only valid for parameters that have $STMTTABs in HASCRIPT.

5. Updates to the statement buffer are not passed to the converter and will not be seen by Exit 6.

6. The following indicators in the XPL can assist you in adding a card image to the current JCL statement:

   **X054LOPR**
   Current card has the last operand in the JCL statement. There can be additional continued comments after the current card.

   **X054QUOT**
   A quoted string is being continued from the current card to the next card. Pay attention if a card is being added after this card.

   **X054CCMT**
   The current card is a continued comment. Operand added to this card or after this card will not be processed.

   **X054LAST**
   This is the last card image in the JCL or JECL statement.

7. To assist you in processing the operands on a statement, you can use either of these services to parse the statement buffer passed in X054STMT:

   - The $SCAN facility can be used to parse the operands using the standard $SCAN rules for statements. This give you the flexibility of $SCAN but the parsing rules are not the same as normal JCL. See the $SCAN and $SCANTAB macros for additional information.

   - The RCARDSCN service and $STMTTAB macro can be used to parse the operands using standard JCL rules. This is the service used by JES2 input processing to parse the statement buffer. However, the RCARDSCN service only parses the operands and calls a processing routine to do all the conversions and storing of data. Conversion of data to binary to store into data areas is the responsibility of the processing routines. See the $STMTTAB macro for more information.

8. To entirely replace standard JES2 control card processing (HASPCCCS) for a particular JES2 control statement, write your routine as a replacement version of the standard HASPRCCS routine and then pass a return code of 8 back to JES2 to suppress standard processing. Note that your routine becomes responsible for duplicating any HASPRCCS function you want to retain. If you merely want to supplement standard HASPRCCS processing, you can write your exit routine to perform the additional function and then, by passing a return code of 0 or 4, direct JES2 to execute the standard HASPRCCS routine.

9. To nullify a JES2 control statement, pass a return code of 8 to JES2 without using your exit routine to perform the function requested by the statement. Note that, based on what appears in the JCL images output data set, the user is not informed that the statement was nullified.

10. To modify a JES2 control statement, also use return code 8. Place the altered statement in the area pointed to by X054JXWR and set X054XSNC to one. If input processing is successful, the user will see the original statement in the output of the JCL images file, and the altered statement. Note that if you modify a JES2 control statement; then pass a return code of 0 or 4, JES2 carries out normal input (HASPRCCS) processing. The modified version of the statement will appear on the user’s output in the JCL images file, but the
original statement will not appear unless you go directly to output phase (bypassing the converter); then, the user will see the original statement when the JCL data set is printed.

11. You also use return code 8 in processing your own installation-specific JES2 control statements. Write your exit routine to perform the function requested by the statement and then pass return code 8 to JES2 to suppress standard processing and thereby prevent JES2 from detecting the statement as "illegal."

12. Extend the JCT Control Block. You can use the $JCTX macro extension service to add, expand, locate, and delete extensions to the job control table ($JCT) control block from this exit. For example, you can use these extensions to store job-related information. Extensions that are added can be SPOOLed extensions that are available to all exits that read the JCT or local extension that are available only to input processing exits (52, 53, 54, and 50) and the SQMOD exit (51). The size of SPOOLed extensions is based on the SPOOL buffer size and is less than 3K. You can have up to 8K of local extension regardless of SPOOL buffer size.

13. To process your own installation-specific JES2 control statement subparameters, you should generally write your exit routine to replace standard HASPRCCS processing entirely. That is, write your exit routine to perform the functions requested by the standard parameters and subparameters, and those requested by any unique installation-defined subparameters on a statement. Then, from your exit, pass a return code of 8 back to JES2. Typically, because the parameters and subparameters on a JES2 control statement are interdependent, you will be limited to this method. However, if you have defined an installation-specific subparameter which can be processed independently of the rest of the control statement on which it appears, you can write your exit routine to process this subparameter alone, delete it, and pass a return code of 0 or 4 to JES2. JES2 can then process the remainder of the statement as a standard JES2 control statement.

14. When passing a return code of 12 or 16, it is also possible for your exit routine to pass an error message to JES2 for display at the operator's console. To send an error message, generate the message text in your exit routine, move it to the area pointed to by X054JXWR, and set the X054XSEM bit in X054RESP to one.

15. If you intend to use this exit to affect the JCT, your exit routine must ensure the existence of the JCT on receiving control. If the JCT has not been created when your exit routine receives control, the pointer to X054JXWR is zero. For example, when your exit routine receives control for a /*PRIORITY statement, the JCT doesn't exist yet. In this case, your routine must store any data to be placed in the JCT until JES2 creates the JCT.

16. Your exit routine does not have access to the previous control card image. You should take this into account when devising your algorithm.

17. An 80-byte work area, pointed to by X054JXWR, is available for use by your exit routine. If your routine requires additional work space, use the $GETMAIN macro to obtain storage (and the $FREMAIN macro to return it to the system when your routine has completed).

18. Exit 54 can use field JCTIPRIO to force a priority for a job subject to the limitations of the input device’s priority increment and priority limit values. When exit 54 receives control, a value of C'*' in JCTIPRIO indicates a priority has not been forced by an exit routine. If you want to force a priority in exit 54, set JCTIPRIO to a value between 0 and 15 in the low-order four bits on the field.
**Note:** Whether you can set field JCTIPRIO and the allowable values depend on the specific exit.

19. When this exit adds or modifies cards, whether the change is sent over NJE (including SPOOL offload) depends on the statement type and the setting of option flags in the $XPL or $RJCB. Modified JECL cards (original and modified card are both JECL) are not sent over NJE. By default, all other changes are sent over NJE. To limit changes to only the local node, you can set the X054RLOC in the XPL (affects the current card) or set the RJCB3LOC bit in any RJCBs that are added.

20. **Accessing $NITs**

The $NIT macro defines the characteristics of NJE nodes. The $NITs are arranged in a table that is indexed by the node number. The table of $NITs is in JES2 private storage and shadowed in a data space for use outside the JES2 address space. Installation exits can use three fields in the $NJEWORK work area to access the $NIT table. Installation exits can use these fields to access a $NIT without regard for what address space they are in.

Because these fields are in the $NJEWORK data area, you can address them using the ‘NJE’ prefix or the prefix for the device dependent work area in which the $NJEWORK is embedded. Therefore, you can address NJENITAD as JRWNITAD in the $JRW.

The following code accesses the origin node’s NIT in an NJE JOB receiver exit:

```plaintext
USING NIT,R1 Est NIT addressability
SPACE 1
$ARMODE ON,SYSTYPE=SET,INIT=CCTZEROS Enter AR mode
SPACE 1
LLGH R1,JRWRDNOD Get origin node number
MH R1,CCTNITSZ Get NIT offset
AL R1,JRWNITBL Get NIT address
LAM AR1,AR1,JRWNITAL Get NIT ALET
```

21. **Determining the device type**

Most exits need to determine the type of device that they are being called under. The $NJEWORK area has copies of $DCT fields that can help identify the device. Which method you use depends on the condition that you are testing for.

The field NJEDEVTP (that corresponds to DCTDEVTP) is a one byte flag that can be used to test for classes of devices. A test of the DCTNET bit in NJEDEVTP indicates that the exit is being called under a networking device. A compare of the byte to DCTINR indicates that the exit is being called under an internal reader. See the $DCT for the meaning of the bits in DCTDEVTP. NJEDEVTP corresponds to DCTDEVTP. This is a 3 byte value that can uniquely identify a device. This is more often used when knowing what specific device you are running under. See the $DCT for the meaning of the fields.

---

**Register contents when Exit 54 gets control**

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Eyecatcher</td>
</tr>
<tr>
<td>XPLLEVEL</td>
<td>Version level for base XPL</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit ID number</td>
</tr>
</tbody>
</table>

Exit 54: JCL and JES2 control statement scan (JES2 user environment) 325
Exit 54

XPLEXLEV  Version number for exit
X054IND    Indicator byte
  00     JCL card detected
  04     JECL card detected
X054COND   Condition byte
  X054CONT Card is a continuation (not the first
card of the JCL or JECL statement)
  X054JOBP /*JOBPARM card detected
  X054CMND /*$ command card detected
  X054SEC  This card has been passed to the
           exit previously for this job (set if
cards added before this card)
X054RESP   Response byte
  X054XSNC  Exit supplied next card in
            X054JXWR
  X054XSEM  Exit supplied error message in
            X054JXWR
  X054JCMT  Skip processing card
  X054KILL  Kill current job (queue job to
            OUTPUT processing)
  X054PURG  Purge current job
  X054RLOC  Changed or added cards are not
            sent through NJE (set RJCB3LOC
            in current RJCB)
XPLSIZE    Size of parameter list, including base section
X054CARD   80-byte card image address
X054FLGX   Pointer to exit flags (same as JRWFLAGX)
X054JXWR   80-byte exit work area address (same as
            JCTXWRK)
X054JCT    JCT address
X054JQE    Update mode JQA address
X054AREA   JRW address
X054STMT   Concatenated statement buffer. This is all the
           operands on all continuations cards for this
           statement
X054STME   End of statement+1 pointer (in buffer)
X054STML   Statement label
X054STMV   Statement verb
X054RJCP   RJCBs to add before the current JCL/JECL
           statement
X054RJCA   RJCBs to add after the current JCL/JECL
           statement
X054RJCC   RJCBs to add after the current card
X054FLG1   Statement flag byte
  X054LOPR  Last operand is on the current card
  X054QUOT  Unfinished quote at end of current card
  X054CCMT  Current card is a continued comment
Exit 54

X054LAST Last card in JCL or JECL statement

1 Address of a 3-word parameter list with the following structure:
   Word 1 (+0) address of the control statement image buffer
   Word 2 (+4) points to the exit flag byte, JRWFLAGX, in the $JRW
   Word 3 (+8) points to the JCTXWRK field in the $JCT

2-10 Not applicable
11 Address of the HCCT
12 Not applicable
13 Address of the save area
14 Return address
15 Entry address

Register contents when Exit 54 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

A return code of:

0 Tells JES2 that if any additional exit routines are associated with this exit, call the next consecutive exit routine. If there are no additional exit routines associated with this exit, perform standard input processing.

4 Tells JES2 to ignore any other exit routines associated with this exit and to perform standard input processing.

8 For JES2 control statements and JCL statements, tells JES2 not to perform standard processing and just write the statement to the JCL data set.

12 Tells JES2 to cancel the job because an illegal control statement has been detected; output (the incomplete JCL images listing) is produced.

16 Tells JES2 to purge the job because an illegal control statement has been detected; no output is produced.

Note: For all JES2 control statements preceding the JOB card (X054PREJ on), a return code higher than 4 is ignored.

Coded example

Modules HASX54A, HASX54B, and HASX54C in SYS1.SHASSAMP contains a samples of Exit 54.
Exit 55: NJE SYSOUT reception data set disposition

Function

This exit allows an installation to change the default processing (delete) for a data set which failed RACF verification upon entering this node on a TCP/IP line. In this exit, you can:

- Continue default processing and delete the data set
- Accept the data set

Environment

Task

General purpose subtask in NETSRV address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 55 in supervisor state and PSW key 0.

Recovery

Your exit routine should provide its own recovery.

Job exit mask

This exit is not subject to job exit mask suppression.

Mapping macros normally required

$HASPEQU, $HCCT, $JCT, $JCTX, $NHD, $PDDB, $XPL, $NJEWORK, $SRW

Point of processing

This exit is taken from HASCNJSR. JES2 passes control to this exit when RACF fails the verification for a SYSOUT data set received from another node on a TCP/IP line.

Programming considerations

When rerouting the data set, your exit routine should ensure the data set has the proper authority for the target node. If your routine accepts SYSOUT already rejected by RACF, there will not be an audit record for the subsequent data set create. The owner of the data set is the userid of the job that created the SYSOUT, even if that userid could not own the data on your system and RACF does not validate the assigned userid. If you are using security labels, RACF assigns a SECLABEL of SYSLOW to the data set created.

Expanding the JCT Control Block

© Copyright IBM Corp. 1988, 2009
Exit 55

You can add, expand, locate, or remove extensions to the job control table (\$JCT) control block from this exit using the \$JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see \textit{z/OS JES2 Macros}.

Related Exits

If you code Exit 55, it may also be necessary for you to code a parallel Exit 39 to provide the same function for SNA and BSC lines.

Register contents when Exit 55 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

\textbf{Field Name} \hspace{1cm} \textbf{Description} \\
\hline
XPLID      | Eyecatcher (\'XPL\') |
XPLLEVEL   | The version level of \$XPL |
XPLXITID   | The exit ID number |
X055IND    | Indicator byte |
X055COND   | Condition byte |
X055RESP   | Response byte |
X055PDDB   | PDDB address |
X055JCT    | JCT address |
X055NDH    | Data set header address |
X055AREA   | SRW address |
| 2-10      | Not applicable |
| 11        | Address of the HCCT |
| 12        | Not applicable |
| 13        | Address of the save area |
| 14        | The return address |
| 15        | The entry address |

Register contents when Exit 55 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Pointer to a parameter list with the following structure, mapped by $XPL:</td>
</tr>
</tbody>
</table>

\textbf{Field Name} \hspace{1cm} \textbf{Description} \\
\hline
X055IND    | Indicator byte. |
X055COND   | Condition byte. |
X055RESP   | Response byte. Set by exit before returning to JES2 |
X055RECV   | If you set this bit on, JES2 can receive the data set. Otherwise, processing continues and the data set is deleted. |
| 2-13      | Not applicable. |
Exit 55

A return code of:

0  Tells JES2 that if additional exit routines are associated with this exit, call the next consecutive exit routine. If no other exit routines are associated with this exit, continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

4  Tells JES2 that even if additional exit routines are associated with this exit, ignore them; continue with normal processing, which is determined by the particular exit point from which the exit routine was called.

Coded example

Module HASX55A in SYS1.SHASSAMP contains a sample of Exit 55.
Exit 56: Modifying an NJE data area before its transmission

Function

This exit allows you to change an NJE data area before transmitting a job to another node through TCP/IP NJE. (See Network Job Entry (NJE) Formats and Protocols for more information about the various NJE data areas that can be transmitted across a network.) Before transmitting the NJE job, your installation might need to add, remove or change information to one or more of the following NJE data areas:

- NJE job header
- NJE data set header
- NJE RCCS (Record Characteristics Change Section) header
- NJE job trailer

Your installation might want to:

- Remove any installation-defined sections your installation added to the NJE job when exit 56 was processing the NJE job. However, it might not be necessary to remove any installation-defined sections because installation-defined sections are ignored when they are received at other nodes.
- Add or change information, such as accounting, security or scheduling information, needed by another node in the network.
- Extract information from user fields in JES2 defined control blocks or installation defined control blocks and transfer them to the NJE data areas.

Related exits

Consider using:

- Exit 40 if you want to change the output characteristics associated with a SYSOUT data set before it prints at your node.
- Exits 2, 52, 47, or 57 to modify NJE job headers for jobs that are received for processing at your installation.
- Exit 56 to receive control for spool TCP/IP NJE lines.
- Exit 46 to receive control for SNA or BSC lines or spool offload.

Recommendations for implementing Exit 56

If you want to remove an installation-defined section from the NJE data area passed to Exit 56, you should:

- Use XPLIND to determine the type of NJE data area that JES2 passed to Exit 56 for processing.
- Issue a $NDHREM macro to remove the installation section.

Environment

Task

JES2 General purpose subtask in NETSRV address space. You must specify ENVIRON=USER on the $MODULE macro.
Exit 56

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 56 in supervisor state and PSW key 0.

Recovery

Your installation should provide its own recovery routine.

Job exit mask

Exit 56 is subject to suppression. Your installation can either implement Exit 2 or Exit 52 to set the 56th bit in the job exit suppression mask (JCTXMASK) or disable the exit in the JES2 initialization stream.

Mapping macros normally required

$HASPEQU, $PDDB, $SCR, $XPL, $NHD, $HCCT, $JQE, $JCT, $JCTX, $JOE, $NJEWORK, $JTW, $STW

Point of processing

JES2 invokes Exit 56 before transmitting a job while transmitting an NJE job across a TCP/IP line. Before invoking Exit 56, JES2:

- Builds the NJE data area in a 32K buffer
- Removes any JES2-specific sections from the NJE data area if JES2 is transmitting the NJE data area to another node in the network. The following NJE data areas contain a JES2 section:
  - Job Header
  - Job Trailer
- Initializes the $XPL parameter and invokes Exit 56.
- After returning from Exit 56, JES2 examines the response byte (XPLRESP) in the $XPL parameter list. If in Exit 56 you set XPLRESP to:
  - X056TERM, it indicates an error occurred. JES2 terminates the transmission of the NJE data area, and places the job in hold.
  - X056BYP, JES2 continues processing the remainder of the NJE job because Exit 56 transmitted the buffer that contained the NJE data area.

If XPLRESP has not been set, JES2 transmits the NJE data area.

Programming considerations

The following are programming considerations for Exit 56:

- Locating the JCT Control Block Extensions
  You can locate extensions to the job control table ($JCT) control block from this exit using the $JCTXGET macro. For example, you can use these extensions to retrieve job-related information from the $JCTX control block to ship across the network in $NHD macro sections. For more information, see z/OS JES2 Macros.

- Accessing $NITs
  The $NIT macro defines the characteristics of NJE nodes. The $NITs are arranged in a table that is indexed by the node number. The table of $NITs is in JES2 private storage and shadowed in a data space for use outside the JES2 address space. Installation exits can use three fields in the $NJEWORK work
area to access the $NIT table. Installation exits can use these fields to access a $NIT without regard for what address space they are in.

Because these fields are in the $NJEWORK data area, you can address them using the ‘NJ’ prefix or the prefix for the device dependent work area in which the $NJEWORK is embedded. Therefore, you can address NJENITAD as JRWNITAD in the $JR, JTWNITAD in the $NJT, SRWNITAD in the $SRW, and STWNITAD in the $STW.

The following code accesses the origin node’s NIT in an NJE JOB receiver exit:

```
USING
   Est NIT addressability
SPACE 1
$ARMODE ON,SYSTYPE=SET,INIT=CCTZEROS Enter AR mode
SPACE 1
LLGH R1,JRWDNOD Get origin node number
MH R1,CCTNITSZ Get NIT offset
AL R1,JRNITBL Get NIT address
LAM AR1,AR1,JRNITAL Get NIT ALET
```

**Determining the device type**

Most exits need to determine the type of device that they are being called under. The $NJEWORK area has copies of $DCT fields that can help identify the device. Which method you use depends on the condition that you are testing for.

The field NJEDEVTP (that corresponds to DCTDEVTP) is a one byte flag that can be used to test for classes of devices. A test of the DCTNET bit in NJEDEVTP indicates that the exit is being called under a networking device. A compare of the byte to DCTINR indicates that the exit is being called under an internal reader. See the $DCT for the meaning of the bits in DCTDEVTP.

NJEDEVID corresponds to DCTDEVID. This is a 3 byte value that can uniquely identify a device. This is more often used when knowing what specific device you are running under. See the $DCT for the meaning of the fields.

### Register contents when Exit 56 gets control

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Parameter List Address having the following structure:</td>
</tr>
</tbody>
</table>

- **Field Name**
  - **XPLID**: Eyecatcher ('$XPL')
  - **X056VERN**: Parameter list version
  - **XPLXITID**: Exit identifier
  - **XPLEXLEV**: Version level of the exit
  - **X056IND**: Indicates the type of NJE data area JES2 passed to Exit 56 for processing. A value of:
    - **X056HDR**: Indicates an NJE job header was passed to Exit 56 for processing.
    - **X056TRL**: Indicates an NJE job trailer was passed to Exit 56 for processing.
    - **X056DSH**: Indicates an NJE data set header was passed to Exit 56 for processing.
Exit 56

X056RCCS
Indicates an NJE RCCS header was passed to Exit 56 for processing.

X056COND
Condition byte

X056R1ST
Indicates that this RCCS header precedes the first data record.

X056RESP
Response byte.

X056HADR
Contains the address of the NJE data area.

(Reserved field)
This field is reserved for Exit 56 to keep the same offsets of the XPL mapping as Exit 46. This value is always zero for Exit 56.

X056JQE
Address of read mode JQA.

X056JCT
Contains the address of the $JCT.

X056PDDB
Contains the address of the $PDDB if Exit 56 is processing an NJE data set header. If Exit 56 is processing an NJE job header or trailer, a 0 is passed as the address.

X056JOA
Contains the address of the artificial JOE (JOA) if Exit 56 is processing an NJE data set header. If Exit 56 is processing an NJE job header or trailer, a 0 is passed as the address.

Note: If the exit must update JOE fields, it should obtain and return an update mode JOA. For more information, see “Checkpoint control blocks for JOEs” on page 384.

X056AREA
Contains the address of the NJEWORK area (JTW or STW) for the transmitter device sending the header.

X056SIZE
Indicates the length of the $XPL parameter list for Exit 56.

2-10
Not applicable

11
Address of the HCCT

12
Not applicable

13
Address of the save area

14
The return address

15
Entry point address of Exit 56

Register contents when Exit 56 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list, which has the following structure:</td>
</tr>
</tbody>
</table>

XPLID  Eye-catcher for the $XPL

X056VERN  Indicates the version number of Exit 56

XPLXITID  Exit identifier

XPLEXLEV  Version level of the exit
**Exit 56**

**X056IND**
Indicator byte

**X056COND**
Condition byte

**X056RESP**
Indicates the processing Exit 56 determined JES2 should perform after processing the NJE data area. A value of:

**X056TERM**
Indicates Exit 56 determined the NJE data area should not be transmitted. JES2 will discard the remainder of the NJE job.

**X056BYP**
Indicates JES2 should not transmit the NJE data area. JES2 will continue to process the remainder of the NJE job.

**X056SIZE**
Indicates the length of the $XPL parameter list for Exit 56.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-13</td>
<td>Not applicable to Exit 56</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Exit effector return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the job.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates JES2 should continue processing the job, but ignore any additional exits associated with Exit 56.</td>
</tr>
</tbody>
</table>

**Coded example**

Module HASX56A in SYS1.SHASSAMP contains a sample of Exit 56. Module HASXJECL in SYS1.SHASSAMP also contains an example.
Exit 57: Modifying an NJE data area before receiving the rest of the NJE job

Function

This exit allows you to:

- Examine and change an NJE data area before receiving the rest of the NJE job from another node through TCP/IP NJE.
- Add, expand, locate, or remove an extension to the $JCT control block where accounting information can be stored.

Before receiving an NJE job, your installation might need to add, remove or change information to one or more of the NJE data areas below. See [Network Job Entry (NJE) Formats and Protocols] for more information about the various NJE data areas that can be transmitted across a network.

- NJE job header
- NJE data set header
- NJE RCCS (Record Characteristics Change Section) header
- NJE job trailer

Your installation might want to:

- Remove any installation-defined sections your installation added to the NJE job when exit 56 was processing the NJE job.
- Add or change information, such as accounting or security information, needed by another node in the network.
- Extract information from the NJE data areas and transfer them to user fields in JES2 defined control blocks or installation defined control blocks.

Related exits

If you want to change the output characteristics associated with a SYSOUT data set, consider using exit 40. Exit 57 only receives control for TCP/IP NJE. If you code exit 57, you may also need a Exit 47 to handle jobs received on SNA or BSC lines or through spool offload.

Environment

Task

General purpose subtask in NETSRV address space. You must specify ENVIRON=USER on the $MODULE macro.

AMODE/RMODE requirements

RMODE ANY, AMODE 31

Supervisor/problem program

JES2 places Exit 57 in supervisor state and PSW key 0.

Recovery

Your installation should provide its own recovery routine.
Exit 57

Job exit mask

Exit 57 is subject to suppression. The installation can suppress the exit either by implementing exit 2 to set the 57th bit in the job exit suppression mask (JCTXMASK) or by indicating the exit is disabled in the JES2 initialization stream.

Mapping macros normally required

$HASPEQU, $PDDB, $SCR, $XPL, $NHD, $HCCT, $JQE, $JCT, $JCTX, $JOE, $NJEWORK, $JRW, $SRW

Point of processing

JES2 invokes Exit 57 before receiving a job while performing receiving an NJE job across a TCP/IP line. Before invoking Exit 57 JES2:
- Allocates a dummy $JCT and $JQE. JES2 initializes these data areas with minimal information.
- Receives the NJE data area and invokes Exit 57 to perform installation-specific processing.
- After returning from Exit 57, JES2 determines if exit 57 indicated whether the NJE data area should be received. If exit 57 indicated the NJE data area should not be received, JES2 places the NJE job in hold on the transmitting node. Otherwise, JES2 continues to process the NJE job. You cannot use this exit to update IBM-defined JCT or JQE fields in the dummy JCT and dummy JQE, respectively. You can, however, update user-defined fields (such as JCTUSERx) or any $JCTX extensions you have created. JES2 propagates changes to 'user' fields to the $JCT and $JQE.

Programming considerations

The following are programming considerations for Exit 57:
- If the exit is being invoked for a job header, the JOE address passed points to a dummy JQE (as indicated by X057BJQE). See "Checkpoint control blocks" on page 382 for more information.
- **Extending the JCT Control Block**
  You can add, expand, locate, or remove extensions to the job control table ($JCT) control block from this exit using the $JCTX macro extension service. For example, you can use these extensions to store job-related information. For more information, see z/OS JES2 Macros.
- **Accessing $NITs**
  The $NIT macro defines the characteristics of NJE nodes. The $NITs are arranged in a table that is indexed by the node number. The table of $NITs is in JES2 private storage and shadowed in a data space for use outside the JES2 address space. Installation exits can use three fields in the $NJEWORK work area to access the $NIT table. Installation exits can use these fields to access a $NIT without regard for what address space they are in.
  Because these fields are in the $NJEWORK data area, you can address them using the 'NJ'E prefix or the prefix for the device dependent work area in which the $NJEWORK is embedded. Therefore, you can address NJENITAD as JRWNITAD in the $JRW, JTWNITAD in the $NJT, SRWNITAD in the $SRW, and STWNITAD in the $STW.
  The following code accesses the origin node's NIT in an NJE JOB receiver exit:
USING NIT,R1  Est NIT addressability
SPACE 1
$ARMODE ON,SYSTYPE=SET,INIT=CCTZEROS Enter AR mode
SPACE 1
LLGH R1, JRWRDNOD  Get origin node number
MH R1, CCTNITSZ  Get NIT offset
AL R1, JRWNITBL  Get NIT address
LAM AR1, AR1, JRWNITAL  Get NIT ALET

**Determining the device type**
Most exits need to determine the type of device that they are being called under. The $NJEWORK area has copies of $DCT fields that can help identify the device. Which method you use depends on the condition that you are testing for.

The field NJEDEVTP (that corresponds to DCTDEVTP) is a one byte flag that can be used to test for classes of devices. A test of the DCTNET bit in NJEDEVTP indicates that the exit is being called under a networking device. A compare of the byte to DCTINR indicates that the exit is being called under an internal reader. See the $DCT for the meaning of the bits in DCTDEVTP.

NJDEVID corresponds to DCTDEVID. This is a 3 byte value that can uniquely identify a device. This is more often used when knowing what specific device you are running under. See the $DCT for the meaning of the fields.

---

**Register contents when Exit 57 gets control**

The contents of the registers on entry to this exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 57</td>
</tr>
<tr>
<td>1</td>
<td>Parameter List Address having the following structure:</td>
</tr>
</tbody>
</table>

**Field Name**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPLID</td>
<td>Eyecatcher ('$XPL')</td>
</tr>
<tr>
<td>X057VERN</td>
<td>Indicates the version number of Exit 57</td>
</tr>
<tr>
<td>XPLXITID</td>
<td>Exit identifier - 57</td>
</tr>
<tr>
<td>XPLEXLEV</td>
<td>Version level of the exit</td>
</tr>
<tr>
<td>X057IND</td>
<td>Indicates the type of NJE data area JES2 passed to Exit 57 for processing. A value of:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X057HDR</td>
<td>Indicates an NJE job header was passed to Exit 57 for processing.</td>
</tr>
<tr>
<td>X057TRL</td>
<td>Indicates an NJE job trailer was passed to Exit 57 for processing.</td>
</tr>
<tr>
<td>X057DSH</td>
<td>Indicates an NJE data set header was passed to Exit 57 for processing.</td>
</tr>
<tr>
<td>X057RCCS</td>
<td>Indicates an NJE RCCS header was passed to Exit 57 for processing.</td>
</tr>
<tr>
<td>X057BJQE</td>
<td>Indicates the JQE address in field X057JQE points to a working copy of the JQE that has not yet been added to the job queue. The working copy should not be used in services that expect the address of a real JQE.</td>
</tr>
</tbody>
</table>
Exit 57

**X057COND**  Condition byte.
**X057RESP**  Response byte.
**X057HADR**  Contains the address of the NJE data area.

*(Reserved field)*

This field is reserved for Exit 57 to keep the same offsets of the XPL mapping as Exit 47. This value is always zero for Exit 57.

**X057JQE**  Contains the address of an update mode JQA.
**X057JCT**  Contains the address of the $JCT.
**X057PDDB**  Contains the address of the $PDDB if Exit 57 is processing an NJE data set header. If Exit 57 is processing an NJE job header or trailer, a 0 is passed as the address.

**X057AREA**  Contains the address of the NJEWORK area (JRW or SRW) for the receiver.
**X057SIZE**  Indicates the length of the $XPL parameter list for Exit 57.

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-10</td>
<td>Not applicable</td>
</tr>
<tr>
<td>11</td>
<td>Address of the HCCT</td>
</tr>
<tr>
<td>12</td>
<td>Not applicable</td>
</tr>
<tr>
<td>13</td>
<td>Address of the save area</td>
</tr>
<tr>
<td>14</td>
<td>The return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address of Exit 57</td>
</tr>
</tbody>
</table>

### Register contents when Exit 57 passes control back to JES2

Upon return from this exit, the register contents must be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Not applicable to Exit 57</td>
</tr>
<tr>
<td>1</td>
<td>Address of the $XPL parameter list which has the following structure:</td>
</tr>
</tbody>
</table>

**X057IND**  Condition byte

**X057COND**  Response byte

**X057RESP**  Indicates the processing Exit 57 determined JES2 should perform after processing the NJE data area. A value of:

**X057TERM**  Indicates Exit 57 determined the NJE data area should not be received. JES2 will stop processing the rest of the NJE job.

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-13</td>
<td>Not applicable to Exit 57</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Exit effector return code</td>
</tr>
</tbody>
</table>

A return code of:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates JES2 should continue processing the job.</td>
</tr>
<tr>
<td>4</td>
<td>Indicates JES2 should continue processing the job, but ignore any additional exits associated with this exit.</td>
</tr>
</tbody>
</table>
Coded example

Module HASX57A in SYS1.SHASSAMP contains a sample of Exit 57. Module HASXJECL in SYS1.SHASSAMP also contains an example.
Chapter 4 - JES2 exit migration considerations

This chapter provides more details on a subset of the migration actions required for JES2. For a complete list of the migration actions from one JES2 release to another JES2 release, see z/OS Migration. The migration details in this chapter are presented in the order in which they were introduced into a z/OS release. See z/OS Migration for more information about when the migration actions are required.

JES2 z/OS V1R11 migration details

A new checkpoint activation level, which is called z11, is available for JES2 release V1R11. The current checkpoint level is z2. JES2 needs certain conditions to activate to the z11 checkpoint level.

JES2 z/OS V1R11 checkpoint activation

Use the $ACTIVATE command to activate to the z11 checkpoint level. The $ACTIVATE, LEVEL=z11 command expands the JES2 checkpoint to support functions that are enabled with the z11 checkpoint level. JES2 will reject the $ACTIVATE command if certain conditions are not met. For information about a complete list of new functions that are enabled by z11 activation, see $ACTIVATE command in z/OS JES2 Commands.

The $D ACTIVATE command provides an exhaustive list of reasons that block checkpoint activation to the z11 checkpoint level. It is suggested that you use the $D ACTIVATE command before attempting an activation to the z11 checkpoint level.

JES2 z/OS V1R11 exits and macros

Before activating to JES2 release V1R11, you should meet certain conditions if your installation is using the following JES2 exits or macros:

- If you reference JOE fields in any of your exits, check the $DOGJOE macro. This macro enables creation of read and update mode artificial JOEs that are called JOAs. For information about $DOGJOE macro, see z/OS JES2 Macros. For a description of JOAs, see “Checkpoint control blocks” on page 382.

- If you use JES2 Exit 1, Exit 15, Exit 38, Exit 46 or Exit 56, your code might need to be updated. Before JES2 release V1R11, real work JOE and characteristics JOE were passed to these various exits. Starting with JES2 release V1R11, an artificial JOE will be passed to each of these exits. For more information about these exits, see “Chapter 3 - IBM-defined exits” on page 59.

- If you use JES2 macros $#ADD, $#ALCHK, $#BLD and $#BUSY, your code might need to be updated. In many cases the interface has changed to require that a JOA be passed into the macro versus a work JOE or work/characteristics JOE combination. In the case of $#BUSY and $#ALCHK, additional rules must be followed. For information about the requirements of the macros, see z/OS JES2 Macros.

- The size of the JOX is changed. If you use the $DOGJQE service, the code should not be impacted.

- JOEs, JOEs and BERTs have new size limits:
  - JOEs = 400,000
  - JOEs = 1,000,000
  - BERTs = 1,000,000
The $#JOE macro returns a real JOE or a read mode JOA. Before JES2 release V1R11, this macro only returned a real JOE. The default for $#JOE are changed to return a read mode JOA. If read mode JOA is used and an early exit is made from $#JOE processing loop, make sure that JOA is properly released by a call to $DOGJOE service ACTION=RETURN.

Before JES2 V1R11, exit 7 could determine which record was being read by using the field CBMTTR. Starting with JES2 release V1R11, CBIO uses MQTR to address a record on spool. If you have exit routines that examine field CBMTTR, change them to examine field CBMQTR.

JES2 is now an exploiter of 64-bit common storage to store information for devices and jobs.

JES2 z/OS V1R7 exit migration details

As of z/OS V1R7 and later, JES2 has moved some processing out of the JES2 main task and address space into other address spaces. This change improves performance by reducing the CPU load on the JES2 main task and also provides failure isolation. One of the effects of this change is that main task exits in the moved processing is no longer called. A new set of exits gives installations and vendors control at the same point of processing as the main task exits. However, these new exits do not run in the JES2 main task or in the JES2 address space and no longer have access to the same data areas as the previous exits.

As a result of these changes, installations that use any of the affected exits need to examine exits to determine if they must write code for the new exits. To simplify the migration, JES2 provides additional functions in the new exits and in some existing exits. This section summarizes the improvements made to the exits and explains how to locate data that was available to the original main task exits.

Consider the following approaches when you evaluate your existing exits:

- First, determine what is needed to migrate existing exits to the new environments with minimal alteration of the logic of the exit code. In most cases, this can be done.
- Second, determine what the exit logic is doing and to evaluate what is needed to accomplish the same function in the new exits. This approach often results in much simpler exit logic that is less error prone. However, it requires a larger upfront design effort.

Data structure changes

To move the internal reader and NJE over TCP/IP processing out of the JES2 address space, a number of fundamental changes have been made to the $PCE data areas for all job input and NJE processing. The changes include that most of the JES2 code that performs job input and NJE processing is moved to common storage modules and is altered to run outside the JES2 address space. The move implies that the code has to function without access to the data areas that are available in the JES2 address space (such as the $HCT and the $PCE). Also, it requires changes to the following JES2 data areas:

In the case of the NJE related $PCE data areas ($RDRWORK, $NSRWORK, $NJTWORK and $NSTWORK), each was split into the following three separate data areas:

- A $PCE work area (with the same name). The $PCE mapping contains the fields only needed in the JES2 main task.
- A new common NJE/input processing work area ($NJEWORK). The common $NJEWORK mapping contains not only fields common to all 4 original SPCE work areas but also copies of fields in the $DCT. Some fields in $NJEWORK are copies of fields that exist in the SPCE. This is to simplify common code that may or may not be running in the JES2 address space.
- A new device type specific data area. The device type specific work areas contain fields needed by processing that may not be running in the JES2 address space. In addition, they contain fields only needed by code running outside the JES2 address space.

$NJEWORK data area: The $NJEWORK data area is placed at the front of each of the device work areas ($JRW, $SRW, $JTW, and $STW). It can be accessed either directly using a DSECT name of NJEWORK and a label prefix of NJE or it can be accessed as part of the device work area with the prefix for that work area. For example, you can access the device name field in $NJEWORK as NJEDEVN in DSECT NJEWORK or SRWDEVN in DSECT SRW. This makes accessing the needed data easier, but can make it difficult to locate where a field is defined.

Device dependent work area: The device dependent work areas are all embedded in the corresponding PCE work areas. For example, $NSRWORK includes space for the $SRW device work area. If you have a PCE address with a USING established and need to address fields in the device dependent work area, you can use a dependent USING. For the $NSRWORK mapping, a USING SRW,SRWCWKAR establishes addressability to the $SRW device dependent area and the $NJEWORK area at the front of the $SRW mapping. For some exits, a USING similar to this might be all that is needed to access the PCE fields needed by the exit.

Exits related to NJE: Exits related to NJE and Input processing (exits 2, 3, 4, 20, 46, 47, 50, 52, 53, 54, 56, and 57) are all passed $XPLs that have a pointer to the appropriate $JRW, $SRW, $JTW, or $STW work area. Most exits should be able to perform their function using the device dependent area and data passed in the $XPL.

$SRW, $JTW, and $STW: For the $SRW, $JTW, and the $STW, most of the fields that an exit might want to access have been moved from the $PCE without renaming the fields. However, the fields in the $JRW have mostly been re-named. Changes to NJE/TCP and internal reader have required significant changes to job input processing. Because fields have changed and data areas have been replaced, fields from the $RDRWORK PCE work area with a prefix of RDW in releases before z/OS V1R7 are now in $JRW with a prefix of JRW for z/OS V1R7 and later releases. Table 12 shows $PCE names, their related NJE work area, and their related device work area.

### Table 12. $PCE names and their related data areas

<table>
<thead>
<tr>
<th>$PCE work area</th>
<th>NJE work area</th>
<th>Device work area</th>
</tr>
</thead>
<tbody>
<tr>
<td>$RDRWORK</td>
<td>$NJEWORK</td>
<td>$JRW</td>
</tr>
<tr>
<td>$NSRWORK</td>
<td>$NJEWORK</td>
<td>$SRW</td>
</tr>
<tr>
<td>$NJTWORK</td>
<td>$NJEWORK</td>
<td>$JTW</td>
</tr>
<tr>
<td>$NSTWORK</td>
<td>$NJEWORK</td>
<td>$STW</td>
</tr>
</tbody>
</table>
Job input processing

Input processing is the process of taking a JCL stream with embedded data and placing it on SPOOL in a format that can be processed by later job phases. During input processing, the JCL is scanned for JECL (Job Entry Control Language cards) and JCL keywords needed by JES2 (such as job class). In addition, the embedded data cards (instream data) are separated from the JCL cards and stored as separate data sets. Input processing also performs the initial security checks for the submitted job and obtains a security token to represent the job.

In releases before z/OS V1R7, all input processing occurs in the JES2 address space under the JES2 main task. In z/OS V1R7, input processing for NJE over TCP/IP job receivers and normal JES2 internal readers (batch, STC and TSU internal readers) have been moved from the JES2 address space. For NJE over TCP/IP, the processing occurs in the NETSERV address space. For the internal reader (including the TSO SUBMIT command), the processing occurs in the submitting address space (the address space that allocated the internal reader).

Table 13 summarizes the exits and the changes that have occurred as a result of changes in job processing for V1R7 and later releases:

<table>
<thead>
<tr>
<th>Affected exit</th>
<th>Description</th>
<th>Processing before z/OS V1R7</th>
<th>Processing changes for z/OS V1R7 and later releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit 2</td>
<td>JOB card scan exit.</td>
<td>Input register 0 is a card continuation indicator.</td>
<td>The interface to this exit includes a $XPL data area to which input register 0 points. Installations need to code an exit 52 for jobs submitted using internal readers (including TSO SUBMIT) and NJE/TCP job receivers.</td>
</tr>
<tr>
<td>Exit 3</td>
<td>Accounting string exit.</td>
<td>Register 0 is the length of the accounting string.</td>
<td>The interface to this exit includes a $XPL data area to which input register 0 points. Installations need to code an exit 53 for jobs submitted using internal readers (including TSO SUBMIT) and NJE/TCP job receivers.</td>
</tr>
<tr>
<td>Exit 4</td>
<td>JCL/JECL card scan exit.</td>
<td>Register 0 is a card type indicator.</td>
<td>The interface to this exit includes a $XPL data area to which input register 0 points. Installations need to code an exit 54 for jobs submitted using internal readers (including TSO SUBMIT) and NJE/TCP job receivers.</td>
</tr>
<tr>
<td>Exit 7</td>
<td>Main task control block I/O exit.</td>
<td>Exit 7 is called for control blocks written by internal reader or NJE/TCP job receivers.</td>
<td>Exit 7 is no longer called for control blocks written by internal reader or NJE/TCP job receivers. Instead, exit 8 is called.</td>
</tr>
<tr>
<td>Exit 20</td>
<td>The job end of input processing exit.</td>
<td>$XPL supports limited functions.</td>
<td>$XPL supports additional functions. Installations need to code an exit 50 for jobs submitted using internal readers (including TSO SUBMIT) and NJE/TCP job receivers.</td>
</tr>
</tbody>
</table>
Table 13. Exits that are affected by changes to input processing (continued)

<table>
<thead>
<tr>
<th>Affected exit</th>
<th>Description</th>
<th>Processing before z/OS V1R7</th>
<th>Processing changes for z/OS V1R7 and later releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exits 36 &amp; 37</td>
<td>Pre-SAF and post-SAF exits.</td>
<td>The calls to these exits for internal readers and NJE/TCP job receivers were made in a subtask in the JES2 address space.</td>
<td>The calls to these exits for internal readers and NJE/TCP job receivers are called from outside the JES2 address space.</td>
</tr>
</tbody>
</table>

Internal readers

Internal reader processing has been moved from the JES2 address space to the address space that allocated the internal reader. As a result of this change, a number of data areas are deleted for z/OS V1R7 and later releases, including the internal reader $DCT (the RID) and the internal reader $PCE. They are replaced in z/OS V1R7 and later releases with two new data areas, the $IRE in CSA and the $IRWD in the private storage of the owner of the internal reader.

$IRE: You can use the $IRE to track processing occurring in an internal reader. Its primary use is for the $D RIDI command and cleanup processing at job termination and address space end of memory. $IREs are created when the internal reader is allocated and deleted when the internal reader is unallocated.

$IRWD: The $IRWD is the main processing area for internal readers. It contains the $JRW used by the internal reader and data areas that are unique to internal reader processing. Fields unique to internal reader processing are located after the $JRW. You can access these fields if an exit has an internal reader $JRW address and establishes the following USING:

```
USING RIDCWKAR,JRW Est. IRWD addressability
```

NJE processing

NJE processing includes four types of NJE devices, two transmitters that send data to other nodes and two receivers that accept data from other nodes. Therefore, there are two types of transmitters and receivers, one for SYSOUT streams (SPOOLed output data sets) and one for JOBs (batch jobs that have not executed).

There are also a number of transport mechanisms that can be used for NJE processing as follows:

- BSC network
- SNA network
- TCP/IP, which becomes available in z/OS V1R7
- SPOOL offload and reload transport to send NJE streams to DASD or tape data sets and the route transmitter and receiver to redo input processing for pre-execution batch jobs that are rerouted to execute on the local node.

z/OS V1R7 JES2 and later releases uses a NETSERV address space to implement TCP/IP transport mechanism. The NETSERV address space is called jesxSnnn, where jesx is the owning JES2 address space name and nnn is the corresponding NETSERVnnn statement. The NETSERV address space is responsible for all communications with TCP/IP and all SPOOL I/O needed to transmit and receive jobs. The JES2 address space only selects work for the transmitters and creates the job and output queue data structures (JQEa and JOEs) for the receivers. JES2 can also display the current status of the NJE devices.
Changes have also occurred to exits in NJE processing that get control for all device types and transport mechanisms. In releases before z/OS V1R7, these exits get control in the JES2 address space running under the JES2 main task. With the new TCP/IP transport mechanism, these exits get control in the new NETSERV address space. In z/OS V1R7 and later releases, these exits are affected by changes to the PCE work areas mentioned earlier in the “Data structure changes” on page 346 section and various functional changes.

NJE transmitters and receivers all have $DCTs and $PCEs that contain device dependent areas in the JES2 address space, regardless of the transport mechanism used; however, for NJE/TCP, JES2 does not use the device dependent area in the $PCEs. Instead, it uses the device dependent areas in the NETSERV address space. See Figure 11 and Figure 12 on page 351 for more details.

$DCT and $PCE data structure for a set of NJE devices using a traditional (non-TCP/IP) transport.

Control Blocks in JES2 Address Space

Figure 11. $DCT and $PCE data structure for a set of NJE devices using a traditional (non-TCP/IP) transport
There are two basic NJE transmitters:

- JOB transmitters
- SYSOUT transmitters

The transmitters take data from SPOOL, build or update NJE headers, and send all this data to other nodes. For z/OS V1R7 and later releases, the exits associated with the transmitters are only affected by the changes to the data structures and the NETSERV address space. Table 14 on page 352 lists the NJE transmitter exits affected by changes made in z/OS V1R7.

Figure 12. $DCT and $PCE data structure for a set of NJE devices using a TCP/IP transport.
Table 14. NJE transmitter exits affected by changes made in z/OS V1R7

<table>
<thead>
<tr>
<th>Affected Exit</th>
<th>Description</th>
<th>Processing before z/OS V1R7</th>
<th>Processing changes for z/OS V1R7 and later releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit 7</td>
<td>Main task control block I/O exit.</td>
<td>Exit 7 is called for control blocks read by NJE/TCP transmitters.</td>
<td>Exit 7 is no longer called for control blocks read by NJE/TCP transmitters. Instead, exit 8 is called.</td>
</tr>
<tr>
<td>Exits 36 &amp; 37</td>
<td>Pre-SAF and post-SAF exits.</td>
<td>The calls to these exits for NJE/TCP transmitters are made in a subtask in the JES2 address space.</td>
<td>The calls to these exits for NJE/TCP transmitters are called from outside the JES2 address space.</td>
</tr>
<tr>
<td>Exit 46</td>
<td>NJE header transmission exit.</td>
<td>Installations do not need to code an exit 56 for NJE/TCP transmitters.</td>
<td>Installations might need to code an exit 56 for NJE/TCP transmitters.</td>
</tr>
</tbody>
</table>

NJE receivers
There are two types of NJE receivers:

- JOB receivers
- SYSOUT receivers

The receivers place data received on SPOOL and convert the NJE headers that are received into JES2 control blocks. The JOB receiver performs normal job input processing in addition to the NJE functions. The effect of job input processing changes on exits is described in “Job input processing” on page 348. In addition, there are changes related to NJE on all receivers and changes specifically related to SYSOUT receivers.

Table 15. NJE Receiver Exits Affected by Changes in z/OS V1R7

<table>
<thead>
<tr>
<th>Affected Exit</th>
<th>Description</th>
<th>Processing before z/OS V1R7</th>
<th>Processing changes for z/OS V1R7 and later releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit 7</td>
<td>Main task control block I/O exit.</td>
<td>Exit 7 is called for control blocks written by NJE/TCP receivers.</td>
<td>Exit 7 is no longer called for control blocks written by NJE/TCP receivers. Instead, exit 8 is called.</td>
</tr>
<tr>
<td>Exit 13</td>
<td>TSO/E NETMAIL notify exit.</td>
<td>Exit 13 only gets control for SYSOUT receivers. Exit 13 controls whether or not the $HASP548 message is received. This exit has been deleted. In some cases, exit 13 is used as a way to process a data set header.</td>
<td>There is an existing external NJEDEF MAILMSG=YES/NO can control the issuing of the message $HASP548. If your exit did more than just setting a return code, you might need an exit 40 to implement those functions. You can use exits 47 and 57 to process a data set header instead of exit 13.</td>
</tr>
<tr>
<td>Exits 36 &amp; 37</td>
<td>Pre-SAF and post-SAF exits.</td>
<td>The calls to these exits for NJE/TCP transmitters are made in a subtask in the JES2 address space.</td>
<td>The calls to these exits for NJE/TCP transmitters are called from outside the JES2 address space.</td>
</tr>
</tbody>
</table>
Table 15. NJE Receiver Exits Affected by Changes in z/OS V1R7 (continued)

<table>
<thead>
<tr>
<th>Affected Exit</th>
<th>Description</th>
<th>Processing before z/OS V1R7</th>
<th>Processing changes for z/OS V1R7 and later releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit 39</td>
<td>SAF SYSOUT rejection exit.</td>
<td>This exit gets control only for SYSOUT receivers when the SAF call made indicates that this data set can never be received because of security restrictions.</td>
<td>Installations might need to code an exit 55 for NJE/TCP receivers.</td>
</tr>
<tr>
<td>Exit 47</td>
<td>NJE header reception exit.</td>
<td>Installations do not need to code an exit 57 for NJE/TCP receivers.</td>
<td>Installations might need to code an exit 57 for NJE/TCP receivers.</td>
</tr>
</tbody>
</table>

Techniques for multi-environment exit points

A number of the new exits in z/OS V1R7 are user environment pairs for existing main task exits. If your installation has implemented a main task version of the exit, you might need to duplicate the function of the old main task exit in the new user environment exit. You can use different techniques for these changes, each with its own advantages and disadvantages.

Decide whether you need the function that the exit provides. In many cases, exits have been carried forward from one release to the next without considering whether the function provided by the exit is available in JES2 or whether the function is even important in the current environment. For example, if your exit has code to set defaults for jobs that arrive through RJE and no longer use RJE, that function could be deleted.

Before deciding whether to code both the main task exit and the new user environment exit, you need to consider the following:

- If the function of a JOB input exit only applies to internal readers, you can move this function to the new exits because internal readers only call the user environment exits.
- If the function of a JOB input exit only applies to RJE devices, you do not need the function in the user environment exits because RJE devices only use main task exits.
- Many installations use exit 7 to get control at the absolute end-of-input processing or the end of NJE job or SYSOUT reception. In z/OS V1R7 and later releases, consider using the new main task exit 51 ($QMOD phase change) instead of coding an exit 8. Also, consider using exit 51 to track a job’s progress through various JES2 phases instead of exit 7. Furthermore, understanding the point in processing where an exit 8 is called is not as simple as determining where in processing an exit 7 call was made.

When you have a function that is required in both a main task and a user environment exit, you need to decide how to best package the code. Information about how to write a multiple environment exits is documented in the appendix F of z/OS JES2 Macros.

Specific exit details

This section provides details on each exit point that has significantly changed in the z/OS V1R7 of JES2.
Enhancements to exits 2, 4, 52, and 54
These exits are called for JCL and JECL cards received during job input processing. Exits 2 and 52 are called for the JOB card and its continuations. Exits 4 and 54 are called for all other JCL and JECL cards. Exits 2, 4, 52, and 54 are grouped together because their interfaces are practically identical. The exceptions are some unique input conditions set in exit 4 or 54, the ability to set a default job class in exit 2 or 52, and a slight difference in return codes.

Exits 2 and 4 are called for jobs submitted through card readers, RJE, SNA and BSC NJE, and SPOOL reload. Exits 52 and 54 are called for jobs submitted through internal readers (including TSO SUBMIT) and TCP/IP NJE.

Changes to register 0: Register 0 on entry to exit 2 and 4 has been changed to the address of a $XPL. In previous releases, register 0 was an indicator. This indicator has been moved to the $XPL. For exit 2, register 0 indicated that this was initial job card (0) or a job card continuation (4). This information is now in X002CONT. For exit 4, register 0 indicated that this was a JECL card (0) or a JCL card (4). This information is now in X004JCL and X004JECL.

Changes to register 1: Register 1 on entry still points to a 3 word parameter list. This list is actually within the $XPL passed in register 0. This parameter list can be used in the same way it was in previous releases. However, it is recommended that the $XPL be used instead.

$XPL mapping: The $XPL maps the fields for exits 2, 4, 52, and 54 using prefixes of X002, X004, X052 and X054. The $XPL mapping for exit 2 and 52 are identical. Exit routines can choose to use either the X002 or X052 fields when processing in exit 2 or 52. The same is true of exits 4 and 54. Furthermore, with the exception of the default job class in exit 2/52, the $XPL for exits 2/52 and 4/54 are also identical. As a result, any of the X002, X004, X052, or X054 fields can be used in any of the 4 exits. Examples will generally use the lower number prefixes for fields in the $XPL.

RDWFLAGX changes: An existing exit may use RDWFLAGX to indicate that the exit has supplied a next card (RDWXXSNC) or that an error message has been provided (RDWXXSEM). These functions are available using bits defined in the $XPL. X0xxXSNC indicates that JCTXWRK (pointed to by X0xxJXWR) contains a next card to be processed and X0xxXSEM indicates that JCTXWRK (pointed to by X0xxJXWR) points to an error message to be inserted into the JCL stream (if the job is being failed with a return code of 8 or by setting the bit X0xxKILL).

You can implement many of these functions in z/OS V1R7 as follows:
• To insert cards, queue $RJCB data areas to the $XPL
• To add messages to the JCL input file without causing immediate job failure, use $RMSGQUE.

Timing issues: The timing of when these exits are called has also changed slightly. In releases before z/OS V1R7, as each card and any continuation are encountered, they were passed to the appropriate exit and then processed by JES2. So by the time the second card of a JOB JCL statement was passed to the exit, JES2 had completed processing the first JOB card. In z/OS V1R7, the entire statement is passed to the corresponding exit (one card at a time) before anything is processed by JES2. This allows any exit to insert a card before the current card being processed. It also allows exit 4/54 to even create a new job by inserting a JOB card before the current statement.
Another result of this change is that if a card is added before the current statement, then processing of the current statement is stopped and the inserted statement is processed. When the inserted statement has been processed, the original statement is processed again and passed to the exit a second time. A bit in the $XPL condition byte X0xxSEC indicates that a card has been passed to the exits once before. You need to test this bit to prevent logical loops that repeatedly insert cards before the current statement. If exit 2/52 returns with RC=8, JES2 does not process the JOB card and the JCT fields are not updated from the JOB statement. As a result, default values for the various JOB parameters will be used for later processing.

Other enhancements: A number of significant enhancements have been made to these exits. Exit routines can choose to ignore these changes. However, these enhancements were designed to simplify common functions implemented in exits and make exit writing less error prone.

For more information about each exit, see Exit 2, Exit 4, Exit 52, and Exit 54.

Enhancements to exits 3 and 53
These exits are called to process the accounting string on a job card. In previous releases, register 0 on entry was the length of the accounting string. In z/OS V1R7, register 0 is the address of a new $XPL and the length of the accounting string is placed in the $XPL (X0x3ACTL).

Exit 3 is called for jobs submitted though card readers, RJE, SNA and BSC NJE, and SPOOL reload. Exit 53 is called for jobs submitted through internal readers (including TSO SUBMIT) and TCP/IP NJE.

Other than the addition of the $XPL, the interface to exit 3 has not changed. The interface of exit 53 is the same as exit 3.

Deletion of exit 13
In releases before z/OS V1R7, exit 13 receives control when a SYSOUT data set that is destined for a TSO user is received over NJE. In those releases exit 13 controls whether the $HASP549 MAIL FROM node or user message is issued.

In JES2 z/OS V1R7, exit 13 has been deleted because the existing notify message and associated exit receive control too early in SYSOUT data set receive processing. When a SYSOUT data set is received over NJE, a header record is received first, followed by the actual data. If the SYSOUT data set is large, a TSO user might issue a RECEIVE command and find no data to receive even though exit 13 has already issued message $HASP549. To avoid these problems, in z/OS V1R7, exit 40 is enhanced to allow for control over the $HASP549 message.

Also much of the processing that exit 13 performs is available through other processes in z/OS V1R7 and later releases.

Another reason for deleting exit 13 is that much of the existing functions that were performed in the exit are available elsewhere. For example, the existing parameter MAILMSG= on NJEDEF can control whether or not to send notification messages to TSO users. Some installations used this exit as a way to get control when SYSOUT data set headers are received. This is better accomplished in exits 47 and 57.
Enhancements to exit 40
In z/OS V1R7 and later releases, processing that gets control after exit 40 uses a new bit (PDB9ONOT) to determine if a notify message is needed. If this bit is on, a notify message must be issued. This bit is normally turned on by NJE processing; however, exit 40 can set the PDB9ONOT bit to cause a notify message for a non-NJE SYSOUT data set that is destined to a TSO user. If PDB9ONOT is on, the code also checks a second bit (PDB1NSOT) to determine if the normal $HASP549 notify message is to be issued to the intended recipient of the data set on the local node, or if the error notify message ($HASP548) is to be issued to the notify user on the originating (or notify) node. If PDB1NSOT is on, the error notify message ($HASP548) is issued.

X040RFNT and X040RNNT bits and message $HASP549: In z/OS V1R7 and later releases, two bits in exit 40 $XPL control processing for the normal $HASP549 mail notification message regardless of the setting of MAILMSG= on NJEDEF:
- X040RFNT to issue the message
- X040RNNT to suppress the message

Note that if the PDB1NSOT bit is set, neither the MAILMSG= setting nor the X040RNNT bit can suppress the $HASP549 message; however, turning off PDB9ONOT suppresses all notification messages.

PDB1NSOT bit and suppressing the creation of data sets: Under existing processing, exit 40 can suppress the creation of a data set by setting the PDB1NSOT bit in the PDDB. If the data set is being suppressed, and a notification is required (PDB9ONOT is on), JES2 issues a $HASP548 message. Exit 40 can suppress the notification by turning off the PDB9ONOT bit, or it can allow the message to be issued. If exit 40 allows the message to be issued, it can alter the variable text that indicates why the SYSOUT data set is being deleted by setting X040VTXT in the XPL. The default text is "INVALID USERID". The exit has the option of setting any text up to 20 bytes in length. The text of the $HASP548 message with the default text is as follows:

$HASP548 MAIL TO(NODENAME/USERID) DELETED, INVALID USERID

The $HASP549 message is sent to the local TSO user to whom the data set is destined. The $HASP548 message is sent to the notify user (on the notify node) associated with the job that created the output.

Enhancements to exit 20 and 50
These exits are called at the end of input processing for a job including jobs from internal readers, NJE, or other devices. These exits are called after all the JCL has been processed and the job has been authorized through SAF (RACF), but before the final writing of the JCT and IOTs. JES2 does not affect any changes to register 0 or 1 from previous releases. It does enhance $XML to perform processing that exit 20 performed in previous releases.

Exit 20 is called for jobs submitted though card readers, RJE, SNA and BSC NJE, and SPOOL reload. Exit 50 is called for jobs submitted through internal readers (including TSO SUBMIT) and TCP/IP NJE.

The exit 20 $XPL has been updated with a number of new fields that are set to the current values for the job being processed and can be altered to new values by the exit. Processing in JES2 locates the information for these new $XPL fields and propagates any updates the exit makes to the correct control blocks. Information passed in the $XPL includes the following:
- JES2 job priority
- execution node number
- system affinity
- ARM indicator
- scheduling environment
- job class
- next job phase (JQE type)

Check the $XPL macro for a current list of options.

**Enhancements to exit 39**

Exit 39 gets control when a SYSOUT data set that is destined to a TSO user is received over NJE and the intended recipient will never be able to receive the data set due to security reasons. With RACF, this will only happen if the intended recipient is not permitted to any SECLABEL that can dominate the SECLABEL assigned to the SYSOUT data set. The normal JES2 action is to delete the SYSOUT data set. However, this exit gives installations the option of taking some other action with the SYSOUT data set. Exit 39 is called for SYSOUT received on SNA and BSC NJE, and SPOOL reload.

**Exit 55 for NJE over TCP/IP processing**

In z/OS V1R7, JES2 adds exit 55 for NJE over TCP/IP processing. Exit 55 is called for SYSOUT received on TCP/IP NJE.

Exit 55 is passed a $XPL that has the same fields as the $XPL passed to the existing exit 39. The only additional field passed in the $XPL, to both exits, is the $SRW address.

**Enhancements to exits 46 and 47**

Exits 46 and 47 are the NJE header transmission and reception exits. Exits 46 and 56 get control after a header is built (or updated) but before it is sent over NJE. Exits 47 and 57 get control after a header is received but before any processing for that header is done.

Exits 46 and 47 are called for SNA and BSC NJE, and SPOOL reload. Exits 56 and 57 are called for TCP/IP NJE.

**Exits 56 and 57 for NJE over TCP/IP processing**

In z/OS V1R7, JES2 adds exits 56 and 57 for NJE over TCP/IP processing. Exits 56 and 57 are passed a $XPL that has the same fields as the $XPL passed to the existing exits 46 and 47. The only additional field passed in the $XPL, to both exits, is the device dependent work area address ($JRW, $JTW, $SRW, or $STW).

**New Exit 51**

For z/OS V1R7 and later releases, Exit 51 gets control when $QMOD is moving a job from one queue to another or when a job is leaving execution but is to be queued for re-execution. This exit is intended to be a final point of control in the JES2 main task for jobs being received through NJE over TCP/IP. It will get control as jobs complete INPUT phase processing and SYSOUT completes RECEIVE phase processing. At this point, all the data areas have been written to SPOOL and the JQE is being moved to the next queue.

The address of the $JCT is passed to exit 51 in the $XPL if it is called from job input processing or NJE SYSOUT receiver processing. Because JES2 has already written the $JCT, JES2 will not write it after return from the exit. This was part of the
requirement for exit 51. Exit 7 is sometimes used for end of job input processing instead of exit 20 because if the write of the $JCT or $IOT fails, a job can fail input processing after exit 20 is called. The next phase for the job is definitely known when exit 51 is called since all processing for the current phase has completed. The exit 51 routine can write the $JCT if needed; however, writing the $JCT can impact performance and you should not write it if possible.

Exit 51 and the internal reader: Internal reader and NJE over TCP/IP processing occurs outside the JES2 address space; however, the code must access the JES2 address space to perform some key functions (for example, build JQEs and queue them to the next phase). This processing is accomplished using a new service call $JQESERV. A set of PCEs (the JQE Request Processors) in the main task can also handle these JQE requests (10 of them in all). Under these PCEs, the $QMOD is done and exit 51 is called. The code is careful not to $WAIT for any extended length of time so that the JQE Request Processors can process as many requests as possible. Adding a $CBIO to write the JCT in exit 51 will limit the number of jobs that can be processed by a given JQE Request Processor to one per $JCT write. The design point for internal readers was a single reader submitting hundreds of jobs at once and completing input processing as fast as possible. If this is the environment you are in, the extra I/O will impact performance. If jobs are arriving at a more leisurely rate, you can wait for a $CBIO.

Exit 51 can be used in a number of ways to address problems with the new user environment exits, in particular, the issue of accessing persistent data that is only available in the JES2 address space. In some cases, this information can be moved to common storage or a data space. However, in other cases, moving this information may not be practical, or the data may be movable but the code that maintains the information is assuming main task serialization. In these cases, exit 51 provides a point in processing where main task data can be accessed and actions taken before a job is queued to the next phase of processing.

Exit 51 and policy enforcement: One common function for input processing or NJE receiver exits is policy enforcement. These policies restrict usage of job attributes such as accounting information or time limits. You can also use them to set defaults for job class or MSGCLASS based on the characteristics of a job. In either case, the decision can be based on a table of data that is only available in the JES2 address space. If these decisions can be made based on characteristics available in the JQE and JCT, then this policy enforcement can simply be moved from the existing exits to exit 51; however, if additional information is needed (such as PGM= values), gather that information in exits 52, 53, and 54 and pass it to exit 51 for processing. Use this application for local $JCT extensions allowing up to 8K of data to be passed from exits in the user address space to exit 51 processing in the JES2 address space. The data is not written to SPOOL; therefore, it is not available past exit 51 processing.

Other processing changes

Other exits might be impacted by the changes for z/OS V1R7. Consider the following changes:

Exit 8
Exit 8 gets control for control block I/O outside the JES2 main task (subtasks and user environment). Many of the control block I/Os that previously invoked exit 7 are changed to invoke exit 8. As a result, you should examine exit 8 to ensure that it can properly handle the new callers, especially when your exit 8 depends on data that is established in earlier exit calls. For example, an input processing exit might
be expected to create a JCT extension on which exit 8 depends, and exit 8 might receive control (for example, for internal readers) before the input processing exit that creates the extension gets control.

**Exit 10**
Exit 10 gets control when a $WTO that is issued in the JES2 main task is being queued to the subtask for processing. In z/OS V1R7, a number of $WTOs that are associated with NJE and internal reader processing are now issued out of another address space, and exit 10 does not detect them. If exit 10 is being used to trigger some processing when it detects certain WTO, you might need to move that exit processing to an MVS WTO processing exit. Keep in mind that /*SETUP and /*MESSAGE cards from job input processing for internal reader and NJE/TCP job receivers are still passed to the main task and exit 10 for processing. This processing occurs under the remote console processor PCE instead of the old input processing PCE.

**Exit 14**
Exit 14 allows installations to replace the standard JES2 $QGET processing. Exit 14 includes an indicator of whether duplicate job name processing has been done for the job that was selected. If so, the execution processing does not check for duplicates before selecting this job. A bit QGTFNDUP in the $QGET parameter list indicates that duplicate job name processing has already occurred.

**Exit 27**
Exit 27 gets control when a PCE is attached or detached. Changes made to the format of the data in the NJE related PCEs can impact this exit if it is examining fields that have been renamed or moved to a separate data area. Because there are no longer PCEs for internal readers, any processing of internal reader PCEs is not needed.

**Exit 40**
Exit 40 gets control when JES2 is building a JOE for a PDDB. Processing for NJE notify processing that was formerly in exit 13 has been moved to this exit. The exit might need to be updated to account for this change. For more information, see "Deletion of exit 13" on page 355. Many of the control block I/Os that in JES2 releases before z/OS V1R7 invoked exit 7 are changed to invoke exit 8.

**Exit 49**
Exit 49 gets control when a job has been selected by $QGET (often called the $QGOT exit). In z/OS V1R7 or later releases, the exit gets control in two environments related to $S J (start job) processing and can request a job not be started.

When you enter a $S J command, code in the command processor examines the job specified and decides if it is eligible to be started. As part of that processing, exit 49 is called with an indicator (in X049IND) of X049SJOB. In this case, X049QGT (the $QGET parameter list) is zero.

When the command processor (and exit 49) has decided the job is to be started, a WLM managed initiator attempts to select the job for processing. At that time, another check of eligibility is made because the selection might be occurring on a member other than where the command was processed and conditions might have changed since the last check. As part of that processing, exit 49 is called again with an indicator (in X049IND) of X049SJSJE. X049QGT (the $QGET parameter list) is zero for this call.
Check exit 49 to ensure that it handles the 2 new calls correctly. A check for X049QGT being zero might be required to prevent accessing low storage.

In z/OS V1R7, JES2 also adds X049NDUP exit 49 to indicate that the exit has performed any needed duplicate job name processing. If this bit is set, XEQ processing bypasses its duplicate job name checks. This bit is valid for normal $QGET calls as well as calls from $S J.

**Changes to all Exits**

**Changes to MTTR:** The format of the MTTR used to address records on SPOOL has changed in z/OS V1R7 and later releases as part of the large volume support. Exits that treat the 4-byte MTTR field as a single entity are not impacted by this change. However, if your exit attempts to examine the TT or R field in the MTTR, you must update it.

For z/OS V1R7 and later releases, the format is a 32-bit field MTTR, where the first 8 bits are the M, the next 20 bits are the relative track address, and the last 4 bits are the record number. The DAS$LGDS bit in the $DAS controls whether the new format MTTTR or the old MTTR format is being used.

In addition, a number of fields in the $DAS, $HCT, and the $IOT have been changed with this support. Fields that contained 3 byte MTTs have been changed to 5 byte MTTRTs (also referred to as MQTs – M quad T) and fields that held 2 byte track counts have been changed to 4-byte track counts. More details can be found in the mapping macros.

**Changes to trace ID 13:** An enhancement was also made to trace ID 13 ($EXIT tracing). If an exit is being passed a $XPL, the contents of the $XPL are included in trace id 13. This allows exit writers to see what data is being passed to them in the $XPL and what data is being returned by them. This enhancement should help identify exits that are not functioning as expected.

As of z/OS V1R7, JES2 no longer supports the R4 $ACTIVATE level. The deletion of this support also deleted all the compatibility fields in JES2 control blocks. You must review all exits for code that supports R4 mode. If any is found, you must delete that support in the z/OS V1R7 version of the exits. In particular, you must remove any references to fields with the “R4” suffix because those fields no longer exist. You can often find migration code by looking for references to the field $MSTRVER. As of z/OS V1R7, $MSTRVER is always set to $MSTRV12 (a value of 8). Any code that only runs when $MSTRVER is less than $MSTRV12 is logically dead as of z/OS V1R7.

**Hints and tips**

This section introduces some techniques that you can use to simplify accomplishing specific tasks in exits and some of the problems they solve.

**Field name equivalence**

Exits moved from the JES2 environment to the user environment often have a number of fields that must be converted from $HCT fields to other data areas. The following table lists some of the common field changes:

<table>
<thead>
<tr>
<th>$HCT field Name</th>
<th>Equivalent field in $HCCT/$NJEWORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ZERO, $ZEROS, $ZEROES</td>
<td>CCTZERO, CCTZEROS</td>
</tr>
<tr>
<td>$BLANKS</td>
<td>CCTBLNKS</td>
</tr>
</tbody>
</table>

**Table 16. Common Field Changes**

360 z/OS V1R11.0 JES2 Installation Exits
Table 16. Common Field Changes (continued)

<table>
<thead>
<tr>
<th>$HCT field Name</th>
<th>Equivalent field in $HCCT/$NJEWORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F1</td>
<td>CCTF1</td>
</tr>
<tr>
<td>$F2</td>
<td>CCTF2</td>
</tr>
<tr>
<td>$F4</td>
<td>CCTF4</td>
</tr>
<tr>
<td>$DOUBLE</td>
<td>NJEDBL, SRWDBL, STWDBL, JRWDBL, JTWDBL</td>
</tr>
<tr>
<td>$DWORK</td>
<td>NJEDBLE, SRWDBLE, STWDBLE, JRWDBLE, JTWDBLE</td>
</tr>
<tr>
<td>$DWORK2</td>
<td>NJEDBLE1, SRWDBLE1, STWDBLE1, JRWDBLE1, JTWDBLE1</td>
</tr>
<tr>
<td>$WORK16</td>
<td>NJEWRK16, SRWRK16, STWRK16, JRWRK16, JTWRK16</td>
</tr>
<tr>
<td>$WORK24</td>
<td>NJEWRK24, SRWRK24, STWRK24, JRWRK24, JTWRK24</td>
</tr>
</tbody>
</table>

The usage of $DOGJQE

In the JES2 main task environment, the rule is that you should not assume that you are being passed an update mode artificial JQE (a JQA) and always use $DOGJQE to obtain one if needed. In z/OS V1R7 and later releases, new exits outside the JES2 main task (such as exits 52, 53, and 54) are being passed JQE data areas. Because $DOGJQE does not support the obtaining of update mode JQAs outside the JES2 main task, you cannot update the JQE unless an update mode JQA is passed. The following table lists what is passed to each exit:

Table 17. The usage of $DOGJQE

<table>
<thead>
<tr>
<th>Exit</th>
<th>What is passed to the exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exit 50</td>
<td>This exit is passed an update mode JQA (pointed to by X050JQE)</td>
</tr>
<tr>
<td>Exit 52</td>
<td>This exit is passed an update mode JQA (pointed to by X052JQE)</td>
</tr>
<tr>
<td>Exit 53</td>
<td>This exit is passed an update mode JQA (pointed to by X053JQE)</td>
</tr>
<tr>
<td>Exit 54</td>
<td>This exit is passed an update mode JQA (pointed to by X054JQE)</td>
</tr>
<tr>
<td>Exit 55</td>
<td>There is not JQE address passed in the XPL. However, SRWJQA points to an update mode JQA.</td>
</tr>
<tr>
<td>Exit 56</td>
<td>This exit is passed a read mode JQA (pointed to by X056JQE)</td>
</tr>
<tr>
<td>Exit 57</td>
<td>This exit is passed an update mode JQA (pointed to by X057JQE). If X057BJQE is set, then the area passed can be used as an update mode JQA but the JQE has not been added to the job queue yet.</td>
</tr>
</tbody>
</table>

If you want to verify the type of JQE you have, you can use the following logic:

- TM JQEFLAG2,JQE2ART Is this an artificial JQE?
- JNO NOTUPDM No, not an update mode JQE
- LA Rxx,JQE Get the address
- AH1 Rxx,-(PBESIZE) of the PREBERT
- TM PBEFLAG1-PREBERT(Rxx),PBE1UPDT Update mode?
- JNO NOTUPDM No, not an update mode JQA
- * Update mode JQA.

Issuing messages (WTOs)

When converting main task exits to the new common code exits, you may encounter a message (WTO) that needs to be converted. Most main task exits will use the $WTO service to issue messages. However, $WTO is a macro that has a
calling sequence that differs greatly based on the environment it is invoked in. To deal with this, you can have a number of options:

- Use the appropriate operands based on the environment. If you are writing a single USER,ANY environment routine, you can use a technique similar to the $SEAS example given in the [z/OS JES2 Macros](#).
- Use the $BLDMSG service to issue the message. See [z/OS JES2 Macros](#) for the explanations of how to use $BLDMSG.
- Use a branch entry WTO. Because branch entry WTOs cannot MVS wait, you can use them directly out of the JES2 main task as well as the user environment.
- If you are writing code in an input processing exit before the end of input (for example, in exits 2, 3, 4, 52, 53, or 54), use the $RMSGQUE macro to queue a message for appropriate processing. Using this service, the message is always issued through WTO, but you can also optionally add the message to the JCLIN data set, send the message to the originating node, and send the message to the originating RJE device. You can also add the source job information to the message.
- If the message is intended for a TSO user (such as the notify user), use the $NOTIFY service to issue the message. You can use this out of any exit.

The $NOTIFY, $RMSGQUE or $BLDMSG macros are sensitive to the USER,ANY environment. The services that these macros invoke behave differently based on the environment of the caller. If you are writing a service that is common to both the JES2 main task environment and the USER environment, use the USER,ANY environment.

### Code locations

The following table shows where the code is moved to in z/OS V1R7.

**Table 18. Code Locations**

<table>
<thead>
<tr>
<th>Old Module</th>
<th>New code location</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASPMMSG</td>
<td>HASPMMSG, HASCBLDM</td>
</tr>
<tr>
<td>HASPNET</td>
<td>HASPNET, HASCNJJE</td>
</tr>
<tr>
<td>HASPNJT</td>
<td>HASPNJT, HASCNJJT</td>
</tr>
<tr>
<td>HASPNSR</td>
<td>HASPNSR, HASCNJSR</td>
</tr>
<tr>
<td>HASPNST</td>
<td>HASPNST, HASCNJST</td>
</tr>
<tr>
<td>HASPRDR</td>
<td>HASPRDR, HASCSRIP, HASCINJR, HASCNJJR</td>
</tr>
<tr>
<td>HASPSCAN</td>
<td>HASCSCAN</td>
</tr>
<tr>
<td>HASPSSRV</td>
<td>HASPSSRV, HASCSRIP</td>
</tr>
</tbody>
</table>

### JES2 z/OS V1R5 exit migration details

In support of multi-level security support through RACF, JES2 can limit job selection based on "security label by system" (SECLABELs). JES2 maps the systems for which SECLABELs are active against an affinity mask associated with each batch job. If you use Exit 14 to replace normal JES2 job selection or Exit 36 and Exit 37 to provide Pre- and Post-security authorization calls, you need to consider the following processing interactions.

**Exit 14:** Specifically, you can use Exit 14 (Job queue work select - $QGET) to replace the normal JES2 job selection for job execution or conversion and use
security label by system to affect system affinity. If the RACF SETROPT option for SECLABEL by system is active, then JES2 considers SECLABEL settings when selecting a job. A new field, JOASCALF, contains an affinity mask of JES2 members where the SECLABEL is available. SECLABEL affinity applies to selection of job for conversion and execution, only.

**Exit 36 and Exit 37:** If you code an Exit 36 (Pre-security authorization call) or an Exit 37 (Post-security authorization call) routine, JES2 can now pass a new RACF request type to the exit. JES2 can request a “branch entry extract” to extract information from SECLABEL profiles (WAVREQST field set to WAVRXTRB). In addition, JES2 now also uses the RACF extract (non-branch entry) to extract SECLABELs from various other profiles (WAVREQST field set to WAVRXTRT). Previously, JES2 defined this call but had never used it. New functions codes (SEASCLA - SECLABEL affinity extract and SEASCLE - DCT SECLABEL extract) are now defined for these requests.

**Migration actions:** Examine your Exit 14, Exit 36, and Exit 37 code to ensure that any exit routine processing that needs to access the new security-related fields does so and the exits do not compromise JES2 by not including code needed to replace normal JES2 processing.

### JES2 z/OS V1R2 migration details

**z2 mode for $ACTIVATE**

A new level of $ACTIVATE was created in JES2 z/OS V1R2. This activation level, called z2 Mode, supports JES2’s ability to process increased limits for job processing and constraint relief. Before migrating to z/OS V1R7, you must $ACTIVATE your existing JES2 to this new z2 mode. The following section describes the migration considerations for z2 mode in previous levels of JES2. Some installations might have updated their exits to work with the z/OS V1R2 level of JES2 but not the z2 mode. These exits must be upgraded to support z2 mode on your pre z/OS V1R7 JES2 before performing the $ACTIVATE to z2 mode.

All members of the MAS must be at the JES2 z/OS V1R2 level or higher when your installation issues a $ACTIVATE to **z2 Mode**. You must include the LEVEL= parameter on the $ACTIVATE command to indicate the JES2 level you intend to activate. For example, you would specify $ACTIVATE,LEVEL=z2 to activate your MAS to **z2 Mode**.

**Note:** To display your current level of JES2 processing, issue the $D must code.

Once running in **z2 Mode**, no level of JES2 prior to JES2 z/OS V1R2 is allowed to join the MAS nor allowed to use the JES2 z/OS V1R2 checkpoint. At cold start, JES2 starts in **z2 Mode** unless you specify S JES2 PARM=(UNACT,COLD). To revert back to **R4 Mode**, you must either issue $ACTIVATE,LEVEL=R4 or perform an all-member warm start or all-member hot start using S JES2,PARM=UNACT.

**Note:** In **R4 Mode**, JES2 operates as in JES2 z/OS V1R1.

See z/OS V1R2 JES2 Commands for a description of the $ACTIVATE command.
JES2 z/OS V1R2 must run at least in **R4 Mode**. You do not need to $ACTIVATE to **z2 Mode**. You gain all the **compatible** functional enhancements of JES2 z/OS V1R2 except the z2–mode increased limit enhancements listed in Table 19.

**Note:** This refers to the pre-z/OS V1R7 systems. As of z/OS V1R7, JES2 no longer supports R4 mode.

### Table 19. New z2–mode increased limits

<table>
<thead>
<tr>
<th>JES2 Entity</th>
<th>z2 Mode</th>
<th>R4 Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job number</td>
<td>999,999</td>
<td>65,534</td>
</tr>
<tr>
<td>JQE</td>
<td>200,000</td>
<td>65,534</td>
</tr>
<tr>
<td>JOE</td>
<td>500,000</td>
<td>161,314</td>
</tr>
<tr>
<td>BERTs</td>
<td>500,000</td>
<td>262,136</td>
</tr>
<tr>
<td>TGSPACE</td>
<td>16,581,181</td>
<td>9,935,680</td>
</tr>
</tbody>
</table>

JES2 z/OS V1R2 increases the potential number of job queue elements (JOEs) and output elements (JOEs) above the previous limits. All shared data areas (SPOOL and checkpoint) used by JES2 are compatible with previous releases of JES2. However, the data structures in **R4 Mode** cannot meet the increased limits. Many binary job number fields are only 2 bytes long and chaining fields use 3–byte offsets which cannot address the new limits. **z2 Mode** binary job number fields are 4 bytes long and chaining is accomplished using 3–byte indexes rather than 3–byte offsets. These changes are incompatible with any exit routine or other unique code that examines checkpointed control blocks or processes job numbers. If you have implemented such installation routines, you might need to update them to support **z2 Mode**.

You can use the $ACTIVATE,LEVEL=R4 command to revert to **R4 Mode** of the checkpoint; this does **not** require a JES2 restart. Job number limits and checkpoint data become compatible with JES2 z/OS V1R1 (see Table 19 for these values).

Operator commands that process jobs by job number are improved to handle the processing increases in **z2 Mode**.

$T JOBDEF,RANGE= is updated to allow you to set a job number limit up to a maximum of 999,999. Also, the following $T command changes allow you to take advantage of new maximums for the number of jobs, number of JOEs and number of BERTs as well as to decrease their settings:

- $T JOBDEF JOBNUM= for the number of jobs
- $T OUTDEF JOENUM= for the number of JOEs
- $T CKPTSPACE BERTNUM= for the number of BERTs

See **z/OS JES2 Commands** for a description of these commands.

### Support for a maximum of 999,999 job numbers

In **z2 Mode**, JES2 allows you to define up to 999,999 job numbers (expanding the limit from 65534).

The JOBDEF initialization statement is changed to expand the amount of job numbers you can define. With a new maximum initialized, JES2 interprets and formats job numbers for use in displays as shown in Table 20 on page 365.
Table 20. JobID format based on job number

<table>
<thead>
<tr>
<th>If Job Number is:</th>
<th>Then JobID Format is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 99,999</td>
<td>Jxxxxxxx</td>
</tr>
<tr>
<td>&lt;= 99,999</td>
<td>JOBxxxxx</td>
</tr>
<tr>
<td>&gt; 99,999</td>
<td>Txxxxxxx</td>
</tr>
<tr>
<td>&lt;= 99,999</td>
<td>TSUxxxxx</td>
</tr>
<tr>
<td>&gt; 99,999</td>
<td>Sxxxxxxx</td>
</tr>
<tr>
<td>&lt;= 99,999</td>
<td>STCxxxxx</td>
</tr>
</tbody>
</table>

Notes:
1. JES2 uses the appropriate jobid format in displays of job information. There is a transition period after you use JOBDEF RANGE= or a $T JOBDEF command to increase jobIDs above 99,999 where the "old" format jobIDs might still be displayed. This can occur for jobs that were created when the old range limits were in use. Until these jobs are purged from the system, the old format for jobIDs can be displayed as part of informational messages (and SMF records) about those specific jobs. See [z/OS JES2 Initialization and Tuning Reference](http://www.ibm.com) for the specific changes to the JOBDEF initialization statement.

2. If the $T JOBDEF command is used to increase this value >64K, and use are monitoring JES2 work using SDSF then you might see the jobID for the same job displayed on panels INIT (initialization) and DA (display active) look differently than on the ST (started task) and H (hold) panels. The INIT and DA panels use the MVS work unit ID rather than the format Jxxxxxxx, Sxxxxxxx, or Txxxxxxx they might display as JOBxxxxx, STCxxxxx, or TSUxxxxx.

**JES2 control block field changes**

**General field and control block changes**
Allowing JES2 to use up to 999,999 job numbers requires that you "ACTIVATE" it to **z2 Mode**. Unless this activation is performed, JES2 operates in **R4 Mode** with the job number limitations as follows:

**Note:** **R4 Mode** EQUATE names are in parentheses above.

After you have performed an ACTIVATE to **z2 Mode**, the new limits become those listed in [Table 21](#).

Table 21. z2- and R4 mode comparisons – Field and control block maximum limits

<table>
<thead>
<tr>
<th>Initialization Statement/ Parameter</th>
<th>R4 Mode</th>
<th>z2 Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>Equate</td>
<td>Maximum</td>
</tr>
<tr>
<td>JOBDEF JOBNUM=</td>
<td>65,534</td>
<td>$MAXJQE_R4</td>
</tr>
<tr>
<td>JOBDEF RANGE=</td>
<td>65,534</td>
<td>$MAXJNM_R4</td>
</tr>
<tr>
<td>OUTDEF JOENUM=</td>
<td>161,314</td>
<td>$MAXJOE_R4</td>
</tr>
<tr>
<td>CKPTSPACE BERTNUM=</td>
<td>262,136</td>
<td>4 * $MAXJQE_R4</td>
</tr>
<tr>
<td>SPOOLDEF TGSPACE=MAX</td>
<td>9,935,680</td>
<td>16,581,184</td>
</tr>
</tbody>
</table>
Activation to **z2 Mode** changes the 3-byte offset pointers for JQE chains, JOE chains, and JOE to JOE chains to indexes, and job numbers are stored in four bytes instead of two bytes. JES2 z/OS V1R2 in **z2 Mode** forces an incompatible change to your JES2 system.

Changes to control blocks and data areas that support increasing job numbers to 999,999 focus on increasing the number of JQEs and JOEs above current limits.

- JQE and JOE chaining fields are changed from 3-byte offsets to 3-byte indexes in the checkpoint data set.
- Processing the job queue index (JIX) to convert job numbers is changed from using 2-bytes entries to a hash table.
- A new JQT CTENT can now hold 20,000 JQE extensions. JQE extensions are used to track spool usage for executing jobs.
- The format of the control information for the checkpoint data set limited the amount of data that could be stored in the data set. The format of the control information is enhanced to allow a larger data set.
- Job number fields in control blocks have increased their length from two bytes to four bytes.

One implication of changing JES2 to support a maximum of 999,999 job numbers manifests itself in the jobID format. Based on your upper limit specified for JOBDEF RANGE=, jobid formats can exist as follows:

- **JOBnnnnnn** format is used for upper limit less than or equal to 99,999.
- **J0nnnnnn** format is used for upper limit greater than 99,999.
- **STCnnnnnn** becomes **S0nnnnnn** for a limit greater than 99,999.
- **TSUnnnnn** becomes **S0nnnnnn** for a limit greater than 99,999.
- NJE jobs can use the new jobid format (local jobs cannot) when the original job number is greater than 99,999, the job number is available, and JOBDEF RASSIGN=YES is specified. You should specify JOBDEF RASSIGN=NO if, for jobs from NJE nodes, you want to use job numbers in the local range.

All member of your MAS must be at JES2 z/OS V1R2 or higher to activate to **z2 Mode**. The following list summarizes the changes that take place when you perform such an activation:

- New limits become available.
- Binary job numbers are four bytes long.
- JOE and JQE chaining is by 3-byte index.
- Checkpoint data set format changes are made.
- JIX becomes a hash table (not a look-up table).
- JobIDs take the form **J0nnnnnn** if the job number range is greater than 99,999.
- JES2 can track up to 8 million track groups per job.

**Note:** You should review the general description of activating your JES2 MAS to **z2 Mode**. See “z2 mode for $ACTIVATE” on page 363 for the details.

**z2–mode field changes:** This section summarizes field and field name changes to aid in migrating exit routines. Fields that have had definition changes have also had their names changed. Assembler errors will occur in any exit that references the ”old” field names. When these errors occur, you must correct the use of the fields as well as the field names.
Table 22 lists mode-sensitive JQE pointers. In **R4 Mode**, the pointers are offsets; in **z2 Mode**, they are indexes. The names of the fields are changed to ensure the fields are properly referenced. The field names JOEJQE and JQENEXT refer to composite fields and are eliminated. Composite fields are not available for installation use.

**Table 22. JQE Indexes**

<table>
<thead>
<tr>
<th>JES2 Mapping Macro</th>
<th>Pre-z/OS V1R2 Field Name</th>
<th>z/OS V1R2 Field Name</th>
<th>Field Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HCT</td>
<td>$JQFREE</td>
<td>$JQFREEI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$JQHEADS</td>
<td>$JQHEADI</td>
<td>47 * 4</td>
</tr>
<tr>
<td></td>
<td>$JQRBLD</td>
<td>$JQRBLDI</td>
<td>4</td>
</tr>
<tr>
<td>$JOE</td>
<td>JOEJQEB</td>
<td>JOEJQE</td>
<td>3</td>
</tr>
<tr>
<td>$JQE</td>
<td>JQENEXTB</td>
<td>JQENEXTI</td>
<td>3</td>
</tr>
<tr>
<td>$CAT</td>
<td>CATQHEAD</td>
<td>CATQHD</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** All uses of the old field names must be changed to the new field names. Also you need to consider whether the checkpoint is **R4 Mode** or **z2 Mode**; this determination tells you whether the field contains an offset or an index, respectively.

A JQE offset is converted to an address by adding the value in $JOBQPTR to the offset. A JQE index is converted to an address by multiplying the index by the field, $JQELEN, and then adding $JOBQPTR to the resulting offset.

**Note:** IBM suggests using the $QJQE service to process JQE queues instead of processing them manually. Refer to **z/OS JES2 Macros** for a description of the $QJQE macro.

Table 23 lists mode-sensitive JOE pointers. In **R4 Mode**, the pointers are offsets; in **z2 Mode**, they are indexes. The names of the fields are changed to ensure that the fields are properly used. The field names JQEJOE, JOENEXT, JOEPREV, JOEJQNX, JOECHAR, JOECHNX, JOEWKPTR refer to composite fields and are eliminated to ensure no one misuses composite fields.

**Table 23. JOE Indexes**

<table>
<thead>
<tr>
<th>JES2 Mapping Macro</th>
<th>Pre-z/OS V1R2 Field Name</th>
<th>z/OS V1R2 Field Name</th>
<th>Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$JOT</td>
<td>JOTFREQ</td>
<td>JOTFREQI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>JOTCHRQ</td>
<td>JOTCHRQI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>JOTPURGQ</td>
<td>JOTPRGQI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>JOTHOLDQ</td>
<td>JOTLDQI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>JOTCLSQ</td>
<td>JOTCLSQI</td>
<td>3 * 36 * 4</td>
</tr>
<tr>
<td></td>
<td>JOTNTWKQ</td>
<td>JONTWKQI</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>JOTRDUWQ</td>
<td>JOTRDUQI</td>
<td>4 * JOTNUMQ</td>
</tr>
<tr>
<td></td>
<td>JOTRBLDQ</td>
<td>JOTRBLQI</td>
<td>4</td>
</tr>
<tr>
<td>$JQE</td>
<td>JQEJOEB</td>
<td>JQEJOE</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 23. JOE Indexes (continued)

<table>
<thead>
<tr>
<th>JES2 Mapping Macro</th>
<th>Pre-z/OS V1R2 Field Name</th>
<th>z/OS V1R2 Field Name</th>
<th>Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$JOE</td>
<td>JOENEXTB</td>
<td>JOENEXTI</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>JOEPREV</td>
<td>JOEPREV*</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>JOEJQNXB</td>
<td>JOENJQI</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>JOECHARB</td>
<td>JOECHARI</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>JOECHNXB</td>
<td>JOECHNXI</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>JOEWKPTB</td>
<td>JOEWKPTI</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>JOENETCH</td>
<td>JOENETCI</td>
<td>4</td>
</tr>
</tbody>
</table>

Notes:
1. *JOEPREV*, in **z2 Mode**, is an index unless the high order bit is "on"; then it is an offset.
2. All uses of the old field names must be changed to the new field names. Also, you must consider whether the checkpoint is **R4 Mode** or **z2 Mode**; this determination tells you whether the field contains an offset or an index, respectively.

A JOE offset is converted to an address by adding the value in $JOTABLE to the offset. A JOE index is converted to an address by multiplying the index by the constant, JOESIZE, and then adding $JOTABLE to the resulting offset.

**Note:** IBM suggests that you use the $#JOE service to process JOE queues. $#JOE is particularly useful because it processes the JOEPREVI anomaly without you having to do so. Refer to [z/OS JES2 Macros](#) for a description of the $#JOE macro.

Table 24 shows the 2-byte job number fields that have been changed to four bytes. These changes apply whether you operate in **R4 Mode** or **z2 Mode**.

Table 24. Four byte job number fields

<table>
<thead>
<tr>
<th>Pre-z/OS V1R2 Field Name</th>
<th>z/OS V1R2 Field Name</th>
<th>Control Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRXJOBNO</td>
<td>CRXJBNUM</td>
<td>$COMWORK</td>
</tr>
<tr>
<td>DCNVJBNO</td>
<td>DCNVJNUM</td>
<td>$DTECNV</td>
</tr>
<tr>
<td>FAXBCJP</td>
<td>FAXBJCJP</td>
<td>$FSAXB</td>
</tr>
<tr>
<td>GTWJQNUM</td>
<td>GTWJBNUM</td>
<td>$GTW</td>
</tr>
<tr>
<td>GTWJQEMX</td>
<td>GTWJQMAX</td>
<td>$GTW</td>
</tr>
<tr>
<td>GTWJQEFR</td>
<td>GTWJQFRE</td>
<td>$GTW</td>
</tr>
<tr>
<td>JIBJOBNO</td>
<td>JIBJBNUM</td>
<td>$JIB</td>
</tr>
<tr>
<td>JNEWJQE</td>
<td>JNEWJNUM</td>
<td>$JNEW</td>
</tr>
<tr>
<td>PSOJOBNO</td>
<td>PSOJBNUM</td>
<td>$PSO</td>
</tr>
<tr>
<td>ROTEJBNR</td>
<td>ROTEJNUM</td>
<td>$ROTT</td>
</tr>
<tr>
<td>SFRJBNRO</td>
<td>SFRJBNUM</td>
<td>$SFRB</td>
</tr>
<tr>
<td>SSWJOBNO</td>
<td>SSWJBNUM</td>
<td>$SFSWORK</td>
</tr>
<tr>
<td>SJBJOBNO</td>
<td>SJBJBNUM</td>
<td>$SJB</td>
</tr>
<tr>
<td>TTEJOBNO</td>
<td>TTEJBNUM</td>
<td>$TTE</td>
</tr>
</tbody>
</table>
Note: You must ensure that any use of these fields refers to 4-byte job numbers whether you operate in R4 Mode or z2 Mode.

Table 25 shows the EQUATES that have changed to four bytes.

Table 25. Four byte EQUATES

<table>
<thead>
<tr>
<th>Pre-z/OS V1R2 Binary Job Number Equate Name</th>
<th>z/OS V1R2</th>
<th>Control Block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Equate Name</td>
<td>Value</td>
</tr>
<tr>
<td>DSIDJBNO</td>
<td>DSIDJNUM</td>
<td>EQU 0,4</td>
</tr>
<tr>
<td>$MAXJBNO</td>
<td>$MAXJNUM</td>
<td>EQU 999999</td>
</tr>
<tr>
<td>$MAXJQES</td>
<td>$MAXNJQE</td>
<td>EQU 200000</td>
</tr>
<tr>
<td>COFSEC</td>
<td>COFSEC</td>
<td>COFOPT2+1,4</td>
</tr>
</tbody>
</table>

Note: You must ensure that any use of these EQUATES refers to 4-byte values whether you operate in R4 Mode or z2 Mode.

$SPID, the prefix area for spool-resident control blocks, is changed to contain a 4-byte job number instead of a 2-byte job number. Table 26 shows the field names and control blocks associated with the $SPID.

Table 26. $SPID four byte fields

<table>
<thead>
<tr>
<th>Pre-z/OS V1R2 Field Name</th>
<th>z/OS V1R2 Field Name</th>
<th>Control Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDBJOBNO</td>
<td>HDBJNUM</td>
<td>$BUFFER</td>
</tr>
<tr>
<td>CHKJOBNO</td>
<td>CHKJNUM</td>
<td>$CHK</td>
</tr>
<tr>
<td>IOTJOBNO</td>
<td>IOTJNUM</td>
<td>$IOT</td>
</tr>
<tr>
<td>IOTJBNMB</td>
<td>deleted</td>
<td></td>
</tr>
<tr>
<td>JCTJOBNO</td>
<td>JCTJNUM</td>
<td>$JCT</td>
</tr>
<tr>
<td>NHSJOBNO</td>
<td>NHSJNUM</td>
<td>$NHSB</td>
</tr>
<tr>
<td>OCTJOBNO</td>
<td>OCTJNUM</td>
<td>$OCT</td>
</tr>
<tr>
<td>SWBJOBNO</td>
<td>SWBJNUM</td>
<td>$SWBIT</td>
</tr>
</tbody>
</table>

Note: You must ensure that you use these field to refer to 4-byte job numbers whether you operate in R4 Mode or z2 Mode.

$SIG is changed to use the 1-byte reserved space before the job number field. The new 3-byte job number field has the name, SIGJOBNUM. SIGJOBNO is no longer valid. You must ensure that you use this field to refer to the 3-byte job number whether you operate in R4 Mode or z2 Mode.

The formatted command DSECTs and network header/trailer DSECTs have been adjusted to handle job numbers up to 999,999. These DSECTs have both a 2-byte version and 4-byte version of the job number. You need not consider control block changes for these DSECTs in z/OS JES2 Version 1 Release 2 because JES2 uses the 4-byte job number field if available; otherwise, JES2 uses the 2-byte job number field. Both job number fields are compatible with older releases.

Table 27. Formatted command DSECTs

<table>
<thead>
<tr>
<th>2-Byte Field Name</th>
<th>4-Byte Field Name</th>
<th>Control Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMFJID</td>
<td>COMFJNO</td>
<td>$COMWORK</td>
</tr>
</tbody>
</table>
Table 27. Formatted command DSECTs (continued)

<table>
<thead>
<tr>
<th>2-Byte Field Name</th>
<th>4-Byte Field Name</th>
<th>Control Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSIJID</td>
<td>COSIJNO</td>
<td>$COMWORK</td>
</tr>
<tr>
<td>NJHGJID</td>
<td>NJHGJNO</td>
<td>$NHD</td>
</tr>
<tr>
<td>NJHOOJNO</td>
<td>NJHOOJBN</td>
<td>$NHD</td>
</tr>
<tr>
<td>NMRFJID</td>
<td>NMRFJNO</td>
<td>$NMR</td>
</tr>
<tr>
<td>CMBFJID</td>
<td>CMBFJNUM *</td>
<td>$CMB</td>
</tr>
</tbody>
</table>

* CMBFJNUM is new for z/OS JES2 Version 1 Release 2.

Table 28 shows the changes to job number fields or JQE counter fields in control blocks that JES2 writes to the checkpoint data set. Existing 2–byte fields are renamed and a new 4–byte field created to accommodate the larger job number. In R4 Mode, JES2 uses the existing (but, renamed) 2–byte job number field to process the job number (JES2 does not examine the 4–byte field). After you have "activated" to z2 Mode, JES2 uses the new 4–byte job number field. The 2–byte job number field is discarded.

Table 28. Checkpoint control block fields

<table>
<thead>
<tr>
<th>Pre-z/OS V1R2 Field Name</th>
<th>z/OS V1R2</th>
<th>Control Block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Field Name (R4 Mode)</td>
<td>4-Byte Field Name (z2 Mode)</td>
</tr>
<tr>
<td>DASJOBNO</td>
<td>DASJOBNO_R4</td>
<td>DASJBNUM</td>
</tr>
<tr>
<td>$MAXJOBS</td>
<td>$MAXJOBS_R4</td>
<td>$JQENUM</td>
</tr>
<tr>
<td>$JQEFREC</td>
<td>$JQEFREC_R4</td>
<td>$JQEFRCN</td>
</tr>
<tr>
<td>JCTINJO</td>
<td>field deleted</td>
<td>field deleted</td>
</tr>
<tr>
<td>JQEJOBNO</td>
<td>JQEJOBNO_R4</td>
<td>JQEJBNUM</td>
</tr>
<tr>
<td>JQEINJNO</td>
<td>JQEINJNO_R4</td>
<td>JQXIJNUM</td>
</tr>
<tr>
<td>JQARMID</td>
<td>JQARMID_R4</td>
<td>JQARMMI</td>
</tr>
<tr>
<td>JQEWSLOK</td>
<td>JQEWSLOK_R4</td>
<td>JQEWSLCK</td>
</tr>
</tbody>
</table>

$SHCT Reserved space in the HCT is now used for the two new 4–byte fields that identify maximum number of jobs ($JQENUM) and JQE free count ($JQEFRCN).

$SJCT The 2–byte original job number JCTINJNO is deleted from the JCT and NOT replaced with a 4–byte job number. A pre-z2 level cannot rely on this field.

$SJOE The new 4–byte initial job number field (JQXIJNUM) uses previous reserved space in the JQX. JES2 processes four byte job numbers (JQEJBNUM) in the JQX in the fields once occupied by JQARMID, JQEWSLOK and two reserved bytes. When JES2 runs in z2 Mode, the contents of the JQARMID and JQEWSLOK fields are stored in the two bytes used by JQENWSU. JES2 accesses this data using the names JQARMMI and JQEWSLCK respectively. Figure 13 on page 371 shows the changes.
When examining fields in the real JQE/JQX, the exit must be sensitive to the checkpoint mode. If in **R4 Mode**, you must use the fields with the "_R4" suffix. In **z2 Mode**, you must use the new z/OS V1R2 4-byte fields. When using a JQA (artificial JQE), always use the new z/OS V1R2 fields regardless of whether you are in **R4 Mode** or **z2 Mode**.

**Note:** When JES2 returns or checkpoints the JQA, $DOGJQE sets the appropriate fields in the real JQE depending on whether running in **R4 Mode** or **z2 Mode**. $DOGJQE also sets the 4-byte fields in the JQA and alleviates the need for code that uses the JQA to determine whether JES2 is processing the checkpoint data set in **R4 Mode** or **z2 Mode**.

**Macro updates relating to JOE and JQE changes:** There are some new and changed macros used to minimize the impact to JES2 exit routines. By using these macros, your exit routines do not have to determine whether JES2 is processing in **R4 Mode** or **z2 Mode**.

**$QJQE and $#JOE**

These macros process the JOE and JQE control block chains regardless of checkpoint mode. $#JOE is changed to process chains (such as the CHAR JOE chain). $QJQE and $#JOE have improvements to their loop control.

**$DOGJQE**

The JQA returned by $DOGJQE always reflects the **z2 Mode** processing.
format. Code that processes an artificial JQE (JQA) and examines JQE/JQA fields need not care whether JES2 is running in z2 Mode or R4 Mode.

$JQEJNUM
$JQEJNUM is a new macro used to obtain the binary job number for a particular JQE. This macro operates in z2 Mode and R4 Mode and processes the JQA or JQE that is passed.

$JBIDBLD
This macro converts a binary job number to a printable jobid. $JBIDBLD is updated to accept a JQE (or JQA) as input and operates in z2 Mode or R4 Mode. $JBIDBLD formats the jobid, in the correct way, based on the current job number range.

JESNEWS: In R4 Mode, JESNEWS processing is exactly like JES2 z/OS V1R1 except for field name changes. In z2 Mode, JESNEWS processing uses a JESNEWS level number to correlate the JOE and the JNEW control blocks. This JNEW control block contains the job number of the JESNEWS JQE.
Appendix A. JES2 exit usage limitations

The following table notes those instances when reader and converter exits (Exits 2, 3, 4, 6, and 20) are invoked or not invoked. Be certain to consider this information when attempting to implement these exits.

*Table 29. Reader and Converter Exits Usage*

<table>
<thead>
<tr>
<th>Exits Taken for</th>
<th>Input Services</th>
<th>Converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source of Job</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job from local reader</td>
<td>Y(2)</td>
<td>Y</td>
</tr>
<tr>
<td>Job from remote reader</td>
<td>Y(2) 1</td>
<td>Y</td>
</tr>
<tr>
<td>TSO session logon (TSU)</td>
<td>Y(52) 1</td>
<td>Y</td>
</tr>
<tr>
<td>TSO submitted job</td>
<td>Y(52) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Started task</td>
<td>Y(52) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Job with /*ROUTE XEQ - INTRDR, NJE/TCP</td>
<td>Y(52) 1</td>
<td>N</td>
</tr>
<tr>
<td>Job with /*ROUTE XEQ - Other sources</td>
<td>Y(2) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Job following /*XMIT JECL or //XMIT JCL</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Job from NJE job receiver:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job for this node - TCP/IP</td>
<td>Y(52) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Job for this node - SNA, BSC</td>
<td>N(2) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Store and forward - TCP/IP</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Store and forward - BSC, SNA</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Job from NJE SYSOUT receiver:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Job for this node - BSC, SNA, TCP/IP</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Store and forward - BSC, SNA</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Job internally generated by JES2 (SYSLOG-RMTMSG)</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Spool offload job receiver 2</td>
<td>Y(2) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Spool offload SYSOUT receiver</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>XBM invocation - INTRDR, NJE/TCP</td>
<td>Y(52) 1</td>
<td>Y</td>
</tr>
<tr>
<td>XBM invocation - other sources</td>
<td>Y(2) 1</td>
<td>Y</td>
</tr>
<tr>
<td>Special Case JCL and JECL</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>JCL from cataloged procedure</td>
<td>N/A</td>
<td>N/A Y</td>
</tr>
<tr>
<td>//*COMMENT cards - INTRDR, NJE/TCP</td>
<td>Y(52) N/A</td>
<td>N/A Y</td>
</tr>
<tr>
<td>//*COMMENT cards - other sources</td>
<td>Y(2) N/A</td>
<td>N/A Y</td>
</tr>
<tr>
<td>*/PRIORITY statements- INTRDR, NJE/TCP</td>
<td>N/A N/A Y</td>
<td>N/A Y</td>
</tr>
<tr>
<td>*/PRIORITY statements - other sources</td>
<td>N/A N/A Y</td>
<td>N/A Y</td>
</tr>
<tr>
<td>*/$command statements - INTRDR, NJE/TCP</td>
<td>N/A N/A Y</td>
<td>N/A Y</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1988, 2009
Table 29. Reader and Converter Exits Usage (continued)

<table>
<thead>
<tr>
<th>Comment</th>
<th>Exit 1</th>
<th>Exit 2</th>
<th>Exit 3</th>
<th>Exit 4</th>
<th>Exit 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>/*$command statements - other sources 3</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(4)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>/*end of SYSIN data</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>/*null statements</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Generated DD*statement - INTRDR, NJE/TCP</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(54)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Generated DD*statement - other sources</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(4)</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>/*with invalid verb - INTRDR, NJE/TCP</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(54)</td>
<td>Y(50)</td>
<td>N/A</td>
</tr>
<tr>
<td>/*with invalid verb - other sources</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(4)</td>
<td>Y(20)</td>
<td>N/A</td>
</tr>
<tr>
<td>//with invalid verb - INTRDR, NJE/TCP</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(54)</td>
<td>Y(50)</td>
<td>N/A</td>
</tr>
<tr>
<td>//with invalid verb - other sources</td>
<td>N/A</td>
<td>N/A</td>
<td>Y(4)</td>
<td>Y(20)</td>
<td>N/A</td>
</tr>
<tr>
<td>/*EOF internal reader</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>/*DEL internal reader - INTRDR, NJE/TCP</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>Y(50)</td>
<td>N</td>
</tr>
<tr>
<td>/*DEL internal reader - other sources</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>Y(20)</td>
<td>N</td>
</tr>
<tr>
<td>/*PURGE internal reader - INTRDR, NJE/TCP</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>Y(50)</td>
<td>N</td>
</tr>
<tr>
<td>/*PURGE internal reader - other sources</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>Y(20)</td>
<td>N</td>
</tr>
<tr>
<td>/*SCAN internal reader</td>
<td>N/A</td>
<td>N/A</td>
<td>N</td>
<td>N/A</td>
<td>N</td>
</tr>
</tbody>
</table>

Where Y(n) = Exit is invoked and number, N = Exit is not invoked, and N/A = Not applicable

Notes:
1. Exit 3/53 is taken only if ACCTFLD=REQUIRED or OPTIONAL is specified on the JOBDEF initialization statement. Exit 3/53 will be taken even if there is no accounting information provided on the JOB statement.
2. This may be the second (or more) pass through these exits for this job.
3. Commands must be outside of a job; they will invoke Exit 4/54 but will not have a JCT (R10=0).
Appendix B. Sample code for Exit 17 and Exit 18

The following is code that your installation can include in installation Exit 17 and Exit 18 to remove blanks from the remote workstation identifier on the RJE signon cards.

```
X1718 $MODULE ENVIRON=JES2,TITLE='JES2 EXIT 017 - $MODULE',
      $CADDR, JES2 Common Address Table
      $HASPEQU, JES2 Equates
      $HCT, JES2 Control Table
      $HFAME, JES2 File Allocation Map Entry
      $MIT, JES2 Module Information Table
      $MITETBL, JES2 MIT Entry Table
      $SPADDR, JES2 Private Routine Address Table
      $SPARMLST, JES2 Parameter list
      $PCE, JES2 Processor Control Element
      $PSV, JES2 Prefix Save Area
      $SCAT, JES2 Sysout Class Attribute Table
      $USERCBS, User Control Blocks
      $XECB JES2 Extended ECB
X17DBLNK $ENTRY CSECT=YES,BASE=R12
      SPACE 1
      $SAVE Save caller's registers
         LR R12,R15 Save base address
         SLR R6,R6 Preset return code
         LTR R0,R0 Is this the first call for signon?
         BNZ X17RET No, return now
         EJECT
      ************************************************************************
      * The card image passed to this routine by JES2 will always have a blank after the characters '/SIGNON'. *
      ************************************************************************
      SPACE 1
      L R2,12(R1) Point to the signon card
      LA R2,15(R2) Point to remote number portion
      SPACE 1
```
Now get past the 'RMT' or 'R'.

SPACE 1
SLR R7,R7 Zero number of blanks found
LA R5,L'X17FIELD Get max length of remote field
LA R4,L'X17REMOT Assume that it is 'REMOTE'
CLC X17REMOT,0(R2) Does it start with 'REMOTE'? 
BE X17FNUM Yes, go process the number
LA R4,L'X17RMT Assume that it is 'RMT'
CLC X17RMT,0(R2) Does it start with 'RMT'? 
BE X17FNUM Yes, go process the number
LA R4,L'X17RM Assume that it is 'RMT'
CLC X17RM,0(R2) Does it start with 'RM'? 
BNE X17RET No, can't do anything with it

X17FNUM LA R2,0(R4,R2) Point to character after remote
SR R5,R4 Get count of numbers in field
LR R4,R5 Save number of numbers
LR R3,R2 Save start of number portion

X17LOOP CLI 0(R2),C' ' Is the next char a blank?
BNE X17SKWSH No, all done
LA R7,1(,R7) Increment number of blanks found
LA R2,1(,R2) Point to next character
BCT R5,X17LOOP And continue de-blanking
B X17RET No numbers, all blanks
EJECT
**Move the characters over and then fill the rest of the remote number portion of the field with blanks.**

*SPACE 1*

```
X17SKWH LTR R7,R7        Were any blanks found?
BZ X17RET                No, line is OK
SR R4,R7                 Get number of numbers
BCTR R4,0                 Less one for execute
EX R4,X17MOVE1           Move the characters over
LA R3,1(R4,R3)           Point past numbers
BCTR R7,0                 Less one for execute
EX R7,X17MOVE2           Blank out remaining characters
SPACE 1
X17RET $RETURN RC=(R6)    Return to the caller
EJECT
```

**Executed statements and storage areas**

*SPACE 1*

```
X17MOVE1 MVC 0(*=*R3),0(R2)        Squish out those blanks
X17MOVE2 MVC 0(*=*R3),X17BLANK    Squish out those blanks
SPACE 1
X17BLANK DC CL9' '
X17FIELD DC C'REMOTE999'
X17REMT DC C'REMOTE'
X17RMT DC C'RMT'
X17RM DC C'RM'
```

**APPENDIX B. Sample code for Exit 17 and Exit 18**

377
Appendix C. Job-related exit scenarios

This appendix identifies the JES2 job-related exits. It also describes the relationship between the JES2 $JCT and MVS/SP™ JMR blocks and provides an overview of the security access service.

Examples of exits that are not job-related are exits such as those taken during JES2 initialization, JES2 termination, RJE signon, JES2 command processing, and other functions not necessarily related to individual jobs.

Job-related exits fall into two categories: specific purpose and general purpose. A specific purpose job-related exit is one that provides a specific function. Although, it may be used for other purposes such as a compromise to avoid in-line modifications.

Examples of specific-purpose job-related exits are job output overflow (Exit 9) and spool partitioning exits (Exits 11 and 12). These exits are used in controlling output limits and spool allocation (fencing) for a particular job. Because these exits do not occur at predictable intervals during the life of a job, using them for a general purpose is not appropriate.

General-purpose job-related exits are exits such as the job statement scan exit (Exit 2), converter internal text scan exit (Exit 6), and the control block read/write exits (Exits 7 and 8). These exits are typically considered when there is a user requirement to control installation standards, job resources, security, output processing, and other job-related functions.

Often the use of more than one exit is required and sometimes combinations of JES2 and other exits such as Systems Management Facilities (SMF) exits must be used. Table 30 on page 380 lists the exits that are discussed. They are not all of the job-related exits but possibly enough to make a decision as to which exits to choose to control certain processes or functions during the life of a job.

Exit sequence

There are two major considerations when selecting an exit to satisfy a user requirement:

1. The environment of the exit -
   The address space, TCB (task), storage key, data areas that are addressable, and facilities are available at the time the exit is taken.

2. The sequence of the exits -
   Which exits precede and which exits follow each other? What processing has preceded and what processing follows the exit?

Selected exits

To provide a user-required function, two or more exits may be needed. In that case, understanding the sequence of exits is important.

---

1. A job, in JES2 terminology, is anything represented by a Job Queue Element ($JQE). The name “job” is also used to describe job output rather than the more specific term - spool data set. It is common for operators to say that a “job” is on the printer or a “job” is printing. It would be awkward, but more accurate, to say that the data set or output group is printing.
Table 30 lists the selected exits that are included here for further discussion.

**Table 30. Job-Related Exits**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Print/Punch Separator</td>
<td>Taken when a job’s data sets have been selected for printing or punching, before the check for the standard separator page.</td>
</tr>
<tr>
<td>2</td>
<td>JOB Statement Scan</td>
<td>The first exit taken for a job and before the statement is processed.</td>
</tr>
<tr>
<td>3</td>
<td>Job Statement Accounting Field Scan</td>
<td>Taken after JOB statement has been processed. Normally used to replace or supplement JES2’s accounting field scanning routine (HASPRSCN), but also used as a post job card exit.</td>
</tr>
<tr>
<td>4</td>
<td>JCL and JECL control statement scan</td>
<td>Taken for each JCL and JECL statement submitted but not for PROCLIB JCL statements.</td>
</tr>
<tr>
<td>6</td>
<td>Converter/Interpreter internal text scan</td>
<td>A good exit for scanning JCL because of structured text and single record for each statement (no continuation).</td>
</tr>
<tr>
<td>7</td>
<td>$JCT Read/Write (JES2 environment)</td>
<td>Receives control when JES2 maintask reads or writes the $JCT.</td>
</tr>
<tr>
<td>8</td>
<td>Control Block Read/Write (User or Subtask environment)</td>
<td>Taken from the user address space or a JES2 subtask each time a spool resident control block ($JCT, $IOT, $SWBIT, $OCR) is read from or written to spool.</td>
</tr>
<tr>
<td>15</td>
<td>Output Data Set/Copy Select</td>
<td>Taken once for each data set where the data set’s SPDDB matches the selected Job Output Element ($JOE) and once for each copy of these data sets.</td>
</tr>
<tr>
<td>20</td>
<td>End of Job Input</td>
<td>Taken at the end of input processing and before $JCT is written. This is typically a good place to make final alterations to the job before conversion.</td>
</tr>
<tr>
<td>28</td>
<td>SSI Job Termination</td>
<td>Taken at the end of job execution before the $JCT is written to spool.</td>
</tr>
<tr>
<td>30</td>
<td>SSI Data Set Open/Restart</td>
<td>Taken for SYSIN, SYSOUT, or internal reader Open or Restart processing.</td>
</tr>
<tr>
<td>31</td>
<td>SSI Allocation</td>
<td>Taken for SYSIN, SYSOUT, or internal reader Allocation processing.</td>
</tr>
<tr>
<td>32</td>
<td>SSI Job Selection</td>
<td>Taken after all job selection processing is complete.</td>
</tr>
<tr>
<td>33</td>
<td>SSI Data Set Close</td>
<td>Taken for SYSIN, SYSOUT, or internal reader Close processing.</td>
</tr>
<tr>
<td>34</td>
<td>SSI Data Set Unallocate - Early</td>
<td>Taken for SYSIN, SYSOUT, or internal reader Unallocate processing. This exit is taken early in Unallocation. You may want to consider Exit 48 (late unallocation) when modifying SYSOUT characteristics.</td>
</tr>
<tr>
<td>35</td>
<td>SSI End-of-Task</td>
<td>Taken at end of each task during job execution.</td>
</tr>
<tr>
<td>36</td>
<td>Pre-SAF</td>
<td>Taken just before JES2 call to SAF.</td>
</tr>
<tr>
<td>37</td>
<td>Post-SAF</td>
<td>Taken just after the return from the JES2 call to SAF.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exit Title</td>
<td>Comment</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>40</td>
<td>Modifying SYSOUT characteristics</td>
<td>Taken during OUTPUT processing (HASPHOPE or HASPSPIN) for each SYSOUT data set before JES2 gathers data sets with like attributes into a $JOE.</td>
</tr>
<tr>
<td>44</td>
<td>Post Conversion - Maintask</td>
<td>Taken in maintask environment after job conversion processing and before the $JCT and $JQE are checkpointed</td>
</tr>
<tr>
<td>46</td>
<td>NJE Transmission</td>
<td>Taken for NJE header, trailer, and data set header during NJE job transmissions.</td>
</tr>
<tr>
<td>47</td>
<td>NJE Reception</td>
<td>Taken for NJE header, trailer, and data set header during NJE job reception.</td>
</tr>
<tr>
<td>48</td>
<td>SYSOUT Unallocation - Late</td>
<td>This exit can be used as an alternative to Exit 34 (early allocation). It is more suitable when modifying SYSOUT characteristics or affecting SPIN processing. When modifying SYSOUT characteristics in Exit 34, subsequent JES2 processing can override changes made to the $PDDB in the exit. If processing is required earlier, use Exit 34.</td>
</tr>
<tr>
<td>49</td>
<td>Job Queue Work Select - QGOT</td>
<td>This exit allows you to gain control whenever JES2 work selection processing has located a pre-execution job for a device. This includes work selected for JES2 and workload management (WLM) initiators.</td>
</tr>
<tr>
<td>50</td>
<td>End of Job Input</td>
<td>Taken at the end of input processing and before $JCT is written. This is typically a good place to make final alterations to the job before conversion.</td>
</tr>
<tr>
<td>51</td>
<td>Job Phase Change</td>
<td>Taken when a job moves from one phase to the next.</td>
</tr>
<tr>
<td>52</td>
<td>JOB Statement Scan</td>
<td>The first exit taken for a job and before the statement is processed.</td>
</tr>
<tr>
<td>53</td>
<td>Job Statement Accounting Field Scan</td>
<td>Taken after JOB statement has been processed. Normally used to replace or supplement JES2’s accounting field scanning routine (HASPRSCN), but also used as a post job card exit.</td>
</tr>
<tr>
<td>54</td>
<td>JCL and JECL control statement scan</td>
<td>Taken for each JCL and JECL statement submitted but not for PROCLIB JCL statements.</td>
</tr>
<tr>
<td>56</td>
<td>NJE Transmission</td>
<td>Taken for NJE header, trailer, and data set header during NJE job transmissions.</td>
</tr>
<tr>
<td>57</td>
<td>NJE Reception</td>
<td>Taken for NJE header, trailer, and data set header during NJE job reception.</td>
</tr>
<tr>
<td>IEFUJV</td>
<td>SMF Job Validation</td>
<td>Receives control for each JCL statement and at the conversion end from the converter subtask. IEFUJV receives control from the user’s address space after all JCL is interpreted.</td>
</tr>
<tr>
<td>IEFUJI</td>
<td>SMF Job Initiation</td>
<td>Taken at job initiation after the $JCT has been checkpointed and before SMF exit IEFUSI.</td>
</tr>
<tr>
<td>IEFUJP</td>
<td>SMF Purge</td>
<td>Taken from subtask in JES2 address space after job is purged.</td>
</tr>
</tbody>
</table>
Table 30. Job-Related Exits (continued)

<table>
<thead>
<tr>
<th>Exit</th>
<th>Exit Title</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEFUSI</td>
<td>SMF Step Initiation</td>
<td>Taken just after SMF exit IEFUJI for the first step of a job. Also taken again at the beginning of each subsequent step.</td>
</tr>
<tr>
<td>IEFACTRT</td>
<td>SMF Termination</td>
<td>Receives control at job and step termination and for the creation of SMF type 5 and 35 records.</td>
</tr>
</tbody>
</table>

SPOOL control blocks

It’s important to understand the status of any control block to be referenced or altered in a user exit. Control blocks associated with a job may not always be in storage. However, all job-related control blocks are written to either the checkpoint data set or a spool data set. This is done to:

- Allow warm starts after JES2 termination.
- Make control blocks accessible to all sharing members of a multi-access spool complex.
- Provide recovery in case of a system failure.

Sometimes job-related control blocks are just read and not written (if they are not altered) but are always written after they are created and after they have been altered. The job-related control blocks on spool are:

- $JCT - Job Control Table
- $IOT - I/O Table (contains spool track allocation and spool data set information)
- $OCT - Output Control Table (contains Output Control Records (OCRs) which are used for /*OUTPUT JCL parameters)
- $SWBIT - SWB Information Table (contains Scheduler Work Blocks used by //OUTPUT JCL)
- $CHK - Checkpoint record for local, RJE and FSS printers.

Checkpoint control blocks

If you write code for JES2 exits that access and update checkpoint control blocks, you need to review this section and apply this information along with those specific “Programming Considerations” described for the JES2 exit that you are implementing.

Checkpoint control blocks for JQEs

JES2 provides different types of JQEs or JQAs to your exit and processes them in differing ways. The types are:

- Real JQE. Your exit receives a read or update mode JQE or JQA.
- Read-mode JQA. Your exit receives an artificial JQE that is a temporary block of storage. This storage contains:
  - Almost the same information as the real JQE.
  - Information from the JQX (new in Version 2 Release 4).
  - Information from BERTs (another checkpointed area).
- Update-mode JQA. Your exit receives an artificial JQE that is a temporary block of storage. This storage is similar to the read-mode JQA. JES2 ensures the integrity of this JQA and manages the storage that each JQA occupies.
- Work area that contains a prototype JQE. In certain circumstances, your exit may be passed the address of a work area that contains a working copy of a JQE. See Exit 47 for more information.

Exits normally use JQEs in read mode (data is extracted or pointed to when calling service routines) or in write mode (data in the JQE is modified). JES2 exit writers need to take the following actions when they use a particular JQE or JQA as the JQE= keyword value on the $DOGJQE macro:

- If the JQE is needed only to access data and that data is within the bounds of the original real JQE, only the address of the real JQE is needed. Regardless of what IBM has provided as the JQE address, use the following action to get the address of the real JQE:
  
  $DOGJQE ACTION=GETJQEADDR,CBADDR=jqe

- If the JQE is needed only to access data and that data is beyond the bounds of the original real JQE (that is, it is stored in fields where the first three characters of the field name are other than JQE), a read mode JQA is needed. Regardless of what IBM has provided as the JQE address, use the following action to get the address of a read mode JQA. The address of the read mode is returned in R0.
  
  $DOGJQE ACTION=(FETCH,READ),JQE=jqe

  After you finish, use the following action to free the memory that is used for the JQA (x is the address that is returned from the first $DOGJQE call):
  
  $DOGJQE ACTION=RETURN,CBADDR=x

- If the JQE is needed in write mode (the fields to be changed are either within the bounds or not within the bounds of the original JQE), use the following action to get the address of an update mode JQE, regardless of what IBM has provided as the JQE address. The address of the JQA is returned in R0. Make all changes to fields in the update mode JQA.
  
  $DOGJQE ACTION=(FETCH,UPDATE),JQE=jqe

  After you finish, use the following action to free the memory that is used for JQA (x is the address from the first $DOGJQE call) and to ensure that the changes in the JQA get propagated to the real JQE, the JQX, and the BERT area.
  
  $DOGJQE ACTION=RETURN,CBADDR=x

**Update-mode JQA considerations:** If an exit requires an update-mode JQA, use the following logic path:

1. Perform the action:
   
   $DOGJQE ACTION=(FETCH,UPDATE),JQE=jqe, WAIT=NO

   where jqe is the address of the JQE that the IBM code gives to the exit.

2. If JES2 returns a return code indicating that the JQA could not be created, you must manage the situation of lock not available.

3. If RC=0, perform rest of logic by using the JQA.

4. Perform the action:
   
   $DOGJQE ACTION=RETURN,CBADDR=jqa

   where jqa is the address that is returned in R0 from FETCH in the first step.

**Note:** It is not necessary or desirable to perform the following action before you attempt to get an update-mode JQA.

   $DOGJQE ACTION=(QUERYLOCK,OBTAINABLE)
This is valid because the non-zero return code (that is, the failure RC) returned by QUERYLOCK indicates that the lock is not available for a new user. This condition is different from requesting an update-mode JQA for the current caller.

**Other processing considerations:** When your exit returns a JQE or JQA to the JES2 systems through these actions, certain errors can occur if JES2 determines that what your exit has returned is not consistent with what JES2 knows to exist. JES2 uses the $ERROR macro and issues the following errors:

- DJ1– Non-IBM code returned an IBM JQE or JQA that violates the consistency checks of JES2.
- DJ2– IBM code returned a non-IBM JQE or JQA that violates the consistency checks of JES2.

**Notes:**

1. You are encouraged to disregard the kind of JQE or JQA that is passed to your exit and always to do the following actions:
   - To obtain the address of the real JQE (for example, your exit needs to compute the offset of the JQE), perform the action:
     
     $DOGJQE ACTION=GETJQEADDR
   - To obtain the address of a read–mode JQE or JQA (for example, your exit needs to examine the MAXCC field), perform the action:
     
     $DOGJQE ACTION=(FETCH,READ)
   - To obtain the address of an update–mode JQE or JQA (for example, your exit needs to change the SYSAFF or PRIORITY or MAXCC), perform the action:
     
     $DOGJQE ACTION=(FETCH,UPDATE)

2. If you are writing Exit 47, do not use $DOGJQE to access a JQE or a JQA.
3. If you are writing user environment exits, such as Exit 50, Exit 52, Exit 53, Exit 54, or Exit 57, do not use $DOGJQE to obtain an update mode JQA. These exits, when passed a JQE, will always be passed an update-mode JQA. Exit 56 will always be passed a read-mode JQA.
4. If you are writing JES2 exits that are in the following situations:
   - Run outside the JES2 main task
   - Need to access or update checkpoint control blocks
     
     you need to follow the specific coding recommendations in “Checkpoint control blocks” on page 382 and those specific “Programming Considerations” listed for the JES2 exit that you are implementing.

**Checkpoint control blocks for JOEs**

JES2 provides different types of artificial JOEs (that is, JOAs) to your exit and processes them in differing ways. The types are:

- Read-mode JOA. Your exit receives an artificial JOE that is a temporary block of storage. This storage contains:
  - Information about the Work JOE
  - Information about the Characteristics JOE
  - Information about the JOE Extension (JOX)
  - Information about BERTs (another checkpointed area). BERT data that is owned by JOEs is new for JES2 release V1R11 code running in z11 checkpoint activation mode. For more information about JES2 z11 activation see the $ACTIVATE and $DACTIVATE commands in [z/OS JES2 Commands](/zos/v1r11/commands/jes2/zos_jes2_commands.html).
• Update-mode JOA. Your exit receives an artificial JOE that is a temporary block of storage. This storage is similar to the read-mode JOA. JES2 ensures the integrity of this JOA and manages the storage that each JOA occupies.

• Work area that contains a prototype JOA. In certain circumstances, your exit may be passed the address of a work area that contains a working copy of a JOA. For example, a prototype JOA is embedded in the JOE Information Block ($JIB). See Exit 23 for more information.

Exits normally use JOAs in read mode (data in the JOA is used but not modified) or in write mode (data in the JOA is modified). The exit should always obtain either a READ or an UPDATE mode JOA. The use of the real JOE should be avoided if possible. JES2 exit writers need to take the following actions:

• If a JOA is needed only to access data, a local read mode JOA should be obtained. Regardless of what IBM has provided as the JOA address, use the following action to obtain the address of a read mode JOA. The address of the local read mode JOA is returned in R0.

  `$DOGJOE ACTION=(FETCH,READ), JOE=joa`

  where `joa` is the address of the JOA that IBM code provides to the exit.

  After you finish, use the following action to free the memory that is used for the local read mode JOA:

  `$DOGJOE ACTION=RETURN, CBADDR=joa`

  where `joa` is the JOA address that is returned from the first `$DOGJOE` call.

• If the exit must modify JOA fields, a local update mode JOA should be obtained. Regardless of what IBM has provided as the JOA address, use the following action to obtain the address of an update mode JOA. The address of the local update mode JOA is returned in R0. Make all changes to fields in the local update mode JOA.

  `$DOGJOE ACTION=(FETCH,UPDATE), JOE=joa`

  where `joa` is the address of the JOA provided to the exit by IBM code.

  After you finish, use the following action to free the memory that is used for the local update mode JOA and to ensure that any changes that are made in the JOA are propagated to the real work JOE, the real characteristics JOE, the JOX, and the BERT area.

  `$DOGJOE ACTION=RETURN, CBADDR=joa`

  where `joa` is the JOA address returned from the first `$DOGJOE` call.

**Update mode JOA considerations for wait conditions:** If an exit requires an update-mode JOA, but cannot wait for a possible conflict to be resolved, use the following logic path:

1. Perform the action:

   `$DOGJOE ACTION=(FETCH,UPDATE), JOE=joa, WAIT=NO`

   where `joa` is the address of the JOA that IBM code provides to the exit.

2. If JES2 returns a return code indicating that the update mode JOA could not be created, you must manage the situation of lock not available.

3. If RC=0, perform rest of the exit logic by using the update mode JOA.

4. Perform the action:

   `$DOGJOE ACTION=RETURN, CBADDR=joa`
where joa is the address that is returned in R0 from FETCH in the first step.

Other processing considerations: When your exit returns a JOA to the JES2 systems through these actions, certain errors can occur if JES2 determines that what your exit has returned is not consistent with what JES2 knows to exist. JES2 uses the $ERROR macro and issues the following errors:

- D01– Non-IBM code returned an IBM JOE or JOA that violates the consistency checks of JES2.
- D02– IBM code returned a non-IBM JOE or JOA that violates the consistency checks of JES2.

Notes:

1. You are encouraged to disregard the kind of JOA that is passed to your exit and always to do the following actions:
   - To obtain the address of a read–mode JOA (for example, your exit needs to examine but not change the JOEFORM field), perform the action:
     $DOGJOE ACTION=(FETCH,READ)
   - To obtain the address of an update–mode JOA (for example, your exit needs to change the JOEHSRSN field), perform the action:
     $DOGJOE ACTION=(FETCH,UPDATE)

2. If you are writing JES2 exits that are in the following situations:
   - Run outside the JES2 main task
   - Need to access or update checkpoint control blocks
   you need to follow the specific coding recommendations in "Checkpoint control blocks" on page 382 and those specific "Programming Considerations" listed for the JES2 exit that you are implementing.

$JCT/JMR relationship

The MVS Job Management Record (JMR) is initialized as part of the JES2 $JCT when the $JCT is built by HASPRDR.

Additionally, the following information should help in the understanding of the $JCT and JMR relationship:

- SMF documentation references to the Common Exit Parameter Area (CEPA) which is actually the MVS JMR.
- During the Conversion, Execution, and Purge phases of JES2, the JMR is built by copying the JMR section of the JES2 $JCT into the MVS JMR.
- At the end of the Conversion and Execution phases of JES2, the MVS JMR is copied back into the $JCT. Any alterations to the JMR is therefore checkpointed along the JES2 $JCT.
- The CEPA User-Communication field (defined as JMRUCOM in the JMR) could be used to provide addressability to the JES2 $JCT for SMF exits.
- There is a MVS Job Control Table (JCT). It’s built by MVS and used during execution by MVS and has nothing to do with the JES2 JCT.

The following table, [Table 31 on page 387](#), displays a side-by-side label comparison of the JMR (CEPA) and the JES2 $JCT/JMR areas.
Table 31. $JCT/JMR Definitions

<table>
<thead>
<tr>
<th>$JCT Label</th>
<th>JMR Label</th>
<th>Length</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCTJMRJN</td>
<td>JMRJOB</td>
<td>8 characters</td>
<td>8-character job name from JOB JCL statement</td>
</tr>
<tr>
<td>JCTRDRON</td>
<td>JMRENTRY</td>
<td>4 bytes</td>
<td>Time, in hundreds of second, on Input processor</td>
</tr>
<tr>
<td>JCTRDTON</td>
<td>JMREDATE</td>
<td>4 bytes</td>
<td>Date on Input processor in form of 00YYDDDF</td>
</tr>
<tr>
<td>JCTCPUID</td>
<td>JMRCPUID</td>
<td>4 bytes</td>
<td>SMF SYSID</td>
</tr>
<tr>
<td>JCTUSEID</td>
<td>JMRUSEID</td>
<td>8 characters</td>
<td>Initialized to blanks by JES2</td>
</tr>
<tr>
<td>JCTSTEP</td>
<td>JMRSTEP</td>
<td>1 byte</td>
<td>Current step number</td>
</tr>
<tr>
<td>JCTINDC</td>
<td>JMRFLG</td>
<td>1 byte</td>
<td>SMF options</td>
</tr>
<tr>
<td>JCTJTCC</td>
<td>JMRCLASS</td>
<td>2 bytes</td>
<td>Byte 1 is condition code and second byte is execution job class</td>
</tr>
<tr>
<td>JCTCLASS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTUCOM</td>
<td>JMRUCOM</td>
<td>4 bytes</td>
<td>User communication area - initialized to zeros by JES2</td>
</tr>
<tr>
<td>JCTUJVP</td>
<td>JMRUTLP</td>
<td>4 bytes</td>
<td>User time limit exit routine</td>
</tr>
<tr>
<td>JCTRDRF</td>
<td>JMRDRSTP</td>
<td>8 bytes</td>
<td>First word is time off input process and second word is date off input process</td>
</tr>
<tr>
<td>JCTRDTOF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JCTJOBIN</td>
<td>JMRJOBIN</td>
<td>4 bytes</td>
<td>Job’s SYSIN count</td>
</tr>
<tr>
<td>JCTRDR</td>
<td>JMRDR</td>
<td>2 bytes</td>
<td>Reader device type and class</td>
</tr>
<tr>
<td>JCTJMOPT</td>
<td>JMOPT</td>
<td>1 byte</td>
<td>SMF option switches</td>
</tr>
<tr>
<td>(none)</td>
<td>(none)</td>
<td>1 byte</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Input phase

The JES2 input service exits provide the functions needed to receive all pre-execution batch jobs, started tasks, and time sharing sessions into the system. There are special cases, as outlined in "Job input sources," where some (non-batch) jobs bypass input service.

Many installations use input service exits to control installation standards, tailor accounting information, and provide additional security controls.

Job input sources

Figure 14 on page 388 shows The possible sources of jobs entered into JES2. Each of the input sources (known internally as devices) is represented by a Processor Control Element ($PCE) and a Device Control Table ($DCT). The $PCE is the dispatchable element used by the JES2 dispatcher and the $DCT contains the device (input source) information.
When designing input service exits, be aware that jobs can be entered from a number of input sources. Consider whether the source of a job could affect the exit processing. For example, in the case of a spool offload job receiver, an individual job could be submitted more than once. This could be an important consideration if the purpose of the exit is to add a JCL or JECL statement. A test for a spool offload device ($DCT) may be in order to see if the additional statement already exists. Also, some exit-provided functions may not apply to all job sources. For example, you might want to bypass started tasks or time sharing sessions when enforcing installations standards. When using spool offload to selectively reload jobs, Exits 2-3-4 will be taken even for jobs that are not selected. This is because the work selection takes place after the JCL has been received.

There are jobs ($JQEs) that do not originate through input service, for example, the system log ($SYSLOG), the JES2 trace facility ($TRCLOG), and remote message spooling (SRMTMSG) that are created internally and do not have JCL associated with them. Additionally, there are jobs created for NJE and spool offload SYSOUT receivers and NJE store-and-forward jobs. These are also specially created jobs that do not go through input service and therefore input service exits are not taken for these special jobs.

**Job input service processing**

The following scenarios describe the exits and the sequence of exits for a normal batch job entered through either main task or user environment of JES2 input service.

**Table 32. Job Input Service Exits - Main Task.** This applies to physical card readers, remote readers, spool offload devices, and SNA/BSC NJE devices.

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If the job source is a NJE job receiver or a spool offload job receiver (reload), Exit 47, the NJE header exit, is processed before Exit 2. For all other job sources, Exit 2 will be the first exit to be taken.</td>
<td>47</td>
</tr>
</tbody>
</table>

*Figure 14. Job Input Sources*
Table 32. Job Input Service Exits - Main Task (continued). This applies to physical card readers, remote readers, spool offload devices, and SNA/BSC NJE devices.

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>A job statement is read and the $JCT is initialized. Exit 2 has control before the actual scanning of the job statement. You can set the job defaults, the spools allowed mask (fencing), and the job exit mask (to prevent certain future exits to be taken). You may also control the message class of a job at this time. The job statement has not been processed. To control or override statement parameters, change either the actual parameter in the buffer or, choose a later exit to alter field in the control block after the job statement scan is complete. For each JOB continuation statement, an additional Exit 2 is taken with a value of 4 in general register 0.</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>After the job and job continuation statements have been processed, a spool track is obtained using $TRACK and Exit 11.</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>An $IOT is initialized, and the spool control blocks ($JCT and $IOT) are written to spool. Exit 7 is taken.</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>Exit 3 processes accounting information. The job statement has already been written to the spool JCL data set. Therefore, it is too late to alter the accounting information passed to the MVS Converter. To alter accounting information, use HASPRSCAN.</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Exit 4 processes submitted JCL, JCL continuation, and JES control statements (JECL). JCL residing in PROCLIB is not processed. To process all JCL, use SMF exit IEFUJV or Exit 6. Exit 4 processes all JECL (*), with the exception of internal reader control statements (such as /*EOF, /*DEL).</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Exit 2 is taken. After Exit 2, the NJE header validation routine is taken to verify the structure of the network job trailer and indicate the end of the job.</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>If the input device is an NJE Job Receiver, Exit 47 is taken for the network job trailer. Exit 47 can be used to: • Reject the job (and hold it at the transmitting node) • Accept the job (and add or remove sections of the NJE header).</td>
<td>47</td>
</tr>
<tr>
<td>9</td>
<td>After all the submitted JCL and JECL have been processed for a job, SAF calls are made to verify the job. Six additional SAF calls are made to process system generated spool data sets (joblog, job messages, JCL, and so on.). For each SAF call, Exits 36, 37 are taken. The SAF router exit (ICHRTX00) is also taken.</td>
<td>36 37 ICHRTX00</td>
</tr>
</tbody>
</table>
Table 32. Job Input Service Exits - Main Task (continued). This applies to physical card readers, remote readers, spool offload devices, and SNA/BSC NJE devices.

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>After all of the job’s submitted JCL and JECL have been processed, and end of file (EOF) condition causes control to be passed to the end of job processing, Exit 20 is taken. Exit 20 allows final changes to the job without the exposure of further job JCL and JECL alterations. The final write of the $JCT and $IOT to spool follows Exit 20. The $JQE has not been checkpointed so you can make changes affecting the $JQE. You can make changes to job class and job priority and JES2 will propagate the changes to the $JQE. To change other fields, such as JQEJNAME which require the alteration of the $JQE, use the $DGQJQE service to obtain an update mode JQE. When the updates are complete, use the $DGQJQE service to return the updated JQE.</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>Exit 7 is taken again when the $JCT and $IOT are written to spool. Exit 7 could be used to create an installation defined spool-resident control block. The headers are kept in separate SPOOL buffers with their address pointers in the $JCT. The $JCTX macro extension service allows you to add, expand, locate, and delete $JCT extensions. These extensions can be used to store job-related accounting information that can be copied throughout a network.</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>The $JQE is moved from the input queue to the conversion queue and checkpointed. If an error occurs, the $JQE is placed on the output queue or purge queue and checkpointed. Exit 51 is taken when the job moves on from one queue to the next.</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 33. Job Input Service Exits - User Environment. This applies to internal readers (batch, STC, and TSU), and TCP/IP NJE job receivers.

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If the job source is a NJE job receiver or a spool offload job receiver (reload), Exit 57, the NJE header exit, is processed before Exit 52. For all other job sources, Exit 52 will be the first exit to be taken.</td>
<td>57</td>
</tr>
<tr>
<td>2</td>
<td>A job statement is read and the $JCT is initialized. Exit 52 has control before the actual scanning of the job statement. You can set the job defaults, the spools allowed mask (fencing), and the job exit mask (to prevent certain future exits to be taken). You may also control the message class of a job at this time. The job statement has not been processed. To control or override statement parameters, change either the actual parameter in the buffer or, choose a later exit to alter field in the control block after the job statement scan is complete. For each JOB continuation statement, an additional Exit 52 is taken with a value of 4 in general register 0</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>After the job and job continuation statements have been processed, a spool track is obtained using $TRACK and Exit 12.</td>
<td>12</td>
</tr>
</tbody>
</table>
Table 33. Job Input Service Exits - User Environment (continued). This applies to internal readers (batch, STC, and TSU), and TCP/IP NJE job receivers.

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>An $IOT is initialized, and the spool control blocks ($JCT and $IOT) are written to spool. Exit 8 is taken.</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Exit 53 processes accounting information. The job statement has already been written to the spool JCL data set. Therefore, it is too late to alter the accounting information passed to the MVS Converter. Use HASPRSC.</td>
<td>53</td>
</tr>
<tr>
<td>6</td>
<td>Exit 54 processes submitted JCL, JCL continuation, and JES control statements (JECL). JCL residing in PROCLIB is not processed. To process all JCL, use SMF exit IEFUJV or Exit 6. Exit 54 processes all JECL (/*), with the exception of internal reader control statements (such as /*EOF, /*DEL).</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>Exit 52 is taken. After Exit 52, the NJE header validation routine is taken to verify the structure of the network job trailer and indicate the end of the job.</td>
<td>52</td>
</tr>
<tr>
<td>8</td>
<td>If the input device is an NJE Job Receiver, Exit 57 is taken for the network job trailer. Exit 57 can be used to:</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>- Reject the job (and hold it at the transmitting node)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Accept the job (and add or remove sections of the NJE header).</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>After all the submitted JCL and JECL have been processed for a job, SAF calls are made to verify the job. Six additional SAF calls are made to process system generated spool data sets (joblog, job messages, JCL, and so on.). For each SAF call, Exits 36, 37 are taken. The SAF router exit (ICHRTX00) is also taken.</td>
<td>36, 37, ICHRTX00</td>
</tr>
<tr>
<td>10</td>
<td>After all of the job’s submitted JCL and JECL have been processed, and end of file (EOF) condition causes control to be passed to the end of job processing, Exit 50 is taken. Exit 50 allows final changes to the job without the exposure of further job JCL and JECL alterations. The final write of the $JCT and $IOT to spool follows Exit 50.</td>
<td>50</td>
</tr>
<tr>
<td>11</td>
<td>Exit 8 is taken again when the $JCT and $IOT are written to spool. Exit 8 could be used to create an installation defined spool-resident control block. The headers are kept in separate SPOOL buffers with their address pointers in the $JCT.</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>The $JCTX macro extension service allows you to add, expand, locate, and delete $JCT extensions. These extensions can be used to store job-related accounting information that can be copied throughout a network.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>The $JQE is moved from the input queue to the conversion queue and checkpointed. If an error occurs, the $JQE is placed on the output queue or purge queue and checkpointed. Exit 51 is taken when the job moves on from one queue to the next.</td>
<td>51</td>
</tr>
</tbody>
</table>
Conversion phase

The conversion phase of JES2 processing is accomplished in two environments. First the Converter Processor Control Element ($PCE) is dispatched in the JES2 maintask environment to select a job from the input queue. Secondly, the Converter subtask, after being posted by the Converter maintask, calls the MVS Converter to do the actual conversion (JCL to C/I text). The reason for the subtask environment is that the conversion process requires the reading of the JCL data set from spool, reading JCL from PROCLIB, writing JCL images to spool, and the writing of C/I text to spool. These I/O operations cannot be accomplished in the maintask environment.

It's important to understand the difference in these two environments when considering exit usage. Exit 7 executes in the maintask environment, and Exit 6, and the SMF IEFUJV exit, execute in the subtask environment. If maintask functions are needed for a subtask exit, it may be necessary to use two exits, for example Exits 6 and 44 in conjunction, to provide a specific function.

Another important consideration is that there can be (and typically are) more than one converter processor (and subtask) and therefore, any exits taken in the subtask (Exits 6 and exit, IEFUJV) must be MVS reentrant. The following scenario describes the processing that occurs during the conversion processing.

Table 34. Conversion Phase Processing

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A job is selected from the input queue, and the job’s $JCT is read from spool. Exit 7 is invoked with a value of zero in general register zero (R0=0). The Daughter Task Element (DTE) is initialized and the Converter subtask is POSTed.</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>The JES2 conversion subtask locates the job’s $PDDBs (JES2 Peripheral Data Definition Blocks) and Fake Opens the ACBs (Access Control Blocks) for internal text, job log, system messages, JCL, and JCL images data sets. The Converter subtask LOADs the MVS Converter, if the Converter has not already been loaded. Exit 8 is taken for reading the $IOTs from spool.</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>The Security Access Service ($SEAS) macro calls the Security Authorization Facility (SAF) to build the security environment in case the jobstream contains MVS commands which if present, would be issued by the Converter using the Command SVC. The userid associated with the command would be the user’s, not JES2. As a result of the $SEAS call, Exits 36 and 37 are called.</td>
<td>36 37</td>
</tr>
<tr>
<td>4</td>
<td>For each JCL image, SMF exit IEFUJV (entry codes 0, 4, 8, and 64) is taken. This includes continuation statements. IEFUJV is called once more with an entry code of 16.</td>
<td>SMF exit IEFUJV</td>
</tr>
<tr>
<td>5</td>
<td>After the statement and all continuation statements have been converted into C/I text, the Converter exit, XTXTEXT is called to provide spool data set names for SYSIN and SYSOUT JCL statements. If the statement represents a SYSIN data set, a $SEAS call is made to audit the creation.</td>
<td>XTXTEXT</td>
</tr>
<tr>
<td>6</td>
<td>After the spool data set names have been generated (if SYSIN or SYSOUT) Exit 6 is invoked (R0=0) with the completed C/I text statement as input to the exit.</td>
<td>6</td>
</tr>
</tbody>
</table>
Table 34. Conversion Phase Processing (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>At the completion of conversion and after the Converter returns to the JES2 converter processor module, a $SEAS call is issued to delete the security environment. Exit 6 (R0=4) is taken again to allow final processing.</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>As a result of the $SEAS call, Exits 36 and 37 are called.</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>Exit 8 is taken to write the $IOTs. The JES2 converter processor module subtask POSTs its maintask and WAITs for the next job.</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Exit 44 is taken to allow user modifications that require the maintask environment. Using the $DOGJQE macro you can access and optionally update fields in the JQE.</td>
<td>44</td>
</tr>
<tr>
<td>11</td>
<td>The JES2 converter module checkpoints the $JCT and invokes Exit 7.</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>The $JQE is queued to the execution queue and Exit 51 is invoked.</td>
<td>51</td>
</tr>
</tbody>
</table>

The conversion phase offers the only chance to have exit control over all of a job’s JCL. Although SMF exit, IEFUJV is taken for each JCL and JCL continuation statement, JES2 Exit 6 offers some advantages.

First, the format of the C/I text is more structured. It is in parsed form and all major syntax errors have been removed. This has all been done by the converter before the exit gets control.

Another advantage of Exit 6 over IEFUJV is that when JCL statements have been converted into C/I text, there are no continuation statements. That is, the entire JCL statement, along with all continuation statements, are represented by a single C/I text statement.

A SAF security environment exists within the subtask and can be used with the RACF FACILITY class to control the specification of options within JCL. Exit 6, messages can be returned to the Converter to be issued by the Converter.

Execution phase

This section attempts to merge those functions provided by a section of JES2 code in the JES2 Job Select/Termination module known as “Job Selection” and the pieces of MVS code known in the broad sense as “The Initiator”. The MVS Initiator consists of many modules which perform job selection, allocation, and initiator attach services (and others). JES2 Job Select also includes end-of-job functions.

For the purpose of this discussion, job selection is defined as the period, starting with the initiator’s Subsystem Interface (SSI) call for job selection by class and ends with the JES2 message, SHASP373 JOB STARTED. The following scenario describes the processing that occurs during the Execution Phase.
### Table 35. Execution Phase Exits

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The MVS Job Selection module issues a SSI call specifying function code 5 which identifies the call to JES2 as a request to select a job by class. SSI calls with a function code of 5 are processed by the JES2 Job Select/Termination module. JBSELECT POSTs JES2 execution processing and WAITs for a job to be selected. If a JES2 initiator is selecting work, JES2 calls Exit 14 to allow your installation to provide its own queue selection routine or to tailor the selection request. Exit 14 is not a job-related exit, that is, JES2 has not selected a job at this time. Exit 14 can select a job or it can tell JES2 to select a job. If a WLM initiator is selecting work, JES2 does not call Exit 14. After JES2 selects a job from the execution queue, it calls Exit 49 which can accept or reject the job. If Exit 49 rejects the job, JES2 searches for another job. JES2 does not call Exit 49 if Exit 14 selects a job. If JES2 execution processing finds a job that matches the Initiator's defined job classes, it POSTs the waiting initiator and provides the job's $JCT spool address in the $SJB. If a job has been found, control is given to the JBFOUND routine.</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>The JBFOUND routine reads the job's JES2 $JCT using the spool address passed in the $SJB. Exit 8 is the first exit taken out of the user's (or job's) address space after a job is selected. This first entry to Exit 8 is taken after the job's $JCT has been read. The job name, jobID, and all the other information in the $JCT are available. If later SMF exits for this job need addressability to the JES2 $JCT, store the JES2 $JCT address (as contained in Exit 8 parameter list) into the JCTUCOM field that later becomes the JMRUCOM.</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Exit 8 is again taken to read the primary allocation $IOT. There may also be additional calls to Exit 8 to read secondary allocation $IOTs or $PDDB-only $IOTs based on the job's JCL. Exit 8 is called for all spool control block reads and writes. JES2 allows installations to create extensions to the $JCT where job-related accounting data can be stored and transmitted through the network. Using the $JCTX macro extension service, you can add, expand, locate, and delete these extensions. For more information about using these extensions, see <a href="https://www.ibm.com/support/docview.wss?uid=swg21426171">z/OS JES2 Macros</a>.</td>
<td>8</td>
</tr>
</tbody>
</table>
Table 35. Execution Phase Exits  (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>The JBFOUND routine calls the MVS SWA Create Control module to obtain storage for and initialize the Interpreter Entry List. The Interpreter Entry List contains information from JES2, such as user ID and security information and is used for linking to the MVS Interpreter. Both JES2 and MVS have a data area named JCT. The two JCTs are not similar and one is not a copy, or partial copy, of the other. The Interpreter Entry List contains a pointer to the in-storage copy of the beginning of the $JCT JMR area which is used to create the CEPA/JMR. The MVS Interpreter Initialization routine calls the MVS Interpreter Router routine and after the internal text has been interpreted, the MVS Enqueue routine issues the call to SMF exit IEFUJV (entry code of 32). This is the first SMF exit for a job during the execution phase. The Scheduler Work Area (SWA) job and step tables have been created. The JMR pointer, called the CEPA in SMF documentation, is provided in the exit parameter list.</td>
<td>IEFUJV</td>
</tr>
<tr>
<td>5</td>
<td>After the Interpreter returns control to the MVS SWA Create Control module, a RACROUTE REQUEST=VERIFY, ENV=CREATE is then issued to create the job’s security environment. The SAF Router exit is invoked if it exists and Message ICH70001I is issued by RACF identifying the user. If an error occurred during Job Select processing, for example a JCL error, the job’s security environment is not created.</td>
<td>SAF Router exit</td>
</tr>
<tr>
<td>6</td>
<td>Exit 32 is called. The $JCT, all $IOTs the JMR, and the ACEE have been created and are available.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>The JBSELECT routine then issues the $HASP373 JOB STARTED message.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Before job select processing is complete and control returns to the Initiator, JES2 checkpoints (writes to spool) the $JCT. Exit 8 is called.</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Job initiation calls SMF exit, IEFUJI. MVS job initiation is a series of calls to step initiation based on the number of steps in a job.</td>
<td>IEFUJI</td>
</tr>
<tr>
<td>9</td>
<td>MVS step initiation consists of a call to SMF exit, IEFUSI, step allocation for those data sets and devices defined in the job’s JCL, and a call to the MVS Initiator Attach routine.</td>
<td>IEFUSI</td>
</tr>
<tr>
<td>10</td>
<td>Allocation of JCL defined SYSIN, SYSOUT, and internal readers initiates a call to Exit 31.</td>
<td>31</td>
</tr>
<tr>
<td>11</td>
<td>The MVS Initiator Attach routine attaches a subtask with an entry point of the program name specified on the EXEC JCL statement for the job step. The job step could dynamically allocate JES2 SYSIN, SYSOUT, or internal readers and therefore Exit 31 can be called.</td>
<td>31</td>
</tr>
<tr>
<td>12</td>
<td>The OPEN and CLOSE of JES2 data sets and internal readers call Exits 30 and 33.</td>
<td>30 33</td>
</tr>
</tbody>
</table>
Table 35. Execution Phase Exits (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Dynamic Unallocation of JES2 data sets and internal readers initiate a call to Exit 34. Exit 48 can be used in preference to Exit 34. Exit 34 may be too early to affect some fields in the $PDDB because unallocation processing takes place after Exit 34. Use Exit 48 when altering fields in the $PDDB, this exit can also be used to control Spin processing.</td>
<td>34</td>
</tr>
<tr>
<td>14</td>
<td>At End-of-Task (EOT) processing an SSI call is made to JES2 and Exit 35 is called.</td>
<td>35</td>
</tr>
<tr>
<td>15</td>
<td>Control is passed (return from Attach) to the MVS Initiator Attach routine and subsequently MVS Step Delete calls Step Unallocation which unallocates those data sets and devices defined in the job's JCL on a step basis. Exit 34 is called for JCL defined SYSIN, SYSOUT, and internal readers. Exit 48 is also taken as mentioned previously.</td>
<td>34, 48</td>
</tr>
<tr>
<td>16</td>
<td>The MVS Unallocation routine calls the MVS SMF Control routine which calls SMF exit IEFAXRT with entry codes 20 and 12. If additional job steps are to be processed, control is passed back to step 8. Otherwise, control is passed to Job Termination at step 17.</td>
<td>SMF exit IEFAXRT</td>
</tr>
<tr>
<td>17</td>
<td>Job Termination (actually this is Step Termination for the last step) again calls SMF exit IEFAXRT with entry codes 20 and 16. Control is then passed to MVS Step Delete where a SSI call (12) is made for Job Termination.</td>
<td>IEFAXRT</td>
</tr>
<tr>
<td>18</td>
<td>End-of-job processing calls Exit 28. This exit can clean up resources obtained over the life of job execution.</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>Spool control blocks are checkpointed. Exit 8 is taken for writing the JCT.</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>The $JQE is placed on the OUTPUT queue waiting output processing, and Exit 51 is invoked.</td>
<td>51</td>
</tr>
</tbody>
</table>

Spin phase

Spin processing typically takes place during the execution phase, however because of processing alternatives, which could occur during execution, the spin phase could happen immediately after the execution phase, but always before the output phase. Spin processing consists of processing the unspun queue and building Job Out Elements ($JOEs) for each unspun spool data set.

The output phase follows the spin phase processing and is sometimes confused with the hardcopy phase. Output phase processing scans the job's $IOT chains and if there are $PDDBs representing non-held output, these $PDDBs will be grouped into $JOEs. Held output data sets are grouped into $JOEs which are the elements representing output groups (spool data sets with like characteristics). $JOEs are queued by class in the Job Output Table ($JOT) and are ordered FIFO, within priority, by route code.

After all $PDDBs have been assigned output groups the job's $JQE is placed on the hardcopy queue to await print, punch, transmission, or canceling of job output. The following describes the Spin Phase processing.
Table 36. Spin Phase Processing

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>After selecting a job from the $SPIN queue, the spin processor scans through the $IOTs which represent unspun data sets. When an unspun $IOT is found, Exit 40 gains control to allow the installation to change the characteristics of the data set before grouping the data set into an output group ($JOE).</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>A $#BLD macro is issued to build a $JOE and a $#ADD macro is issued to add the $JOE to the $JOT.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>The $QMOD macro queues the job ($JQE) to the OUTPUT queue for processing, and Exit 51 is invoked.</td>
<td>51</td>
</tr>
</tbody>
</table>

Output phase

The following describes the Output Phase processing.

Table 37. Output Phase Processing

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The $QGET service searches the job queue to find a candidate for output processing. Exit 14 ($QGET) is taken before a job is selected so this is not a job-related exit.</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Because there can be multiple output processors, the job lock ($GETLOCK) provides serialization on a job basis. When the lock is obtained, the $JQE is checkpointed using the $CKPT macro.</td>
<td>$CKPT</td>
</tr>
<tr>
<td>3</td>
<td>After the job is selected and the job lock obtained, the job’s $JCT is read from spool and Exit 7 is called.</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>If NOTIFY= was coded on the JOB JCL statement, NOTIFY processing calls Exit 16. This exit, is conditionally based on the job’s JCL parameter.</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>After NOTIFY processing, the job’s $IOTs are read from spool, $PDDBs are scanned, and the non-HELD $PDDBs are assigned to $JOEs. HELD $PDDBs are also assigned to $JOEs. $JOEs represent output groups, an output group can represent one or more spool data sets with like characteristics. Before each data set is grouped, Exit 40 is taken for each data set. Any changes made to the $PDDB will be used to determine data set grouping. Use Exit 40 to change SYSOUT characteristics. Exit 40 is taken before the data set has been gathered into an output group ($JOE). After all non-HELD PDDBs are processed, the $JCT is checkpointed. This is done to update the spool-resident $JCT with alterations made during output processing.</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>After the $JCT is checkpointed, the job’s $JQE is moved to the hardcopy queue to await printing or other processing of job output. The $JQE is checkpointed after being moved to the hardcopy queue. Exit 51 is invoked when the job moves to the hardcopy queue.</td>
<td>51</td>
</tr>
</tbody>
</table>
Hardcopy phase

The hardcopy phase of JES2 processing takes place after output processing. The job’s $JQE is placed on the hardcopy queue where it waits until all output is processed.

To be processed, HELD data sets must be either released, canceled, or transmitted (SPOOL Offload or NJE). All data sets are grouped into $JOEs. However, held data sets are not eligible for hardcopy processing even though they are represented by $JOEs. Since $JOEs are always resident in memory, the performance of held data sets is improved.

A common misconception with JES2 users is that output is assigned to a printer or output device. Output is only assigned to an output class and has other output characteristics. Output devices, printers, punches, external writers, and so forth, select job output from the output queues ($JOT or Job Output Table) by class and other output characteristics. Output has no affinity to an output device, for example, a printer. Output must be selected by the device based on the output data set characteristics matching the device work selection (WS=) criteria. Route code is the most common characteristic used to match job output with an output device.

This section discusses two types of hardcopy processing, JES2 controlled devices and Print Services Facility™ (PSF) controlled devices. The JES2 Print/Punch Processor module contains the necessary functional routines for controlling and writing to JES2 output devices, both local and remote.

Only line mode printing is supported for JES2 devices. Page mode output data must be processed by PSF. Printing to coax connected printers (printers attached through 3174 and so on.), such as 3270 type printers (3276, ...), is not controlled by JES2. Applications, such as JES/328X, are required to support these types of printers.

The following describes the Hardcopy Phase processing.

Table 38. Hardcopy Phase Processing

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HASPPRPU initialization consists of assigning an available output device and initializing control blocks and buffers as a result of a Start command (e.g., $S PRT(5)).</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>When an output device (either remote or local) has been started a call is made to scan the output queues $JOT using the $#GET macro. This is the work selection service which scans the $JOT to search for output as specified in the work selection parameter list. When an output group ($JOE) has been selected the job’s $JCT is read from spool and Exit 7 is taken.</td>
<td>7</td>
</tr>
</tbody>
</table>
### Hardcopy Phase Processing (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>If the image subtask has not already been attached, it is done now. A call is made for Exit 1 to allow installations to provide their own separator routine. After Exit 1 (and based on Exit 1 if it exists) the standard JES2 supplied separator page may be produced. The jobs $IOTs are read from spool and the $PDDBs (contained within the $IOTs) are obtained. Setup is called to check if device and data set characteristics match. Operator intervention may occur here.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>A call is made ($SEAS) to verify that the data set userid (owner) is allowed to print on this device. Exits 36 and 37 are taken.</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>Exit 15 (R0=0) is called for data set select. This exit point could be used to control copy count, print translate table, or the CCW translate tables.</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Exit 15 (R0=4) is again called to allow user produced data set separators. The $#CHK macro is used to produce a checkpoint at this time. A checkpoint produces a checkpoint $JOE that allows for recovery in case of a system or device failure.</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>The main print/punch loop is where SPOOL buffers are read, channel programs are constructed for the output device, and $EXCPs are issued to print or punch lines of output. This process continues until the entire data set is read and written to the output device. The data set is repeated if copy count is greater than one and a return to step 3 is made if there are additional data sets in the output group to be processed.</td>
<td>There are no exits available during this process.</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Exit 1 is called (R0=8) to allow for installation separator routines to replace the JES2 routine. The $JOE is placed on the free queue. When there are no more output data sets to be processed for the job, the $JQE is placed on the Purge queue. Exit 51 is invoked when the job moves to the purge queue.</td>
<td>1</td>
<td>51</td>
</tr>
</tbody>
</table>

### NJE hardcopy phase exits

The following describes the NJE Hardcopy Phase processing:

#### Table 39. NJE Hardcopy Phase Processing

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Network SYSOUT Transmitter initializes a SYSOUT Transmitter device ($DCT) and acquires resources (lines, buffers, and so on.) to prepare for SYSOUT transmissions. The $#GET service routine is used to search the Job Output Table ($JOT) to find an eligible $JOE on the network queue. When a candidate is found the $CBIO macro is used to read the $JCT, $IOTs and $SWBITs from spool. Exit 7 or 8 is taken for each control block read. If the network job header does not exist, the NJE SYSOUT transmitter builds it.</td>
<td>Exit 7 (JES2 main task), Exit 8 (TCP/IP NJE)</td>
</tr>
</tbody>
</table>
### Purge phase

The purge phase is the final phase of JES2 processing. Jobs are placed on the purge queue after all spool data set have been processed or if the job gets canceled. Spool tracks are returned, the SMF 26 record is written and the $JQE is placed on the free queue. The following scenario describes the processing that occurs during the Purge Phase.

#### Table 40. Purge Phase Exits

<table>
<thead>
<tr>
<th>Step</th>
<th>Processing</th>
<th>Exit Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A job is selected from the purge queue, the $JCT is read and Exit 7 is invoked.</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>$PURGE macro calls the purge service routine for each spool data set. If data set purge verification is active, the $SEAS macro will be issued for authorization. This invokes Exits 36 and 37 for each purged data set. Spool tracks assigned to the job are returned.</td>
<td>36, 37</td>
</tr>
<tr>
<td>3</td>
<td>Buffers are gotten to build the SMF type 26 record and the JMR. The SMF 26 record is formatted. $QUESMFB macro calls the SMB buffer queue routine Exit 21 is called and a $POSTQ is issued to POST the HASPACCT (SMF Writer) subtask. Because $QPOST was issued, we do not WAIT on the completion of the SMF write. $QUESMFB returns to HASVPRG immediately.</td>
<td>21</td>
</tr>
</tbody>
</table>
Exit 7 could possibly be used as a general purpose exit. Exit 21 and SMF exit IEFUJP are taken after the return of spool tracks. When IEFUJP is invoked, the in-storage buffer containing the $JCT could be reused and contain another job’s $JCT.
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](http://www.ibm.com/systems/z/os/zos/bkserv/) and [z/OS ISPF User’s Guide Vol I](http://www.ibm.com/systems/z/os/zos/bkserv/) for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

**z/OS information**

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS DOCUMENT “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 document. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this document 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 want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs...
and other programs (including this one) and (ii) the mutual use of the information
which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and condition
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.

This information contains examples of data and reports used in daily business
operations. To illustrate them as completely as possible, the examples include the
names of individuals, companies, brands, and products. All of these names are
fictitious and any similarity to the names and addresses used by an actual business
enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which
illustrates programming techniques on various operating platforms. You may copy,
modify, and distribute these sample programs in any form without payment to IBM,
for the purposes of developing, using, marketing or distributing application programs
conforming to the application programming interface for the operating platform for
which the sample programs are written. These examples have not been thoroughly
tested under all conditions. IBM, therefore, cannot guarantee or imply reliability,
serviceability, or function of these programs. You may copy, modify, and distribute
these sample programs in any form without payment to IBM for the purposes of
developing, using, marketing, or distributing application programs conforming to
IBM’s application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations
may not appear.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the
customer to write programs to obtain the services of JES2.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS™, HCD, JES2, JES3, and MVS, contain
code that supports specific hardware servers or devices. In some cases, this
device-related element support remains in the product even after the hardware
devices pass their announced End of Service date. z/OS may continue to service
element code; however, it will not provide service related to unsupported hardware
devices. Software problems related to these devices will not be accepted for
service, and current service activity will cease if a problem is determined to be
associated with out-of-support devices. In such cases, fixes will not be issued.
Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.
Index

Special characters
(exit 26) 195
$$WTO macro 19
$$WTOR macro 19
$#CHK macro 399
$CHK 382
$CKPT macro 397
$CRET macro 19
$CWTO macro 19
$D EXIT(nnn) command 51
$DCT 387
$ENTRY macro 21, 28
$ERROR macro 14
$ESTAE macro 24
$EXIT macro 28, 56
MAXRC= operand 14
$FREEBUF macro 77
$FREEBUF 77
$GETBUF 77
$GETBUFI macro 77
$GETSMFB usage 177
$HASP426 message 71
$HASP427 message 71
$HASP428 message 72
$HASP864 message 74, 188
$HASPGBL copying 28
$IOT 382
$JCAN macro 180
$JCT 382
$JCT/JMR 386
$JCTX extension
accounting field 92
exit 1 77, 230, 236
exit 11 137
exit 12 141
exit 15 150
exit 16 156
exit 2 84
exit 20 173, 295
exit 23 184
exit 25 192
exit 28 201
exit 3 92, 314
exit 30 208
exit 32 216
exit 33 220
exit 34 224
exit 35 227
exit 39 212, 246
exit 4 101, 116
exit 40 250
exit 43 263
exit 44 266
exit 46 276
exit 47 282
exit 48 286
exit 52 307
$JCTX extension (continued)
exit 53 314
exit 6 116
exit 7 120
exit 8 123
exit 9 128
$JIB 10
$JOE 10
$MODULEEND macro 28
$MODULE macro 28
$OCT 382
$PBLOCK service routine 77
$SEPPDIR usage 77
$PCE 387
$UESMFB usage 177
$RETURN macro 12, 13
$SAVE macro 12
$SCAN facility 72, 168
$STMTLOG macro 168
$STORE macro 13
$STRAK (exit 12) 139
$SWBIT 382
$T EXIT command 168
$T EXIT(nnn) command 51
$T EXIT(nnn) operator command 24
$TRACE macro 24
$TRACK (exit 11) 135
$USER1 through $USER5 189
$WTO messages, modifying 155
$WTO parameter list usage 155
$WTO screen exit 131
&RJOBOPT usage 93

A
accessibility 403
account field scan 68
accounting field scan exit 87, 310
across environment exits 8
addressability of the exit 21
addressing requirements
$AMODE
  AMODE 13
  31-bit 13
  residency 13
  RMODE 13
affinity
  system 171, 293
allocation 135
  spool partitioning ($STRAK) 139
alter console routing 132
alter SMF control block 177
altering operating states of exits 5
analyzing initialization statements 167
APPC (Advanced Program-to-Program Communication)
  transaction program (TP) 261
areas of modification in JES2 2
assembler language for exits 7
assembly environment
   $MODULE macro  7
assign system affinity  171, 293
automatic tracing  24

B
BSC RJE devices
   controlling 159
BSC RJE signon/signoff exit  159
buffer
   use in Exit 1  77

C
calling environment  7
cancel
   exit 178
cancel status exit  178
CEPA  386, 401
change notify routing  155
changing message text (exit 10)  132
changing output grouping keys 253
changing SYSOUT characteristics
   exit  249
checking initialization statements  167
checkpoint  399
checkpoint control blocks  382
codes  14
   exit-dependent return codes  14
   return (greater than 4)  14
   return codes  14
coding considerations  11
   $ENTRY macro  21
   addressability of the exit  21
   control blocks for exits  14
   exit-dependent return codes  14
   linkage conventions  11
   main task exits  11
   multiple exit routines  12
   naming the exit  21
   nonreentrant  11
   packaging the exit  28, 47
   received parameters  13
   recovery for exits  24
   reentrant  11
   return codes (greater than 4)  14
   return codes for exits  14
   service routine usage  17
   source module conventions  21
   subtask exits  11
   tracing the exit  24
coding language for exits  7
COMAUTH structure  108
command  51
   $D EXIT(nnn)  51
   $T EXIT  168
   operator ($T EXIT(nnn))  24
   preprocessor exit  105
communication
   $CWTO macro (exit 5)  109
communication (continued)
   exit routine-to-exit point
   response byte  18
   exit-to-operator  19
   JES2-to-operator  2
condition byte
   exit point-to-exit routine
   communication  18
CONSOLE initialization statement  168
console message buffer (CMB)  131
   CMBFLAG usage (exit 10)  132
   CMBJOBN usage (exit 10)  132
   CMBROUT usage (exit 10)  132
   CMBTEXT usage (exit 10)  132
   interrogating (exit 10)  131
   usage (exit 16)  155
control block read/write (JES2)  119
   JCTJQE usage  119
   JQETYPE usage  119
   PCEID usage  119
   specific description  119
control block read/write (JES2) exit  119
control block read/write exit  123
control blocks for exits  14
control statement
   /*JOBPARM
      job control field table  92, 314
   /*ROUTE
      job control table field  92, 314
control statement scan  97
   HASPRCCS replacement  97
   recovery  98
   specific description  97
control statements
   /*SETUP
      job control table fields  91, 313
controlling BSC RJE devices  159
   controlling SNA RJE devices  163
converter
   exit 44  265
converter/interpreter text scan 113
   CNVWORK usage  116
   recovery  115
   specific description  113
   Converter/Interpreter text scan exit  68
   Converter/Interpreter text scan exit  111
COPY $HASPGBL  28
create SMF control block  177
creation of installation control blocks  187

D
data set  168
   log data set  168
   separator exit  149
deleting initialization statements  167
device  159
   BSC RJE remote  159
   SNA RJE remote  163
disability  403
disabled exit state  5
disabling the exit 51
DTE 392
dual execution environments 8

E
enabled exit state 5
enabling trace (ID 13) for tracing 24
end of job input exit 171
environments 7
environments for exits 7
caller’s environment 7
execution environment 7
JES2 main task 7
JES2 subtask 7
user address space 7
error 51
isolating them 51
ESTAE 205
  recovery 199, 201, 205, 207, 211, 215, 219, 223, 227, 229, 235
execution environment
  FSS (functional subsystem address space) 7
  JES2 (main task) 7
  SUBTASK (subtask) 7
  USER (user address space) 7
execution node 171, 293
exit 1, 190, 293, 299, 303, 319, 329, 332, 333, 339
$WTO screen 131
  across environments 8
  addressability 21
  BSC RJE signon/signoff 159
  cancel/status 178
  control block read/write 68
  control block usage 14
  Converter/Interpreter text scan 68
  end of job input 171
  IBM-defined 4, 59
  implementation table 68
  individual purposes 59
  initialization JCL 39
  initialization statement scan 167
  initializing in the system 48
  installation-defined 4, 55
  integrating exit routines 47
  introduction 1
JCL/JES2 control statement scan 68
JES2 command preprocessor 68
job queue work select 145, 289
Job Queue Workload Selection (initiator jobs) 68
job separator page process 182
job statement account field scan 68, 87, 310
job statement scan 68
job-related 51
job-related (defined) 5
linkage conventions 11
logic 18
mask (JOBMASK) 51
modifying a notify user message 257
modifying SYSOUT characteristics 249
multiple exit routines 6

exit (continued)
  linkage conventions 12
  naming the exit 21
NJE SYSOUT reception data set disposition 245
notify 155
operating environment 7
output data set/copy separators 149
packaging 47
packaging the code 28
passing control to them 51
PCE attach/detach 197
post initialization 185, 187
pre-initialization 68
pre-initialization (exit 0) 71
pre-security authorization call 229
print/punch job separator 74
print/punch separator 68
received parameters 13
recovery considerations 24
reentrant code considerations 11
return code responsibility 12
return codes 14
service routine usage 17
SMF record 175
SNA RJE logon/logoff 163
source module conventions 21
specific individual uses 59
specific titles of each 59
specific uses 59
spool partitioning allocation ($STRAK) 139
spool partitioning allocation ($TRACK) 135
SSI data set allocation 211
SSI data set CLOSE 219
SSI data set OPEN and restart 207
SSI data set unallocation 223
SSI end-of-memory 203
SSI end-of-task 227
SSI job selection 215
SSI job termination 200
SSI SYSOUT data set unallocation 285
status (enabled, disabled) 51
synchronization 10
termination 193
testing exit routines 47
tracing status 53
tracing their execution 24
TSO/E receive data set disposition 241
using control blocks 14
writing an exit routine 7

Exit 1
$FREEBUF macro 77
$GETBUF macro 77
buffer usage 77

exit 10 131
CMBFLAG usage 132
CMBJOBN usage 132
CMBROUT usage 132
CMBTEXT usage 132

exit 11 135
$TRACKX exit point 136
JCTSAMSK usage 135
exit 12 139
  $STRAKX exit point 140
  JCTSAMSK usage 139
exit 14 145
  finding job queue work 145
exit 15 149
  CCW translate table usage 149
  PRTRANS table 149
exit 16 155
  change notify routing 155
  CMB usage 155
  modify $WTO messages 155
exit 17 159
exit 18 163
  MICEXIT exit point 164
  MSNALXIT exit point 163
  MSNALXT2 exit point 164
exit 19 167
  $SCAN facility usage 168
  $STMTLOG macro 168
  $T EXIT command 168
  EXIT(nnn) usage 168
  LOADmod usage 168
exit 20 171
  JCTIPTIO usage 172, 294
  PCE work area usage 172, 294
exit 21 177
  $GETSMFB usage 177
  $QUESMFB usage 177
exit 22 179
  $JCAN macro 180
  IKJSN2161 message 180
exit 23 183
exit 24 187
  $HASPR864 message 188
  $T EXIT command usage 188
  $USER1 through $USER5 189
  EXITTnnn statement 188
  recovery 187
exit 26 195
exit 27 199
exit 28 201
exit 29 205
exit 3 89
  &RJOBOPT use 93
  exit 3 90
  HASPRSCN replacement 89
  JCTJOBID usage 92
  JCTWORK usage 92
  JCTXWRK usage 93
  recovery 90, 312
exit 30 207
exit 31 211
exit 32 215
exit 33 219
exit 34 223
exit 35 227
exit 36 229
exit 37 235
  post-security authorization call 235
exit 38 241
exit 39 245
exit 4 97
  HASPRCCS replacement 97
  recovery 98
exit 40 249
exit 41 253
exit 42 257
  recovery 257
exit 43 261
exit 44 265
exit 45 269
exit 46 275
exit 47 281
exit 48 285
exit 49 289
exit 5 105
  $CWTO macro 109
  COMAUTH structure 108
  recovery 107
exit 50 293
exit 51 299
exit 52 303
exit 53 311
  exit 53 312
exit 54 319
exit 55 329
exit 56 333
exit 57 339
exit 6 113
  CNVWORK usage 116
  recovery 115
exit 7 119
  JCTJQE usage 119
  JQETYPE usage 119
  PCEID usage 119
Exit 9 127
exit effector 7
  definition 6
  tracing 56
exit facility
  introduction 1
  using 3
exit implementation table 68
exit module 21
  security considerations 22
  source conventions 21
exit point 3
  $STRAKX (Exit 12) 140
  $TRACKX (exit 11) 136
  definition 3
  identifying them 3
  logoff 164
  logon 163
  MICEXIT (exit 18) 164
  MSNALXIT (exit 18) 163
  MSNALXT2 (exit 18) 164
exit routine 3
  definition 3
  integration 47
  language used 7
exit routine (continued)

load module 48
loading one 39
multiple ones 6, 45
passing them control 51
placement 50
writing one 7
exit selection table 59
exit-to-exit communication
among exits
exit point-to-exit routine condition byte 18
EXIT(nnn) initialization parameter 24
exits 117
control block read/write 122
Converter/Interpreter text scan 111
execution phase 393
hardcopy phase 398
JCL/JES2 control statement scan 97
JES2 command preprocessor 104
Job Input Service 388
job-related 379
output phase 397
purge phase 400
sequence 379
spin phase 396
exits in processing order 59
external names 48

F
FSACB 10
FSS environment 9
FSSCB 10

G
generic grouping
modifying selection with an exit 253

H
hardcopy
console 168
HASJES20 19, 28
location 8
HASPCOMM 19
HASPINIT 8, 28
HASPIRPL 167
HASPRDR 386

I
I/O 1
control block 119
IBM-defined exits 4
description 59
identifying the exit 21
IEFACTRT 396
IEFUJI 395
IKJ56216I message 180
implementation
exit table 68
implementing initialization statements 167
initialization 2
$ADD LOADmod(jxxxxxx) command 48
$T EXIT(nnn) 49
&RJOBOPT use 93
EXIT(nnn) parameter 24
EXIT(nnn) statement 3, 49
EXIT(nnn) TRACE= usage 53
exits in the system 48
JCL 39
LOADMOD statement 3
LOADMOD(jxxxxxx) initialization statement 48
modifying control blocks 187
placement of exits 50
pre-initialization exit 71
processing 2
initialization statement exit 167
$SCAN facility usage 168
$STMTLOG macro 168
$T EXIT command 168
checking and analyzing 167
CONSOLE 168
CONSOLE initialization statement 168
EXIT(nnn) 168
EXIT(nnn) usage 168
implementing 167
LOADmod 168
LOADmod usage 168
tailoring 167
initialization statement scan exit 167
initializing a user defined exit 39
initializing an exit 39
initializing the exit in the system 48
initiator jobs 68
work selection exit 289
input/output
See I/O
inserting initialization statements 167
installation 4
control blocks 187
exits 4
work areas 189
installation-defined exits 55
integrating the exit routine 47
interrogate CMB 131
introduction
checkpoint control blocks 382
job-related exits
exit sequence 379
selected exits 379
job-related Exits 379
spool control blocks 382
IOT 1
isolating an exit error 51

J
JCL (job control language) 1
initializing an exit 39

Index 413
JCL/JES2 control statement scan exit 68, 97
JCT 379
JCT (job control table) 1
   JCTIPTIO usage 172, 294
   JCTJOBID usage 92
   JCTJQE usage 119
   JCTSAMSK usage (Exit 11) 135
   JCTSAMSK usage (exit 12) 139
   JCTWORK usage 92
   job control table 1
   job exit mask address 51
   read/write 119
   selected fields 90, 312
JCT read 190
   exit 25 191
   recovery 191
JCT read/write exit 68
JES 2 Print /Punch processor 398
JES2 1
   $ESTAE macro usage 24
   $SCAN facility 72
   address space 9
   areas of modification 2
   dispatching unit (PCE) 11
   exit 3
   exit effector 7
   main task 8
   modifying 1
   primary load module (HASJES20) 8
   processors 11
   reentrant sense 11
   source language (assembler) 7
   subtasks execution 9
   terminating 179
JES2 command preprocessor exit 68, 105
JES2 converter exit (JES2 main) 265
JES2 exits
   exit 1 399
   exit 11 379
   exit 12 379
   exit 14 397
   exit 15 399
   exit 16 397
   exit 21 400
   exit 28 396
   exit 30 395
   exit 31 395
   exit 32 395
   exit 33 395
   exit 34 396
   exit 35 396
   exit 36 399, 400
   exit 37 399, 400
   exit 51 396
   exit 7 397, 398, 400
   exit 8 394, 395, 396
   exit 9 379
JES2 Exits
   exit 2 379
   exit 6 379
JES2 Exits (continued)
   exit 7 379
   exit 8 379
JES2 main
   converter exit 265
JES2 main task 8
JES2 reentrancy 8
JES2 subtask 9
JES2-to-operator communication 2
JMR 1
   SMFTYPE field
      meaning 178
      values 178
   usage 94
job 2
   end of input exit 171
   exit mask (JOBMASK) 51
   input processing 2
   priority 171, 293
   related exits (defined) 5
   terminating processing 171, 293
job control language (JCL)
   See JCL (job control language)
job control table
   read write (USER) exit 123
   job control table field 92, 314
job exit mask 52
job input 171
   end 171
   processing 2
job management record
   SMFTYPE field
      meaning 178
      values 178
job management record (JMR)
   See JMR
job output
   processing 2
job queue 145
   finding work (exit 14) 145
   work select exit 145
job queue element
   See JQE (job queue element)
job queue initiator jobs 68
job queue work select exit 145, 289
job separator page process 182
job statement account field scan exit 68, 87, 310
JOB statement accounting field scan 87
&RJOBOPT use 93
HASPRSCN replacement 89
JCTJOBID usage 92
JCTWORK usage 92
JCTXWRK usage 93
recovery 90, 312
specific description 87, 310
job statement scan exit 68
general description 60
job termination 171, 293
job-related exits 51
JOBMASK parameter 51
passing control to exit routines 51
PCE 1
  PCEID usage 119
  work area for HASPRDR 172, 294
PCE attach/detach exit 197
PCEs 397
phases
  conversion 392
    overview 392
  execution
    exits 393
    overview 393
  hardcopy
    exits 398
    overview 398
input
  exits 388
  overview 387
output
  exits 397
  overview 396
purge
  exits 400
  overview 400
spin
  exits 396
  overview 396
placement of exits 50
post initialization exit 185, 187
post-security authorization call exit 235
pre-initialization 71
  $HASP426 message 71
  $HASP428 message 72
  $HASP864 message 74
  LOAD macro 72
  specific description 71
pre-initialization exit 68
  $HASP426 message 59
  $HASP427 message 59
  general description 59
pre-security authorization call exit 229
pre-SFJ service request exit 269
print/punch 74
  $SEPPDIR usage 77
  specific description 74
print/punch job separator exit 74
  general description 59
print/punch separator exit 68
priority 171, 293
processing 51
  disabled exits 51
  enabled exits 51
  job-related exits 51
processing area, exit arrangement 59
processor control element
  See PCE
processors invoking exits 59
programming considerations (continued)
  exit logic 18
  exit-to-operator communication 19
  integrating the exit routine 47
  multiple exit routines 12
  naming the exit 21
  other ones for exits 21
  packaging the exit 28, 47
  passing control to exit routines 51
  recovery for exits 24
  security 22
  service routine usage 17
  source module conventions 21
  testing exit routines 47
  tracing status of exits 21
  tracing the exit 24
Q
  queue SMF records 177
R
  received parameters for exits 13
  record, job management
    SMFTYPE field
      meaning 178
      values 178
Recovery 205
  exit 24 187
  exit 27 199
  exit 28 201
  exit 29 205
  exit 30 207
  exit 31 211
  exit 32 215
  exit 33 219
  exit 34 223
  exit 35 227
  exit 36 229
  exit 37 235
  exit 38 241
  exit 4 98
  exit 42 257
  exit 5 107
  exit 6 115
recovery for exits 24
reentrant
  JES2 sense 11
  MVS sense 11
reentrant considerations for exits 11
register
  linkage information for exits 11
remote attribute table (RAT) 160
  usage (exit 17) 160
  usage (exit 18) 164
remote job entry (RJE) 3
  BSC signon/signoff exit 159
  processing 3
  SNA logon/logoff exit 163
replacing initialization statements 167
specific description (continued)
$TRACKX exit point 136
$USER1 through $USER5 189
CCW translate table usage 149
change notify routine 155
CMB usage 155
CMBFLAG usage 132
CMBJOBN usage 132
CMBROUT usage 132
CMBTEXT usage 132
COMAUTH structure 108
EXITn exit statement 188
finding job queue work 145
IKJ56216I message 180
JCTIPTIO usage 172, 294
JCTSAMS usage 135, 139
MICEXIT exit point 164
modify $WTO messages 155
MSNALXIT exit point 163
MSNALXT2 exit point 164
PCE work area usage 172, 294
PRTRANS table 149
recovery 107, 187, 191
specific description 185, 187, 190
specific uses of exits 59
spool 2
partitioning allocation ($STRAK) 139
partitioning allocation ($STRAK) exit 139
partitioning mask (JCTSAMS) 135
processing 2
spool control blocks 382
spool partitioning allocation ($STRAK) exit 139
spool partitioning allocation exit ($STRACK) 135
SSI data set allocation exit 211
SSI data set CLOSE exit 219
SSI data set OPEN and restart exit 207
SSI data set unallocation exit 223
SSI end-of-memory exit
(JES2) 203
SSI end-of-memory JES2 exit 203
SSI end-of-task exit 215
SSI job termination exit
(JES2) 200
SSI job termination JES2 exit 200
SSI SYSOUT data set unallocation exit 285
status
changing exit status 51
exit 178
exit status 51
tracing exit status 53
subtask 6
protect key 6
subtask environment 9
synchronization services 10
for exits 10
main task
SWAIT macro 10
SYSOUT characteristics
exit to change 249
system affinity 171, 293
system initializing for exits 48
system management facilities (SMF)
  record exit 175

T
  tables 149
    CCW translate table (exit 15) 149
    exit implementation table 68
    exit selection table 59
    PRTRANS table (exit 15) 149
tailoring initialization statements 167
  termination exit 193, 195
  termination JES2 exit 193
testing 24
    exit routines 47
    tracing usage 24
    TYPE=TEST ($EXIT macro) 55

TGB 1
titles of exits 59
tracing 24
  $D EXIT(nnn) command usage 53
  $T EXIT(nnn) command usage 53
  automatic tracing 56
  automatically 24
  AUTOTR= ($EXIT macro) 56
  disabled (exit 19) 168
  enabling trace (ID 13) 24, 53
  exit effectors 56
  exit status 53
  exits 24
  job-related tracing 53
  necessary conditions 24
  TRACE= usage on EXIT(nnn) 53
tracing status of exits 53
transaction program (TP)
  selection/change/termination exit 261
TSO/E CANCEL/STATUS (exit 22) 179
TSO/E receive data set disposition exit 241

U
  use of exit facility 3
  user address 6
    protect key 6
  user address space environment 7
User Control Table 189
  usage 189
USER environment 9
using control blocks in exits 14
using service routines in exits 17

V
  verify a job's existence 92

W
  weak external names 48
Readers’ Comments — We’d Like to Hear from You

z/OS
JES2 Installation Exits

Publication No. SA22-7534-11

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via e-mail to: mhvrdfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address