z/OS

z/OS MVS Using the Functional Subsystem Interface

Version 2 Release 1

SA38-0678-00

z/OS

z/OS MVS Using the Functional Subsystem Interface

Version 2 Release 1

SA38-0678-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 147.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this information

This book is intended for anyone responsible for writing and installing a functional subsystem (FSS) and its functional subsystem applications (FSA). This book describes the functional subsystem interface (FSI) and shows how the FSS and a job entry subsystem (JES) communicate using the FSI.

How this information is organized

The organization and content of each section are:

- Chapter 1, "Functional Subsystem Interface Concepts," on page 1, briefly describes functional subsystem concepts, terminology, address space relationships, and services that the functional subsystem interface supplies.
- Chapter 2, "An Overview of FSI Processing," on page 7, describes the overall flow of processing from the time the FSS is started, through data set processing, until the FSS is terminated.
- Chapter 3, "Installing a Functional Subsystem," on page 11, provides examples of JES initialization statements needed to install a functional subsystem and a sample JCL procedure.
- Chapter 4, "The FSIREQ Macro," on page 13, presents the FSIREQ macro and explanations of the macro parameters.
- Chapter 5, "FSI Communication," on page 17, describes the communication mechanisms that allow JES to make service requests to the FSS or FSA and allows the FSS or FSA to respond to JES.
- Chapter 6, "Establishing FSS/JES Communication," on page 29, describes the processing for starting the functional subsystem.
- Chapter 7, "Establishing FSA/JES Communication," on page 37, describes the processing for starting a functional subsystem application that is associated with an individual device.
- Chapter 8, "Starting an FSS Device," on page 47, describes the processing for starting a device that runs under an FSS.
- Chapter 9, "Issuing Data Requests to JES," on page 51, describes how to obtain and free a data set and its records, and how to ask JES to record checkpoint information.
- Chapter 10, "Responding to Device Orders From JES," on page 89, describes the processing for orders that request a change in device or data set characteristics, affects the flow of data through the device, or requests information about a data set currently being processed by an FSA device.
- Chapter 11, "Stopping an FSS Device," on page 103, describes the processing involved in stopping a device that is running under an FSS.
- Chapter 12, "Stopping an FSA," on page 107, describes the processing for stopping a functional subsystem application that is associated with an individual device.
- Chapter 13, "Stopping an FSS," on page 113, describes the processing for terminating the functional subsystem address space.
- Chapter 14, "FSS Output Descriptor Support," on page 117, describes the scheduler JCL facility and how it interfaces with JES and the FSS to provide FSS scheduler work block support.

- Chapter 15, “FSI Trace,” on page 123, describes FSI trace facilities useful for diagnosing problems with the FSI.
- Appendix A, “FSIREQ Parameter List,” on page 133, contains storage representations for the fields in the IAZFSIP mapping macro and other related storage.
- Appendix B, “Numeric Values of FSI Services and Orders,” on page 141, provides the numeric values for the FSI services.
- “Notices” on page 147, provides notices, programming interface information, and trademark information.
- A Glossary and Index are also provided.

How to use this information

Use Chapter 1, “Functional Subsystem Interface Concepts,” on page 1, and Chapter 2, “An Overview of FSI Processing,” on page 7, to familiarize yourself with the terminology and processing related to the functional subsystem interface.

Use Chapter 3, “Installing a Functional Subsystem,” on page 11, to install the functional subsystem.

Use Chapter 4, “The FSIREQ Macro,” on page 13, through Chapter 14, “FSS Output Descriptor Support,” on page 117, when you are coding your functional subsystem and your functional subsystem applications. These chapters explain how to use the FSI to make requests to JES and explains how JES will respond to those requests. These sections explain the values that JES expects to receive from your functional subsystem during processing. These sections also show the values that your functional subsystem can expect to receive from JES when your FSS receives control.

Use Chapter 15, “FSI Trace,” on page 123, to diagnose any problems your FSS might encounter.

Where to find more information

For the titles and order numbers of other documents referenced in this document, see *z/OS Information Roadmap*.

How to send your comments to IBM

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

Use one of the following methods to send your comments:

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

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS MVS Using the Functional Subsystem Interface
  SA38-0678-00
- The topic and page number that is related to your comment.
- The text of your comment.

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

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

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Functional Subsystem Interface Concepts

The functional subsystem interface (FSI) allows communication between JES and your functional subsystem (FSS) and functional subsystem application (FSA). The FSS/FSA allows installations to support sophisticated devices.

Besides using an FSS to drive sophisticated printers, the FSS can also drive a simple device, or a process that is not a device at all. This chapter defines the key concepts related to the functional subsystem interface.

What is a Functional Subsystem?

A functional subsystem (FSS) is a collection of programs residing in an address space separate from JES that communicates with JES to provide a JES-related function, such as print processing. An FSS extends the scope of JES processing. Because an FSS operates in its own address space, it functions independently of JES in several areas.

An FSS is responsible for:

- The management of storage resources that it needs during data set processing including print buffers.
- Its own recovery and serviceability.
- Its performance and accounting measurements.
- The security of its own resources.

Managing an FSS

An FSS is dependent on JES for control and services. JES manages an FSS in the following ways:

- The FSS is defined during JES initialization using JES initialization statements and parameters.
- JES initiates the FSS address space.
- JES provides services for use by the FSS. FSS messages sent to the JES operator are in a format chosen by the writer of the FSS.
- JES controls its own resources, such as the job queues and spool.
- JES controls output scheduling for FSS-controlled devices. The FSS application does not control selection criteria when acquiring data sets for print processing. JES uses its own work selection criteria to provide the proper data sets to the FSS application.
- JES coordinates the termination and restart of the FSS.

What is a Functional Subsystem Application?

A functional subsystem application (FSA) is a collection of programs residing in the FSS address space that control one device. There can be multiple FSAs per FSS. IBM recommends that each of the FSAs for the FSS be a separate task. The FSA can be thought of as a logical subset of the FSS and is the lowest level of connection with JES.

What is the Functional Subsystem Interface?

JES and the FSS/FSA communicate through the functional subsystem interface (FSI). The FSI is a one-level interface which provides two way communication. The FSI consists of a set of macro-invoked service routines provided by both JES and the FSS/FSA. These service routines are:

- JES routines that reside in the FSS address space
- SSI routines that JES provides
- FSS/FSA-supplied routines.

Figure 1 shows the types of address space communication (SSI,XM, and FSI) that exist between **JES2** and the FSS. Figure 2 on page 3 shows the types of address space communication (SSI and FSI) that exist between **JES3** and the FSS.

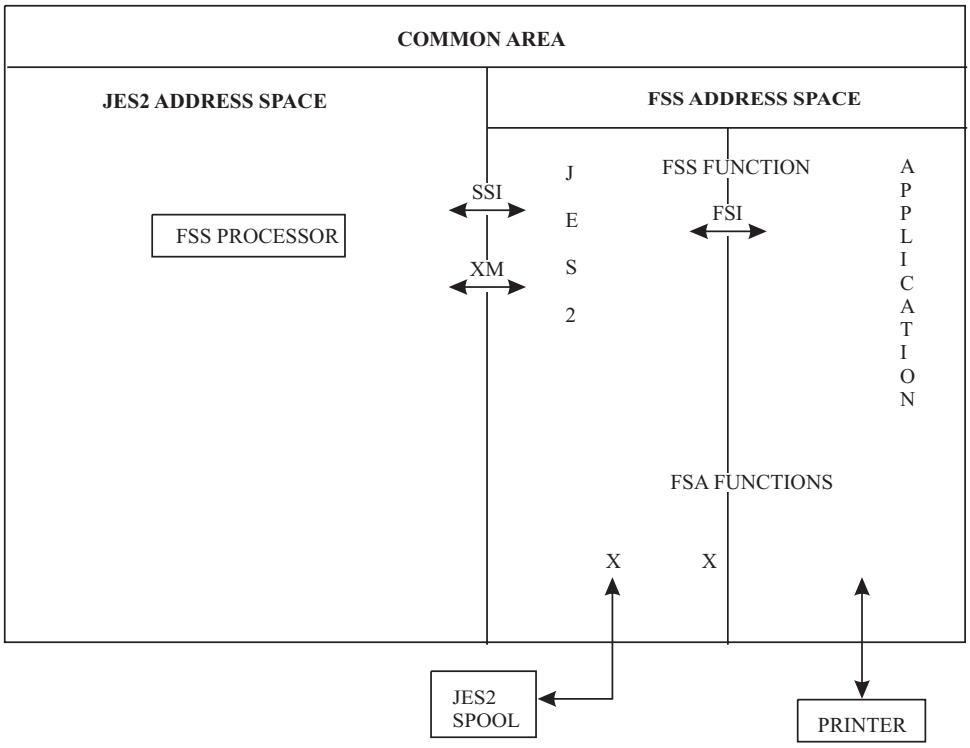

Figure 1. Address Space Communication Between JES2 and the FSS

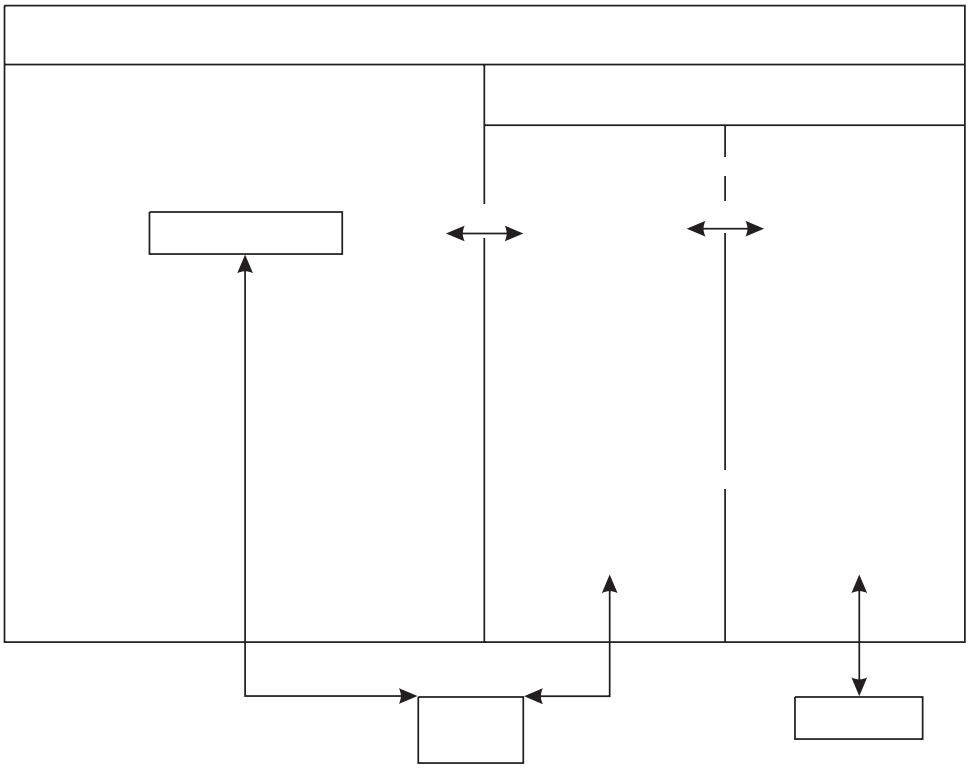

Figure 2. Address Space Communication Between JES3 and the FSS

Invoking the FSI

The FSS/FSA and JES use the FSIREQ macro to invoke functional subsystem interface (FSI) services. The FSIREQ macro allows JES to issue orders to the FSS/FSA and the FSS/FSA to issue requests to JES.

The IAZFSIP mapping macro maps the FSIREQ function-dependent parameter lists. The FSS/FSA and JES use these parameter lists to pass information to each other. On the FSIREQ macro call, the caller specifies the service requested, the subsystem (FSS or JES) that provides the service, and the address of the caller's parameter list. Chapter 4, "The FSIREQ Macro," on page 13 describes each operand on the FSIREQ macro and Appendix A, "FSIREQ Parameter List," on page 133 illustrates the storage maps for the associated FSIREQ parameter lists.

FSI Services

FSI services are actually JES and FSS/FSA supplied routines that allow interaction between JES and the FSS/FSA. FSI services fall into three categories:

- Communication services
- Data access services
- Control services.

Communication Services

The functions of the individual FSI communication services are:

- FSI CONNECT

  The FSS and FSA invoke the FSI CONNECT service to establish the functional subsystem interface to JES. FSI CONNECT processing tells JES that the FSS/FSA

is started. It also identifies to the FSI the addresses of FSS/FSA routines that are to receive control when JES issues the FSIREQ macro and the addresses of JES routines that are to receive control when the FSS/FSA issues the FSIREQ macro.

- FSI DISCONNECT

  The FSS and FSA invoke the FSI DISCONNECT service to terminate connection with JES.

- FSI ORDER

  JES invokes the FSI ORDER service to issue orders to the FSS/FSA. When an operator issues a JES command that requires the participation of an FSS/FSA, JES converts that command into an order. An order represents a unit of work known to both JES and the FSS/FSA. The FSS/FSA performs the actions associated with the order and then responds to JES with the required information. The valid orders are:

  Start FSA
  Requests the FSS to start the FSA. When the FSA is started, the FSA responds to JES with the FSA CONNECT request.

  Start Device
  Requests the FSA to start the device. Once the device is started, the FSA can begin requesting data sets for processing.

  Stop Device
  Requests the FSA to stop the device. Once the FSA stops the device, it does not request any more work.

  Stop FSA
  Requests the FSS to stop the FSA. When the FSA completes its processing, it responds to JES with an FSA DISCONNECT request.

  Stop FSS
  Requests the FSS to shut down. When the FSS completes its processing, it responds to JES with an FSS DISCONNECT request.

  Query
  Requests the FSA to obtain information about the data set currently at the operator observation point (OOP).

  Set
  Requests the FSA to set or change device characteristics.

  Synch
  Requests the FSA to synchronize its processing to the point of actual printing. JES issues a synch order when an action needs to be performed against the data set currently at the operator observation point (OOP) of the device.

  Intervention
  Requests the FSA to prepare the device for operator intervention. JES issues this order when a change in device setup (such as a change in forms) that involves operator intervention is required.

- FSI SEND

  The FSS/FSA invokes the FSI SEND service to send an asynchronous response to a JES order. The FSS/FSA can also use the SEND to send unsolicited material to the JES.

Data Access Services

The functions of the individual FSI data access services are:

- FSI GETDS

The FSA invokes the FSI GETDS service to request access to a JES spool data set and its characteristics. The GETDS service is functionally equivalent to allocating and opening a data set.

- FSI GETREC

  The FSA invokes the FSI GETREC service to obtain one or more records from a data set obtained by use of the FSI GETDS service.

- FSI FREEREC

  The FSA invokes the FSI FREEREC service to free one or more logical records that it previously acquired with a GETREC request.

- FSI RELDS

  The FSA invokes the FSI RELDS service to release a data set previously obtained by the FSI GETDS service. The RELDS service is functionally equivalent to closing and unallocating a data set.

- FSI CHKPT

  The FSA invokes the FSI CHKPT service to request JES to record checkpoint information for the JES spool data set currently being processed on the FSA device.

  The checkpoint information recorded is used for restart situations. For example, if processing of the data set is interrupted, the FSA returns the data set to JES with an incomplete processing status. When the data set is again selected for processing, the device can begin printing the data set from the point of the last valid checkpoint taken.

Control Services

The FSI POST service is the only FSI control service. JES invokes the FSI POST service to signal completion of asynchronous requests. If no work is available to satisfy a GETDS request, JES returns control to the FSA indicating it will satisfy the request at a later time. When work becomes available, JES issues an FSI POST request to notify the FSA that GETDS requests can now be satisfied and that the FSA should reissue the request.

The following table lists each FSI service and shows the type of interaction it allows, the valid caller(s) of the service, and the subsystem/application that provides the service.

FSI Service	Type of Interaction	Used by	Provided by
CHKPT	Data Access	FSA	JES
CONNECT	Communication	FSS/FSA	JES
DISCONNECT	Communication	FSS/FSA	JES
FREEREC	Data Access	FSA	JES
GETDS	Data Access	FSA	JES
GETREC	Data Access	FSA	JES
ORDER	Communication	JES	FSS/FSA
POST	Control	JES	FSA
RELDS	Data Access	FSA	JES
SEND	Communication	FSS/FSA	JES

FSS Interface Example

IBM provides a functional subsystem (FSS) interface example in *SYS1.SAMPLIB*. The example is a working illustration of how you might implement functional subsystem interface (FSI) functions. This example is meant as a starting point for applications programmers to develop their own FSS applications, which can include functions such as driving output devices, for example, plotters and microfiche writers, or other devices.

The example is written in S/370 basic assembler language. Its component members, as well as a brief description of each, are listed in Table 1 below.

Table 1. FSS Interface Example Components

Member Name	Description
IAZSFSS	Module that operates the (main) FSS task. This module contains the entry point for the load module. It also contains general documentation for the FSS interface example.
IAZSFSA	Module that operates the FSA subtask(s).
IAZSFSD	Module that operates an output device.
IAZSFSJ	The module that facilitates communication between JES and the FSS or an FSA.
IAZSFSE	Module that provides recovery for the FSS and FSA tasks.
IAZSSCB IAZSDTE IAZSACB IAZSDCB IAZSPLE IAZSSMF IAZSMSG IAZSOPT	Macros that describe data areas specific to the FSS interface example.

In addition to providing an illustration of basic FSI functions, the example also provides function and/or logic that may or may not be appropriate for your general-purpose FSS. These functions appear throughout the code and are denoted in comment boxes that contain the following special header:

```
* <<<<<<<<<<<< SUPPLEMENTARY INFORMATION >>>>>>>>>>>>>>>>>>> *
```

As appropriate for your application, you should delete and/or modify these additional functions or logic.

The example is meant only as a documentation and coding shell for the general flow for JES/FSS processing and does **not**:

- Use all of the functions of the FSI
- Fully complete, in every case, a function that was initiated. (For example, you might receive console messages indicating a function completed, when in fact, it did not actually perform the function.)
- Show a definitive way to develop an FSS
- Attempt to match the requirements of any particular installation-written FSS.

Chapter 2. An Overview of FSI Processing

Functional subsystem interface (FSI) processing consists of three major consecutive stages: FSS startup, data set processing, and FSS termination. Figure 3 illustrates the logical processing steps within each stage of FSI processing and shows the flow of control between JES and the FSS for each step.

FSS Startup

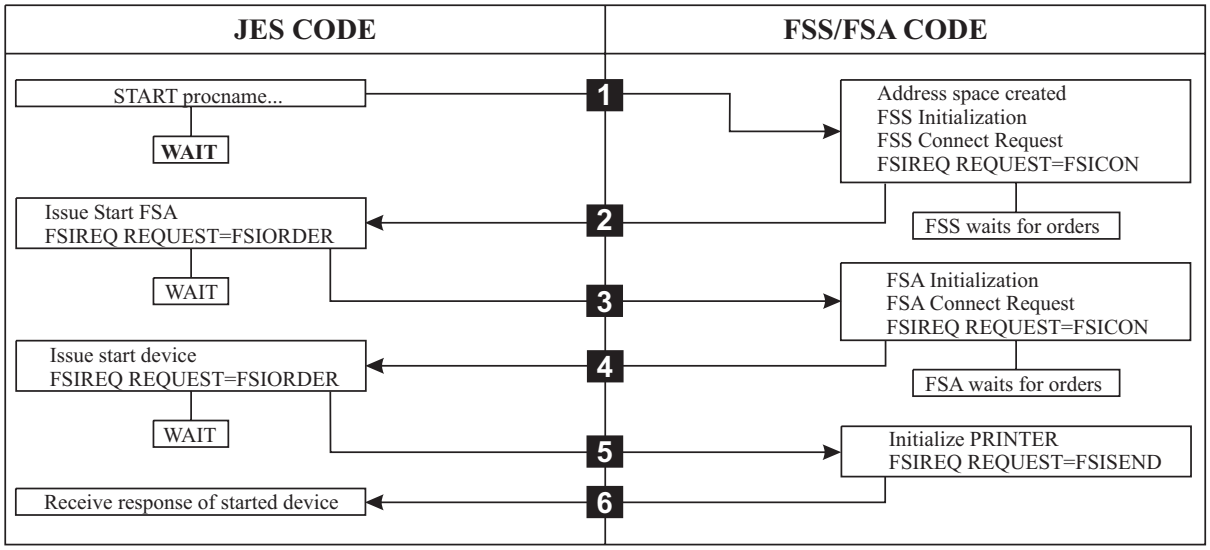

Figure 3. Overview of FSS Startup Processing

Figure 3 shows how JES starts an FSS, one or more FSAs associated with the FSS, and the FSA printer.

1. JES starts the FSS either during JES initialization or in response to an operator command to start a printer under control of the FSS. JES gets information from the FSS initialization statement to use in the MVS START command. JES then issues the MVS START command causing the creation of the FSS address space.
2. Once the FSS address space is created, the FSS performs initialization. When initialization is complete, the FSS responds to JES with an FSIREQ CONNECT request. Successful completion of FSS CONNECT processing signals JES to issue a START FSA order.
3. JES issues the START FSA order to the FSS order routine.
4. The FSS order routine receives the order and then the FSS attaches an FSA task to perform FSA and device initialization. When FSA initialization is complete, the FSA responds to JES with an FSIREQ CONNECT request. Successful completion of FSA CONNECT processing signals JES to issue a START DEVICE order.
5. JES issues the START DEVICE order to the FSA order routine.
6. The START DEVICE order indicates to the FSA that JES is ready to receive GETDS requests. The FSIREQ SEND request notifies JES that the FSA has completed the order. At this point, the FSA can issue GETDS requests.

FSI Data Set Processing

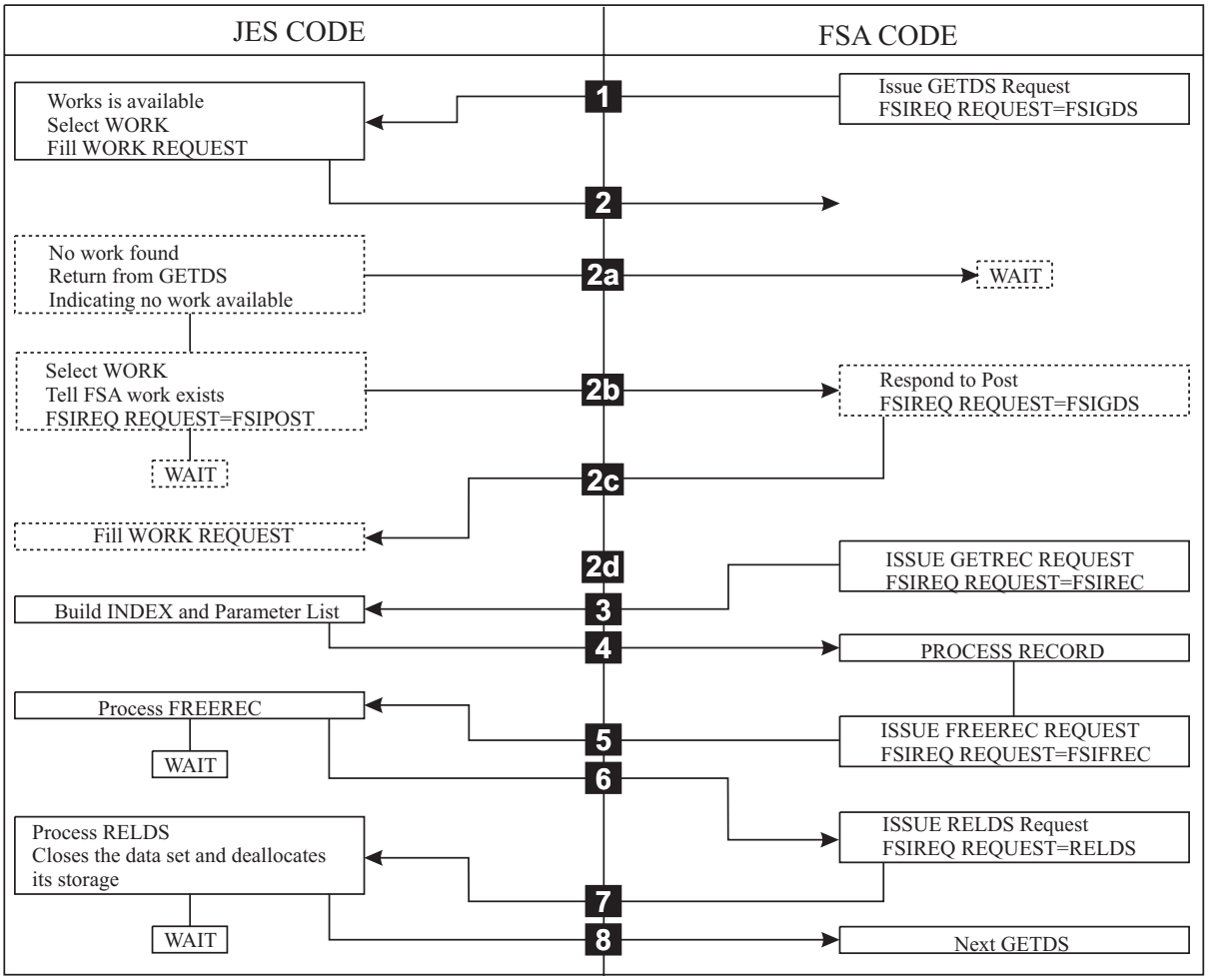

Figure 4. Overview of FSI Data Set Processing

Once the device is started, the FSA can begin issuing data requests to JES. Figure 4 shows the FSI data set processing steps.

1. The FSA issues an FSIREQ GETDS request to JES to obtain a JES spool data set and its attributes for processing. The GETDS service is functionally equivalent to allocating and opening a data set.
2. If work is available, JES immediately satisfies the GETDS request. JES assigns a data set to the FSA and returns data set related information in the GETDS parameter list.
   a. If no work exists for print processing, JES returns control to the FSA. The FSA will not issue GETDS requests until JES notifies the FSA by using the FSI POST service. JES will issue an FSIREQ POST request when JES determines that work has become available.
   b. When work becomes available, JES issues a FSIREQ POST request.
   c. The FSA POST routine gets control and posts the FSA task to reissue the GETDS request.
   d. When JES receives the GETDS request, JES satisfies the GETDS request as was described above.

3. Once the FSA has obtained access to a SYSOUT data set, it uses the data set identifier returned to issue a FSIREQ GETREC request to JES to obtain logical records for the data set.
4. When JES receives the GETREC request, it obtains one or more logical record pointers using an index table. JES then returns a pointer to the index in the GETREC parameter list to the FSA.
5. The FSA processes the records associated with the index and then issues a FSIREQ FREEREC request to release the storage associated with these logical records. Storage resources are a fixed quantity. It is important that the FSA issue FREEREC requests or record processing may eventually not be able to continue because of a buffer shortage.
6. JES processes the FREEREC request by releasing the storage for the specified record. This storage is then available for subsequent GETREC processing.
7. After all of the records in a data set have been processed or when end-of-file is reached, the FSA issues a FSIREQ RELDS request to return the data set to JES.
8. When JES receives the RELDS request, it closes the data set and deallocates the storage resources associated with it. If the FSA indicated that valid checkpoint information exists for the data set, JES writes the final checkpoint record to spool. JES then waits for the next GETDS request.

FSS Shutdown

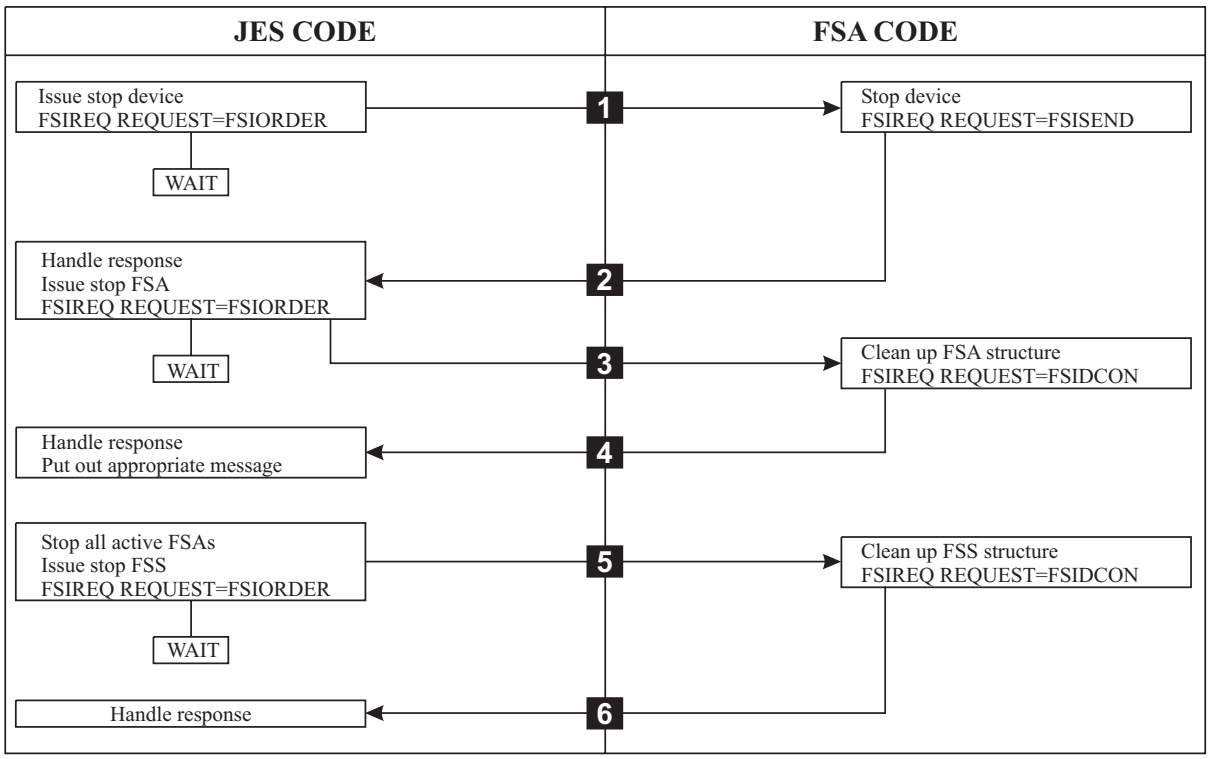

Figure 5. Overview of FSI Shutdown Processing

Figure 5 shows how JES stops a printer, the FSA associated with that printer, and the corresponding FSS.

1. When an operator issues a command to either drain a specific device or to shut down JES cleanly, JES issues a STOP device order to the FSA order routine for the FSA controlling that device.

2. The FSA order routine processes the order and the appropriate FSA task stops the printer device. When the printer is stopped, the FSA must issue an FSIREQ SEND request to notify JES that the FSA has completed the order and that the printer is stopped. At this point, JES can pass another order to the FSA.
3. After the FSA notifies JES that a device was stopped, JES issues a STOP FSA order to the FSS order routine.
4. The STOP FSA order causes the FSA to perform cleanup processing and then terminate itself by issuing an FSA-level DISCONNECT to JES. When JES receives the FSA-level DISCONNECT it validates the information and then issues a message the operator.
5. An FSS receives a STOP FSA order for every active FSA that it controls. After all active FSAs are stopped, JES issues the STOP FSS order to the FSS order routine.
6. The STOP FSS order causes the FSS to perform cleanup processing and then terminate itself by issuing an FSS-level DISCONNECT to JES. JES validates the FSS information and terminates the FSS address space.

Chapter 3. Installing a Functional Subsystem

To install an FSS, you need to:

- Code FSS-related initialization statements and include these in your JES initialization stream.
- Define a JCL procedure that will be used to start the FSS.

The following sections contain examples of the required initialization statements and an example JCL procedure.

FSS-Related Initialization Statements

Both JES2 and JES3 have FSS-related initialization statements that you use to define the FSS requirements and install the FSS.

JES2 FSS-Related Initialization Statements

JES2 has two FSS-related initialization statements. These are:

Statement
Function

PRTnnnn
Defines each FSS printer device to JES2.

FSSDEF
Defines the characteristics of an FSS to JES2.

The following example shows how you might code the above statements to define an FSS and an IBM 3820 printer device running under that FSS.

```
PRT1000 FSS=MYFSS,MODE=FSS,CKPTPAGE=100,PAGECKPT

FSSDEF FSSNAME=MYFSS,PROC=SAMPPROC,HASPFSSM=HASPFSSM
```

Refer to *z/OS JES2 Initialization and Tuning Reference* for more information about each of these statements and the parameters defined on them.

JES3 FSS-Related Initialization Statements

JES3 has three FSS-related initialization statements. Which ones you use depends on the type of printer that will run under the FSS. The three JES3 statements are:

Statement
Function

DEVICE
Defines each FSS printer device to JES3.

SETNAME
Specifies all user-assigned names and device type names associated with MDS-managed devices (for example, 3800 printers). MDS-managed devices are those devices which JES3 allocates, instead of MVS.

FSSDEF
Defines the characteristics of an FSS to JES3.

Refer to *z/OS JES3 Initialization and Tuning Reference* for more information about each of these statements. The following example shows possible initialization statements for defining an FSS to control two IBM 3820 printer devices.

```
DEVICE,DTYPE=PRT3820,JNAME=P2G18,MODE=FSS,FSSNAME=MYFSS,JUNIT=(,SY1,D1,ON)
DEVICE,DTYPE=PRT3820,JNAME=P2X43,MODE=FSS,FSSNAME=MYFSS,JUNIT=(,SY1,D2,ON)
```

Defining JCL Procedure Used to Start an FSS

The following is an example of a JCL procedure used to define requirements for a printing device running under an FSS.

```
//SAMPPROC  PROC
//STEP01    EXEC PGM=MYFSS,REGION=1750K
```

Chapter 4. The FSIREQ Macro

The FSIREQ macro enables communication to be established between JES and the FSS/FSA. The following types of communication can be established by invoking the FSIREQ macro.

- Connect the FSS/FSA to JES (CONNECT)
- Disconnect the FSS/FSA from JES (DISCONNECT)
- Get a SYSOUT data set from JES (GETDS)
- Get records for a SYSOUT data set (GETREC)
- Release records for a SYSOUT data set (FREEREC)
- Release a SYSOUT data set (RELDS)
- Write checkpoint information to spool (CHKPT)
- Send a response to JES (SEND)
- Notify the FSA that a request was completed (POST)
- Send an order to the FSS/FSA (ORDER)

The FSIREQ function dependent parameter lists are mapped by the IAZFSIP mapping macro. The FSS/FSA and JES use the FSIREQ parameter lists to pass information.

In addition to the information in the IAZFSIP macro, other information is passed in additional parameter lists pointed to by the IAZFSIP. The appendix describes these parameter lists and their relationship to the IAZFSIP macro.

This section describes the parameters on the FSIREQ macro and explains the rules for executing the macro. The specific values that the FSS and JES assign are discussed in the chapter specific to the task being performed.

FSIREQ Macro Format

The format of the FSIREQ macro is:

```
                      {FSICON}
                      {FSIDCON }
                      {FSIGDS  }
                      {FSIRDS  }
FSIREQ      REQUEST = {FSIGREC }
                      {FSIFREC }
                      {FSICKPT }
                      {FSISEND }
                      {FSIORDER}
                      {FSIPOST }

{     ,TARGET = JES             }
{               FSS             }

{     ,PARM = parm list address }
{             (R1)   }

{    ,FSID = functional subsystem identifier }
{            (R2 - R12)                      }
```

REQUEST =

Specifies the FSI service to be invoked by either JES or the FSS. If you do not specify REQUEST, you must have previously stored one of the following values in the FSIFUNC field of the FSI parameter list (IAZFSIP).

Note: You must specify the REQUEST= parameter for FSICON and FSIDCON requests.

FSICON
The FSI CONNECT service communicates the initiated status of the FSS/FSA to JES and identifies FSI routines supplied by the FSS/FSA.

FSIDCON
The FSI DISCONNECT service communicates the terminated status of the FSS/FSA to JES.

FSIGDS
The FSI GETDS service enables the FSA to get a data set from JES.

FSIRDS
The FSI RELDS service enables the FSA to release a data set to JES.

FSIGREC
The FSI GETREC service enables the FSA to get records from an obtained data set.

FSIFREC
The FSI FREEREC service enables the FSA to free records from an obtained data set.

FSICKPT
The FSI CHKPT service allows the FSA to request JES to record checkpoint information about a data set currently undergoing print processing on an FSS device.

FSISEND
The FSI SEND service enables the FSS/FSA to send a response to JES.

FSIORDER
The FSI ORDER service enables JES to send an order to the FSS/FSA.

FSIPOST
The FSI POST service enables JES to notify the FSA that work is now available and that the GETDS request can be reissued.

TARGET =

Specifies the subsystem whose routines are invoked when the FSIREQ macro is executed. If target is not specified, JES is the default.

JES
Indicates that JES is to receive control. TARGET=JES is only used by the FSS/FSA.

FSS
Indicates that the FSS is to receive control. TARGET=FSS is only used when JES issues the FSIREQ macro for POST and ORDER requests.

PARM=

Specifies the address of the FSI parameter list. This list contains the data that the specified service will use. If PARM is not specified, JES assumes that you have put the address of the FSI parameter list in register 1. IBM recommends

that you save the address of the parameter list somewhere other than register 1. When JES returns control to the FSS the contents of register 1 may be unpredictable.

The RMODE (residency mode) values that are allowed for the FSI parameter lists depend upon the settings of the CDFFL331 and CDFS1A31 bits.

FSID=
Specifies a value that uniquely identifies the FSS/FSA. JES assigns the FSS an identifier of the form xxxx0000, where xxxx is a unique number. JES assigns the FSA an identifier of the form xxxxyyyy, where xxxx corresponds to the controlling FSS identifier, and yyyy is a unique number for each the FSA. If FSID is not specified, you must have previously stored the FSID in the FSIFSID field of the FSI parameter list.

FSIREQ Macro Execution

The FSIREQ macro is used to request all of the FSI services. When JES or your FSS/FSA issues the FSIREQ macro, the FSI services that receive control are actually JES and FSS/FSA supplied routines. Each subsystem or subsystem application identifies the addresses of its FSI routines during CONNECT processing. The definition of each function's input and output parameters is supplied in the IAZFSIP mapping macro.

Note: The FSI routines adhere to standard OS linkage conventions.

The register conventions on entry to all the services are:

Register 1
Contains the address of the parameter list (FSIPARM).

Register 13
Contains the address of a save area provided by the issuer of the FSIREQ macro.

Register 14
Contains the address of the return point.

Register 15
Contains the address of the entry point.

The FSIREQ macro services return the following return codes in register 15:

0 Successful

non-zero
Request failed

Chapter 5. FSI Communication

The communication mechanism of the FSI allows JES to make service requests to the FSS or FSA. The FSS or FSA receives the request, attempts to provide the requested service, and then returns an indication to JES of whether or not the request was successfully processed. In most cases, JES initiates the communication. The case where the FSA initiates the communication (unsolicited FSIREQ SEND function call) is discussed later in this section.

JES tells the FSS/FSA about the service that the FSS/FSA needs to provide through the FSIREQ ORDER function call.

JES sends only one service request (order) at a time to the FSS or FSA for processing. JES will not send another order to the FSS or FSA until it has received a response from the FSS or FSA indicating the status of the previous order. The response clears the order path and allows JES to issue other orders to the FSS or FSA. If the FSS or FSA does not respond to a JES order, JES cannot communicate with the FSS or FSA. The FSS/FSA responds to JES through the FSIREQ SEND or FSIREQ CONNECT/DISCONNECT function calls.

JES uses the FSA POST service to let the FSA know that it has data sets that are ready to be processed.

Order Processing - Communication from JES to the FSS/FSA

JES tells the FSS/FSA that there is work to be done by using the FSI ORDER service. When an operator issues a JES command that requires the FSS/FSA to do some work, JES converts that command into an **order**. That order represents work that the FSS/FSA will do on behalf of JES.

The FSI Order Routine

The FSI order routine receives control when JES issues the FSIREQ ORDER function call. There must be an FSI order routine associated with the FSS and each FSA. JES knows about the FSI order routine because the FSS/FSA supplied the address of the order routine during FSS or FSA Connect processing as part the CONNECT parameter list. See “Preparing for FSS CONNECT” on page 32 for more information about FSS Connect processing. See “Preparing for FSA CONNECT” on page 42 for more information about FSA Connect processing.

Although part of the FSS or FSA, the FSI order routine runs under the control of a JES TCB or SRB. While the order routine runs, JES is unable to provide any other services and overall system performance may be impacted. Therefore, the order routine should not do any lengthy processing.

FSI ORDER function calls are split into two categories; **synchronous** orders that require minimal processing and can be responded to immediately, and **asynchronous** orders that require substantial processing by the FSS or FSA and therefore cannot be responded to immediately. The order routine of the FSS or FSA responds directly to the order. IBM recommends that the order routine immediately return control to JES with an indication that the order response will be returned later. Later sections describe how to respond to the JES order.

Since orders occur asynchronously, the FSS or FSA main task will check at appropriate points in its processing to see if an order has been issued.

Order Processing Parameter List

Before JES issues the FSIREQ macro to initiate FSI Order processing, JES fills in the fields of the FSIREQ parameter list. The address of the FSIREQ parameter list is in register 1. This section discusses those fields that are common to all order processing. Fields that are specific to a particular order are discussed in the section where the particular order is discussed (Chapter 6, "Establishing FSS/JES Communication," on page 29 through Chapter 13, "Stopping an FSS," on page 113).

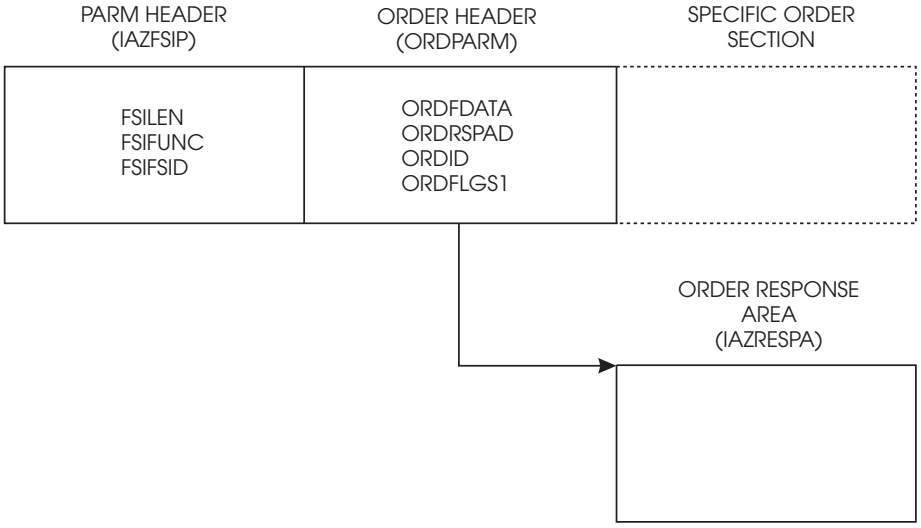

Figure 6. Parameter List for Order Processing

JES fills in the following fields of the **common parameter header** portion of the FSIREQ parameter list:

FSILEN
The total length of the order parameter list. The order parameter list consists of the common parameter header, the common order header and the section for the specific order.

FSIFUNC
JES assigns the symbolic value FSIORDER to this field to indicate the type of function that is required.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA. JES assigns the FSS an identifier of the form xxxx0000, where xxxx is a unique number. JES assigns the FSA an identifier of the form xxxxyyyy, where xxxx corresponds to the controlling FSS identifier, and yyyy is a unique number for each FSA.

FSIPEXT
If this field is non-zero, then there is an existing extension to this parameter list. The address of the extension is the contents of this field.

The only function that has an extension area is GETDS.

JES fills in the following fields of the **common order header** portion of the FSIREQ parameter list:

ORDFDATA
A 4-byte field that is used by the FSS or FSA. This field can be the address of a control block that contains information to allow the order routine to respond immediately to orders or notify (POST) the appropriate FSS or FSA task that an order is waiting to be processed. The FSS/FSA passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value in the order parameter list.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The specific order ID number. Refer to Chapter 6, "Establishing FSS/JES Communication," on page 29 through Chapter 13, "Stopping an FSS," on page 113 to determine the order ID number for each specific order.

The FSA is responsible for filling in the following field of the **common order header** portion of the FSIREQ parameter list:

ORDFLGS1
An indicator for whether the FSA is responding to the order synchronously or asynchronously.

ORDSRESP
Synchronous response - The order response area is currently filled in. The FSS or FSA needs to do no further processing.

ORDARESP
Asynchronous response - The order response area is not currently filled in. The FSS or FSA needs to do further processing and will notify JES, by using an FSIREQ SEND function call, that it has completed processing the order.

Function of the FSI Order Routine

When the FSI order routine receives the order, it:

- Determines the type of order issued
- Either processes the order directly or posts the appropriate FSS/FSA task to process the order asynchronously
- Saves the FSIPARM parameter list.

The order routine determines the type of order by testing the value of the ORDID field in the common order header of the order parameter list. The address of the order parameter list is in register 1.

The order routine also has the responsibility of determining whether to respond to the order **synchronously** or **asynchronously**.

Synchronous Processing: If the FSS/FSA can immediately respond to the order, it must:

1. Initialize the appropriate field(s) of the order response area. The ORDRSPAD field of the order parameter list contains the address of the order response area (mapped by IAZRESPA).
2. Set ORDFLG1 equal to ORDSRESP to inform JES that the required information is in the order response area. (ORDFLG1 is a field in the common order header of the IAZFSIP mapping macro.)
3. Return control to JES.

Notes:

1. If, for some reason, the FSS/FSA cannot handle a specific order, the FSS/FSA should set register 15 equal to a non-zero return code and return control to JES. This will cause JES to terminate the FSS address space.
2. If, for some reason, the FSS/FSA decides to ignore a specific order, the FSS/FSA should respond to the order synchronously by using the previous procedure. In this case, however, no processing will be done by the FSS or FSA before control is returned to JES. JES will think that the order has been processed and processing will continue.

Asynchronous Processing: If the FSS/FSA cannot immediately respond to the order, it must:

1. Set ORDFLG1 equal to ORDARESP to inform JES that the FSA will respond to the order at a later time by means of a FSI SEND request. (ORDFLG1 is a field in the common order header of the IAZFSIP mapping macro.)
2. The order routine should save the address of the FSIREQ parameter list into storage the FSS/FSA has access to. The ORDFDATA field can be used to accomplish this.
3. Let the FSS/FSA main line code know that an order has been received. The FSS/FSA main line code initializes the appropriate field(s) of the order response area after the request from JES has been fulfilled. The ORDRSPAD field of the order parameter list contains the address of the order response area (mapped by IAZRESPA). Register 1 contains the address of the order parameter list. The FSS or FSA responds to the order by using the FSI SEND request.
4. Return control to JES.

Coding Considerations

When coding the FSI order routine you must consider that:

- The FSI order routine must reside below the 16-megabyte line.
- No SVCs can be issued from an FSI Order routine.

Responding to an Order - Communication from the FSS/FSA to JES

The FSS/FSA tells JES that it has completed a piece of work by using the FSIREQ SEND or FSIREQ CONNECT/DISCONNECT function calls. These function calls are used only for asynchronous responses. “Synchronous Processing” on page 19 discusses immediate (synchronous) responses to an order from JES.

When the FSS/FSA uses the FSIREQ SEND function call to respond to an order from JES, it is referred to as a **solicited** SEND request. Most instances of the FSI SEND service are for solicited SEND requests. When the FSS/FSA uses the FSI SEND service for a reason other than a response to a JES order, it is referred to as an **unsolicited** SEND request.

Send Processing in Response to an Order

Before the FSS or FSA issues the FSIREQ macro to initiate FSI SEND processing, the FSS or FSA fills in the fields of the FSIREQ parameter list.

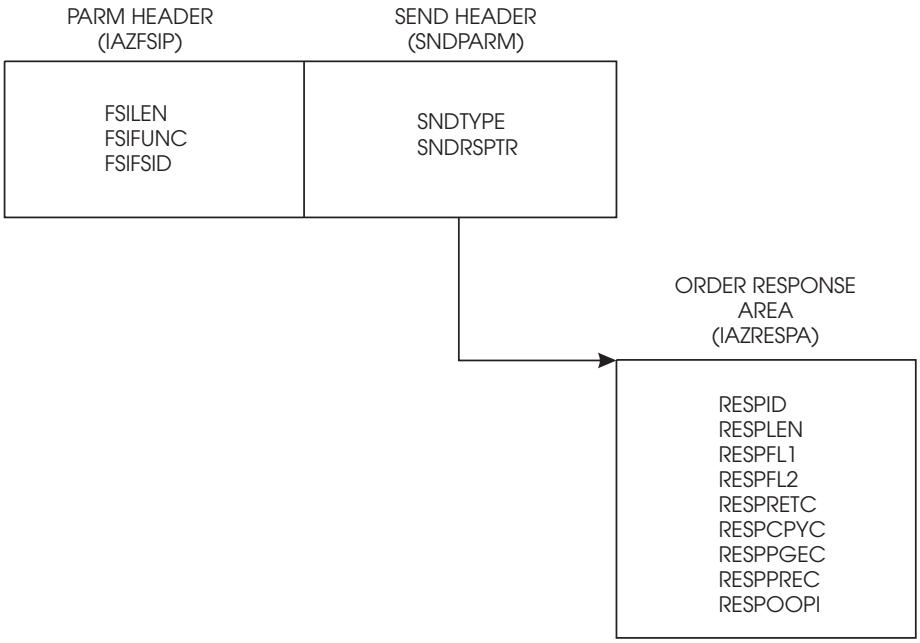

Figure 7. Parameter List for Send Processing

The FSS/FSA needs to initialize the following parameters in the common parameter header section before it issues the FSIREQ SEND function call:

Field Name	Value (bytes)	Value to be Assigned
Common Parameter header		
FSILEN	4	Length of SEND parameter list
FSIFUNC	4	FSISEND
FSIFSID	4	The FSS/FSA identifier.
SEND Function Dependent Area		
SNDTYPE	1	SNDTYRSP
SNDRSPTR	4	Address of the order response area for unsolicited send requests

FSILEN
The length of the entire SEND parameter list. The SEND parameter list consists of both the IAZFSIP common header section and the SEND function dependent section.

FSIFUNC
The SEND function ID number. The FSS/FSA assigns the symbolic value FSISEND to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

FSIFSSID
This field contains the FSS portion of the FSS/FSA identifier.

FSSFSAID
This field contains the FSA portion of the FSS/FSA identifier.

The FSS/FSA needs to initialize the following parameters in the SEND function dependent section before it issues the FSI SEND request:

SNDTYPE
The SNDTYPE ID number. The FSS/FSA sets this field equal to SNDTYRSP. SNDTYRSP indicates that the SEND request is in response to an order.

SNDRSPTR
The address of the order response area. For a solicited SEND request, JES supplies this address in the ORDRSPAD field. For an unsolicited SEND request, the FSA supplies the address.

Initializing the Order Response Area

After the FSS or FSA does some processing to fulfill the JES order, the FSS or FSA initializes the order response area (IAZRESPA).

The following table lists the IAZRESPA fields, the lengths of these fields, and the information the FSA may provide in each field. Detailed descriptions of the value assignments follow the table.

Field Name	Length (bytes)	Value to be assigned
IAZRESPA Order Response Area		
RESPID	4	"RESP"
RESPLEN	4	Length of the response area
RESPFL1	1	Device status
RESPFL2	1	Order processing status
RESPRETC	4	Return code of requested function
RESPCPYC	2	Copy number of data set at OOP
RESPPGEC	4	Page number of data set at OOP
RESPLREC	4	Logical record number of data set at OOP
RESPOOPI	12	Identifier of data set at OOP

The FSS/FSA needs to initialize the following parameters in the Order Response Area before it issues the FSI SEND request:

RESPID
"RESP" - The identifier of the response area

RESPLEN
The length of the response area

RESPFL1
If the device is not active, the FSA initializes this flag byte with one of the following indicators:

RESP1DIN
The device is inactive

RESP1DSP
The device is stopped

RESPFL2
The FSA uses this flag byte to notify JES of order processing status. The FSA can set one of the following indicators:

RESP2EOD
The end of data (EOD) was reached on a forward synch action.

RESP2NDS
No data set was active at the OOP (operator observation point).

RESP2ETE
Environmental-type error. An environmental-type error is one that can be fixed without bringing the FSA down. For example, the printer might simply need more toner. The FSS uses RESP2ETE to tell JES not to bring the FSA down for this minor problem.

RESPRETC
The return code for the requested function. If the FSS or FSA completed the order successfully, this field is set to zero. If the FSS or FSA could not complete the order, it sets this field to a value greater than zero.

RESPCPYC
The copy number of the data set at the OOP.

RESPPGEC
The page number of the data set at the OOP.

RESPLREC
The approximate logical record number of the data set at the OOP.

RESPOOPI
The identifier of the data set at the OOP.

Specific responses to individual orders will vary in the amount of the above information that needs to be included in the order response area. Refer to Chapter 7, "Establishing FSA/JES Communication," on page 37 through Chapter 13, "Stopping an FSS," on page 113, for that specific information.

Issuing the FSIREQ SEND Request

When the FSA completes the initialization of the response area and SEND parameter list, it issues the FSIREQ macro to invoke the FSI SEND communication service. The format of this macro call is:

```
FSIREQ REQUEST=FSISEND,TARGET=JES,PARM=SEND
parm-list-addr,FSID=value-addr
```

Note: See "FSIREQ Macro Format" on page 13 for a complete description of this macro.

On return from SEND processing, register 15 contains either a zero return code indicating success or a non-zero return code indicating an error occurred.

Unsolicited Send Processing

The FSA can initiate communication with JES through unsolicited SEND requests. The FSA processes an unsolicited SEND the same way it processes a solicited SEND (a send in response to an order). The same parameter list is used, the parameter list is filled in the same, and it is passed to JES by using the FSIREQ SEND function call.

The only difference is that for an unsolicited SEND request, the FSA must provide the order response area. The FSIREQ SEND parameter list the address of this order response area. JES will not provide the order response area for unsolicited SEND requests.

There are four occasions when the FSS/FSA can use the FSI SEND service to initiate an unsolicited SEND request. The first is when a data set has reached the operator observation point (OOP) of a device. The second occasion is when the

FSA needs to notify JES that it is terminating. The third occasion is when intervention is required, and the fourth occasion is when intervention is cleared.

Initializing the FSIREQ Parameter List

The FSA must fill in the following parameters in the common parameter section before it issues the FSI SEND request:

FSILEN
The length of the entire SEND parameter list. The SEND parameter list consists of both the IAZFSIP common header section and the SEND function dependent section.

FSIFUNC
The SEND function ID number. The FSA assigns the symbolic equate value FSISEND to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

The FSA must fill in the following parameters in the SEND parameter section before it issues the FSI SEND request:

SNDTYPE
The FSA uses this flag byte to indicate to JES the type of information being sent. For this issuance of the SEND request, the FSA is expected to set the following indicator:

SNDTYTDS or SNDTYFIT
The FSA is satisfying JES's request for notification (GDSTRKDS) when the data set reaches the OOP or the FSA is terminating. Refer to Chapter 9, "Issuing Data Requests to JES," on page 51 and Chapter 12, "Stopping an FSA," on page 107 for more information.

SNDTYINT
The FSA is satisfying JES's request for intervention.

SNDTYICL
The FSA is satisfying JES's request to clear intervention.

SNDRSPTR
The address of the FSA-provided response area.

CONNECT/DISCONNECT Processing in Response to an Order

During FSS and FSA initialization and termination the CONNECT/DISCONNECT FSIREQ function call is used to indicate to JES that processing of a START FSS or START FSS order is complete. The FSS or FSA CONNECT request is issued as a response to a START FSS request or START FSA order from JES. The FSS or FSA DISCONNECT request is issued as a response to a STOP FSA or STOP FSS order from JES.

Post Processing

The FSI post routine receives control when JES issues the FSIREQ POST function call. There must be an FSI post routine associated with each FSS and FSA. The FSS/FSA supplies the address of the post routine during FSS or FSA Connect processing as part of the CONNECT parameter list. See "Preparing for FSS CONNECT" on page 32 for more information about FSS Connect processing. See "Preparing for FSA CONNECT" on page 42 for more information about FSA Connect processing.

The FSI Post Routine

Although part of the FSS/FSA, the FSI post routine code runs under the control of a JES TCB or SRB. While the post routine is running, JES is unable to provide any other services and overall system performance may be impacted. Therefore, the post routine should not do any lengthy processing.

Function of the FSI Post Routine

The only use of the FSIREQ POST function call is for JES to notify the FSA that there are data sets available for processing.

The FSA POST routine uses information passed in the POST parameter list to indicate to the appropriate FSA that GETDS requests are now allowed. The POSFDATA field points to this information This field is filled in from the CDFFDATA field during connect processing.

If the POST processing is successful, the FSA POST routine returns control to JES with a zero return code in register 15. If an error occurs during processing, the FSA POST routine returns control to JES with a non-zero return code in register 15. JES abnormally terminates the FSS address space if JES receives a non-zero return code.

Post Processing Parameter List

Before JES issues the FSIREQ macro to initiate FSI POST processing, JES fills in the fields of the FSIREQ parameter list.

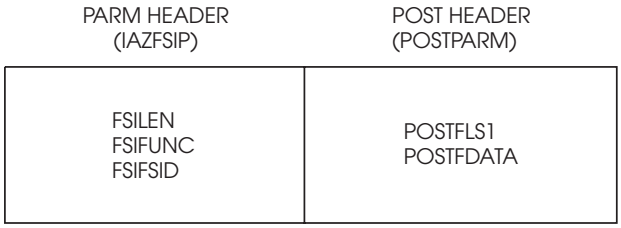

Figure 8. Parameter List for Post Processing

In the common parameter header section of the POST parameter list, JES passes the following information:

FSILEN
The length of the POST parameter list, which consists of the common header section and the POST function dependent section.

FSIFUNC
The POST function ID number. The symbolic equate FSIPOST represents this value.

FSIFSID
The FSS/FSA IDs that JES assigned to the FSS/FSA during startup.

In the function dependent section of the POST parameter list, JES passes the following information:

POSTFLS1
This status flag byte indicates the reason for the POST request. The following indicator is set:

POSTGDS B'10000000'
GETDS requests can now be satisfied.

POSFDATA
A 4-byte field that is used by the FSS or FSA. This field can be the address of a control block that contains information that allows the post routine to notify the appropriate FSA task that a GETDS can be issued.

Types of Orders

There are ten types of orders or work that the FSS/FSA does when JES invokes the FSI ORDER service. The following table describes:

- The function you want to perform (Function)
- The order needed to perform that function (Order)
- The expected response to that order (Response)
- The response method required (Response Method)
- A reference to where detailed information about that order can be found (Reference)

Table 2. Orders and Responses

Function	Order	Response	Response Method	Reference
Start an FSS	MVS START command	Connect	Asynchronous	Chapter 6, "Establishing FSS/JES Communication," on page 29
Start an FSA	Start FSA	Connect	Asynchronous	Chapter 7, "Establishing FSA/JES Communication," on page 37
Start device	Send	Asynchronous	Chapter 8, "Starting an FSS Device," on page 47	
Stop a device	Stop device	Send	Asynchronous	Chapter 11, "Stopping an FSS Device," on page 103
Stop an FSA	Stop FSA	Disconnect	Asynchronous	Chapter 12, "Stopping an FSA," on page 107
Stop an FSS	Stop FSS	Disconnect	Asynchronous	Chapter 13, "Stopping an FSS," on page 113
Obtain information about the current data set	Query	Send	Synchronous	"The Query Order" on page 89
Change device characteristics	Set	Send	Asynchronous	"The Set Order" on page 91
After the current data set	Synch	Send	Asynchronous	"The Synch Order" on page 93
Change the set up of the device	Intervention	Send	Asynchronous	"The Intervention Order" on page 99

Addressing Mode - AMODE

AMODE is a program attribute that can be specified (or defaulted) for each CSECT, load module, and load module alias. AMODE states the addressing mode that is expected to be in effect when the program is entered. AMODE can have one of the following values:

AMODE 24
The program is designed to receive control in 24-bit addressing mode.

AMODE 31
The program is designed to receive control in 31-bit addressing mode.

AMODE ANY
The program is designed to receive control in either 24-bit or 31-bit addressing mode.

You should be concerned about allowable values for the following:

- AMODE in which your FSS/FSA code runs. This AMODE value is set by the CDFFL331 bit in the CONNECT parameter list.
- AMODE in which JES enters FSS/FSA code. This AMODE value is set by the CDFS1A31 bit in the CONNECT parameter list.

Initially the FSS sets the CDFFL331 bit of the CONNECT parameter list to indicate whether or not it supports AMODE(31). If the FSS supports AMODE(31), JES sends all FSI parameter lists above the line.

JES sets the CDFS1A31 bit of the CONNECT parameter list to indicate whether or not it supports AMODE(31). If JES supports AMODE31 the FSS/FSA can, but does not necessarily have to, pass FSI parameter lists above the line.

The following matrix presents the allowable AMODE values under various conditions.

	AMODE	
Conditions	**for FSS/FSA code**	**in which JES enters FSS/FSA code**
FSS supports AMODE(31) JES supports AMODE(31)	31	31
FSS does not support AMODE(31)	24	24
JES does not support AMODE(31)	ANY	24

Pointer-defined Linkage

Pointer-defined linkage sets the appropriate addressing mode when control is passed from one routine to another. See *z/OS MVS Programming: Assembler Services Guide*.

Residency Mode - RMODE

RMODE states the virtual storage location (either above 16 megabytes or anywhere in virtual storage) where the program should reside. RMODE can have the following values:

RMODE 24
The program is designed to reside below 16 megabytes in virtual storage. MVS places the program below 16 megabytes.

RMODE ANY
The program is designed to reside at any virtual storage location, either above or below 16 megabytes. MVS places the program above 16 megabytes unless there is no suitable storage above 16 megabytes.

The following matrix presents the allowable RMODE values for the FSI parameter lists under various conditions.

	Residency Location of Parameter List	
Conditions	**for FSS CONNECT PSIPARM**	**for all other FSIPARMs**
FSS supports AMODE(31) JES supports AMODE(31)	31 31	31 31

Chapter 6. Establishing FSS/JES Communication

JES starts the functional subsystem (FSS) address space either during JES initialization or in response to an operator command to start a printer under control of the FSS. When the FSS receives control, it performs initialization and then responds to JES.

If the FSS successfully starts, it issues an FSI CONNECT request to JES to establish the FSS-level functional subsystem interface (FSI). FSS CONNECT processing:

- Notifies JES that the FSS is started.
- Identifies to the FSI the addresses of FSS routines that are to receive control when JES issues the FSIREQ macro.
- Identifies to the FSI the addresses of JES routines that are to receive control when the FSS issues the FSIREQ macro.

Completion of FSS level CONNECT processing signals JES to issue a START FSA order to the FSS.

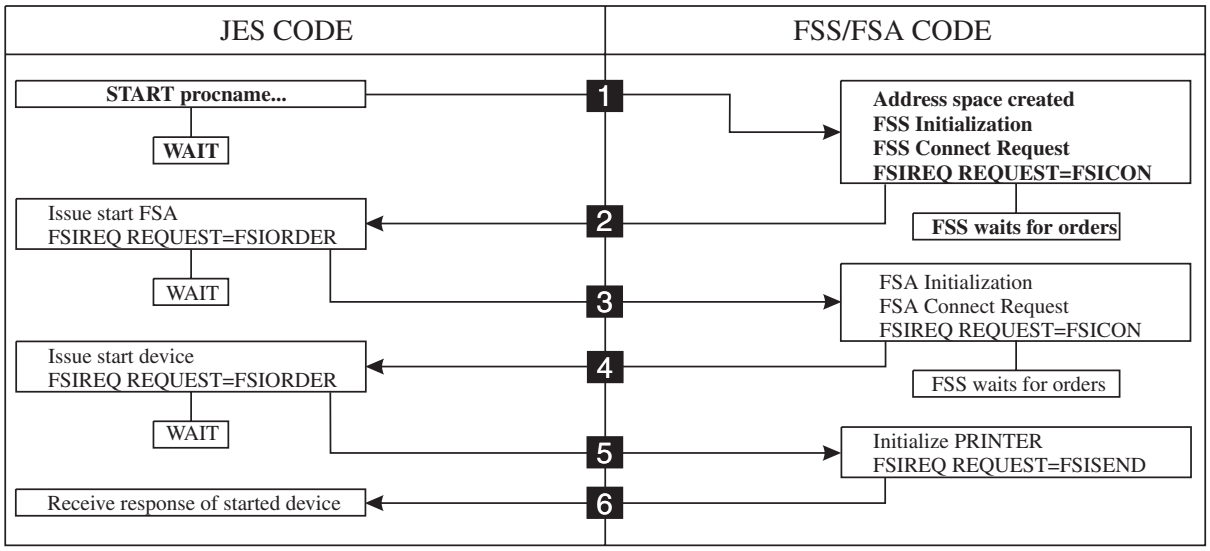

Figure 9. An Overview of FSI Startup Processing

The following topics describe:

- How JES starts the FSS
- Initialization required by the FSS
- How the FSS connects to JES.

Starting an FSS

When JES determines that an FSS should be started, it creates an MVS START command using information from the corresponding FSSDEF initialization statement. The START command creates and initializes the FSS address space.

Figure 10 on page 30 shows the format of the MVS START command and the relationship between the FSSDEF parameters and the MVS START command parameters. In the diagram, for JES2 3.1.1 and above, the FSSDEF *acccccccc* value

maps to the START *ssname* value, and the PROC=*cccccccc* value maps to the START *membername* value. For JES3, the FSSDEFPNAME *acccccccc* value maps to the START *membername* value, and the FSSNAME=*acccccccc* value maps to the START *ssname* value.

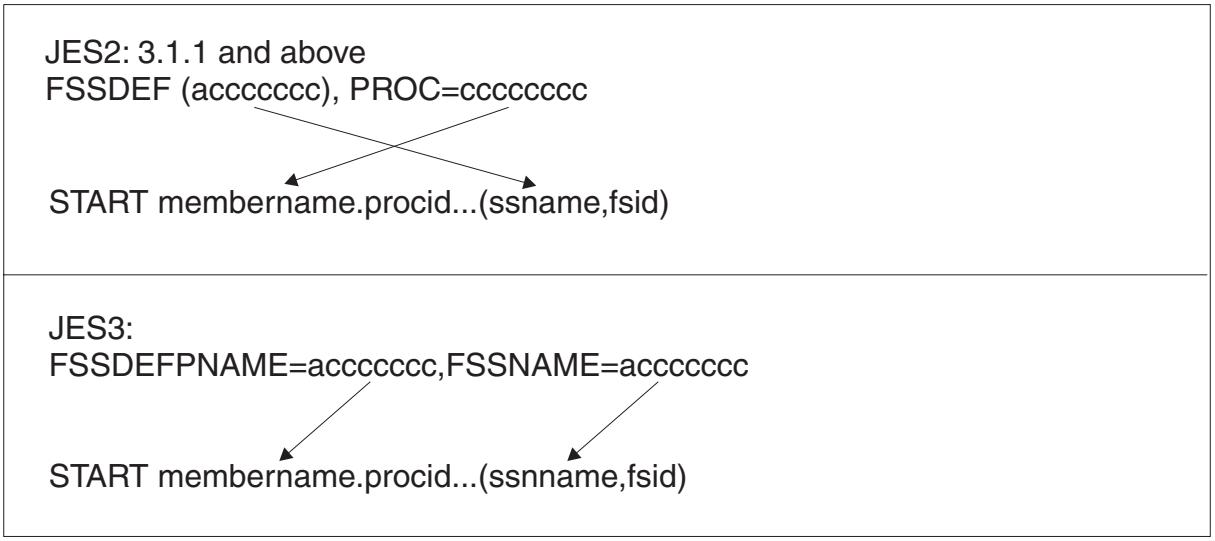

Figure 10. FSSDEF/MVS START Command parameter relationships

membername
The membername of a procedure in SYS1.PROCLIB or one of its concatenations that contains the JCL required to start the FSS address space.

procid
The identifier that will be assigned to the started task (the FSS) created by the issuance of the MVS START command.

ssname
The subsystem name of the JES that issued the MVS START command.

fsid
The FSS part of the FSID for all FSAs running under this FSS (the FSS id). This field contains the EBCDIC representation of the high-order halfword of the FSID that is assigned by JES during JES initialization. JES assigns the FSS an identifier of the form xxxx0000, where xxxx is a unique number. JES assigns the FSA an identifier of the form xxxxyyyy, where xxxx corresponds to the controlling FSS identifier, and yyyy is a unique number for each the FSA.

Initializing the FSS Address Space

The FSS receives control from JES in the normal MVS task control block (TCB) environment created for a started task. The FSS performs initialization and then establishes the FSS-level interface to JES. General initialization procedures and recommendations are described below.

The FSS must place itself into supervisor state, key 1, to use the FSI services. The FSS uses the MODESET macro to perform this task. The format of the MODESET macro calls are:

```
MODESET MODE=SUP (places the FSS in supervisor state)

MODESET EXTKEY=JES (places the FSS in key 1)
```

The FSS must also be running non-swappable. The FSS uses the SYSEVENT macro to perform this task. The format of the SYSEVENT macro call is:

```
SYSEVENT DONTSWAP (causes the FSS to run non-swappable)
```

Note: An FSS can enter supervisor state only if it is running with APF authorization.

The name of your FSS program must be added to the program properties table (PPT) with the KEY parameter set equal to one. Refer to *z/OS MVS Initialization and Tuning Reference* for information about the SCHEDxx parmlib member.

It is **recommended** that the FSS establish an ESTAE routine so that it can handle its own recovery processing.

You should use the appropriate AMODE and RMODE values for defining the FSS load module. See 26 for information about the factors that govern AMODE and RMODE settings.

Retrieving the MVS START Command and Token

The FSS needs to retrieve the information passed in the MVS START command during FSS startup so that it can perform verification and initialize the CONNECT parameter list. The FSS uses the EXTRACT macro to retrieve information. The FSS must provide an area to receive the EXTRACT information and it must supply the address of this 'answer-area' on the EXTRACT macro call.

The format of the EXTRACT macro call is:

```
EXTRACT answer-area-address, 'S',FIELDS=(COMM)
```

On return from the EXTRACT macro request, the 'answer-area' has the address of the communications area (mapped by IEZCOM). This communications area consists of:

- the communications event control block
- the command input buffer (CIB) for the MVS START command

The FSS should verify that a token was provided during startup to insure it was not started by an operator-issued MVS START command. If a token was specified, as in the case of a JES-issued START command, the high-order bit of the token field will be set to one. If a token was not provided, the FSS needs to decide whether or not it should terminate.

If it continues in this environment, it cannot run as an FSS in the JES environment.

The CIB contains the MVS START command parameters. The FSS must verify that the MVS START command parameters were specified by insuring that the CIBDATLN field of the CIB is greater than zero. If the START command parameters were not specified, the FSS must terminate.

The FSS needs to retrieve the FSS id (fsid) and JES subsystem name (ssname) from the CIB and save this information for subsequent FSI processing. The fsid is an identifier of the form xxxx0000, where xxxx is a unique number assigned by JES. The FSS must not release the MVS START command CIB until after issuing the FSIREQ CONNECT request. The FSS must supply the FSS id in all FSIREQ requests. It must supply the JES subsystem name in CONNECT/DISCONNECT FSIREQ requests.

Preparing for FSS CONNECT

If the FSS successfully starts, it can establish the FSS-level interface to JES. Preparation for the FSIREQ CONNECT request consists of three steps. The FSS needs to:

1. GETMAIN enough storage for the IAZFSIP mapping macro and the SSOB/SSIB pair. The storage for the SSOB/SSIB pair must be contiguous.
2. Provide an 18-word save area.
3. Initialize the CONNECT parameter list.

If the FSS discovers a problem during initialization and is unable to connect, the FSS should go through normal MVS termination. MVS services notify JES of this termination.

Initializing the FSS Level FSIREQ CONNECT Parameter List

The FSS needs to initialize certain fields of the FSIREQ CONNECT parameter list. The following figure shows the connection between the different sections of the FSIREQ parameter list.

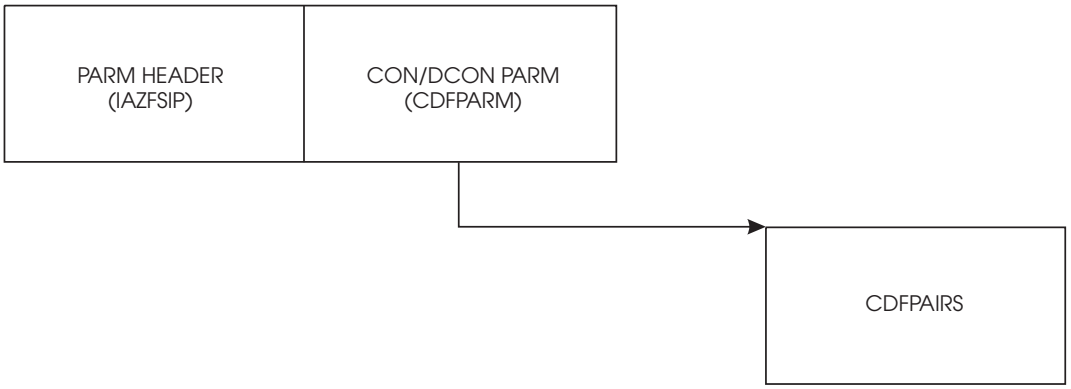

Figure 11. FSIREQ Parameter Lists for FSS CONNECT Processing

The following table lists the required fields, the lengths of these fields, and the values that the FSS must assign. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section (IAZFSIP)		
FSILEN	4	Length of CONNECT parameter list
FSIFUNC	4	FSICON
FSIFSID	4	The FSS ID
CONNECT Function Dependent Section (CDFPARM)		
CDFFLGR2	1	Functions that involve operator intervention
CDFFLGR3	1	Specifies JES functions supported by the FSA
CDFSTOR	4	Address of storage for SSOB/SSIB pair
CDFFDATA	4	Address of a control block containing FSS information
CDFIDNO	4	2

Field Name	Length (bytes)	Value to be assigned
CDFIDNA	4	Address of FSS function ID/address pairs
CDFSSID	4	Name of the JES to which the FSS is connecting
CDFFLGS1	1	Functions supported by JES
Function ID/Address Pairs (CDFPAIRS)		
CDFID	4	FSIORDER
CDFAD	4	Address of the FSI ORDER routine
CDFID	4	FSIPOST
CDFAD	4	Address of the FSI POST routine

FSILEN
The length of the entire CONNECT parameter list. The CONNECT parameter list consists of both the IAZFSIP common header section and the CONNECT function dependent section. The length does not include the CDFPAIRS section.

FSIFUNC
The CONNECT function ID number. The FSS must issue the FSIREQ macro with REQUEST=FSICON specified.

FSIFSID
The FSS ID that JES assigned when it started the FSS. The FSS obtains the FSID from the command input buffer (CIB).

CDFFLGR2
Indicates functions that require intervention. When any of these bits are on, JES issues the setup required message, when the device is started. When all bits are off, JES suppresses the setup required message.

CDFFL2BT B'10000000'
FSS might use the burster-trimmer-stacker.

CDFFL2FL B'01000000'
FSS might use a flash.

CDFFL2FO B'00100000'
FSS might need a forms change.

CDFFL2CF B'00010000'
FSS might use continuous paper.

CDFFLGR3
Indicates the functions supported by this FSS. These indicators may be set:

CDFFL3MS B'10000000'
Extended message routing is supported.

CDFFL331 B'01000000'
AMODE(31) is supported.

CDFFL34D B'00100000'
Support for 4-digit device numbers.

CDFSTOR
The address of the storage that the FSS GETMAINed for the contiguous SSOB/SSIB pair. The length of the SSOB/SSIB pair can be obtained by the IEFJSSOB and IEFJSSIB macros.

CDFFDATA
A 4-byte field that is used by the FSS. This field can be the address of a control block that may contain FSS information needed by the FSI ORDER and FSI POST routines. JES will return this field when JES invokes the FSI POST and FSI ORDER routines. For FSI ORDER processing, JES returns the address in the ORDFDATA field. For FSI POST processing, JES returns the address in the POSFDATA field. One of the things this control block may contain is an ECB that the ORDER or POST routines can post.

CDFIDNO
The number of function ID/address pairs pointed to by CDFIDNA.

CDFIDNA
The address of the first function ID/address pair. The function pairs should be defined in the format mapped by CDFID and CDFAD in the IAZFSIP mapping macro. The CDFID and CDFAD format are repeated for each function the FSS or FSA provides. The FSS should provide a CDFID and CDFAD pair for an FSS ORDER routine and an FSS POST routine.

CDFSSID
The name of the JES to which the FSS is issuing the CONNECT request. If the FSS does not specify this parameter, it will be connected to the primary JES defined to your installation. The FSS obtains the name of the JES from the CIB. In the JES2 environment, it is crucial that the FSS supply the CDFSSID since JES2 supports poly-JES (Many versions of JES2 can run under the same MVS.)

CDFFLGS1
Indicates functions supported by JES. The JES fills in this value which is used by the FSS on return from the CONNECT FSIREQ.

CDFS1INT B'10000000'
Unsolicited send for intervention conditions.

CDFS1ETE B'01000000'
Support for environmental-type errors. (Environmental-type errors are minor errors that do not require the FSS or FSA to be brought down.)

CDFS1A31 B'00100000'
Support for AMODE(31).

CDFS1ESS B'00010000'
Support for ESS keywords.

CDFS1DNR B'00001000'
Supports device not responding conditions.

CDFID
The FSI ORDER function ID. The FSS may assign the symbolic equate FSIORDER to this field.

CDFAD
The entry point address of the FSS's FSI ORDER routine.

Issuing the FSS Level FSIREQ CONNECT Request

When the FSS has completed initializing the CONNECT parameter list, it issues the FSIREQ macro to invoke the FSI CONNECT service. communications list. The format of this macro call is:

```
FSIREQ REQUEST=FSICON,TARGET=JES,PARM=CONNECT
parm-list-addr,FSID=value-addr
```

See Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

FSS CONNECT processing

The FSIREQ CONNECT request results in a call to the SSI CONNECT routine of the subsystem specified in the CDFSSID field of the CONNECT parameter list. The CONNECT parameter list is used as the SSOB extension for the SSI call. The SSI CONNECT routine loads the JES functional subsystem support modules into the FSS address space and then passes control to the FSI CONNECT routine in that module.

The FSI CONNECT routine allocates storage for and initializes the various FSI-related control blocks, for example, the functional subsystem vector table (FSVT) and the functional subsystem control tables (FSCTs). JES builds two FSCTs for the FSS. JES initializes one FSCT with the address of the FSS' FSI ORDER routine which was passed in the CDFAD field of the CONNECT parameter list. JES initializes the second FSCT with the addresses of FSS level FSI services provided by JES. On subsequent FSIREQ requests, the FSIREQ macro searches the appropriate FSCT to obtain the address of the FSI routine it needs to branch to. The JES also sets the flag CDFFLGS1 to indicate those special functions supported by the JES. These functions include: Unsolicited Sends for Intervention and ETE Type Errors.

If the FSS is connecting to JES2, the FSI CONNECT routine also establishes the cross memory environment between the FSS address space and the JES2 address space.

At completion of FSS CONNECT processing, register 15 contains the SSI CONNECT function dependent return code. A zero return code indicates the FSS level interface to JES is established.

How JES Handles Logic Errors and Abends

If an error occurs during FSS CONNECT processing, JES sets a non-zero return code in the SSOBRETN field of the SSOB. An invalid FSID is an example of a possible error. JES then performs the same processing as if the FSS issued a DISCONNECT request requesting abnormal termination. See Chapter 13, "Stopping an FSS," on page 113 for more information about DISCONNECT processing.

How JES Monitors Timing of FSS CONNECT

When JES issues the MVS START command to the FSS, it starts a timer. If the FSS does not respond with an FSS CONNECT request in the specified time interval, JES simulates receiving an FSS level DISCONNECT response. See Chapter 13, "Stopping an FSS," on page 113 for more information about DISCONNECT processing.

JES2 issues message, HASP706 every five minutes, indicating that it is still waiting for the START command to complete, it continues to reset the timer and wait.

JES3 issues messages, IAT6373 and IAT6374, indicating that it is still waiting for the START command to complete, it continues to reset the timer and wait.

Chapter 7. Establishing FSA/JES Communication

When the JES operator issues the command to start an FSS printer device, JES determines if the FSS for which the printer is defined is currently active. If that FSS is not currently active, JES starts the FSS. Immediately after JES receives a response for the START FSS order from the FSS, JES issues a START FSA order for each FSA defined to that FSS. Refer to Chapter 6, "Establishing FSS/JES Communication," on page 29 for more information about starting an FSS. If the FSS is currently active, JES converts the command into a START FSA order to start the printer device.

If the printer is successfully allocated and initialization is complete, the FSA issues an FSIREQ CONNECT request to JES to establish the FSA-level functional subsystem interface (FSI). FSA CONNECT processing:

- Notifies JES that the FSA has successfully started.
- Identifies to the FSI the addresses of FSA routines that are to receive control when JES issues the FSIREQ macro.
- Identifies to the FSI the addresses of JES routines that are to receive control when the FSA issues the FSIREQ macro.

Completion of FSA level CONNECT processing signals JES to issue a START device order.

If the FSA could not be successfully started, either the FSS or the FSA (depending on when in time the failure was detected) issues a SEND request to JES indicating that the START FSA order was unsuccessful. See "FSA Could Not Be Started" on page 45 for more information about unsuccessful starts.

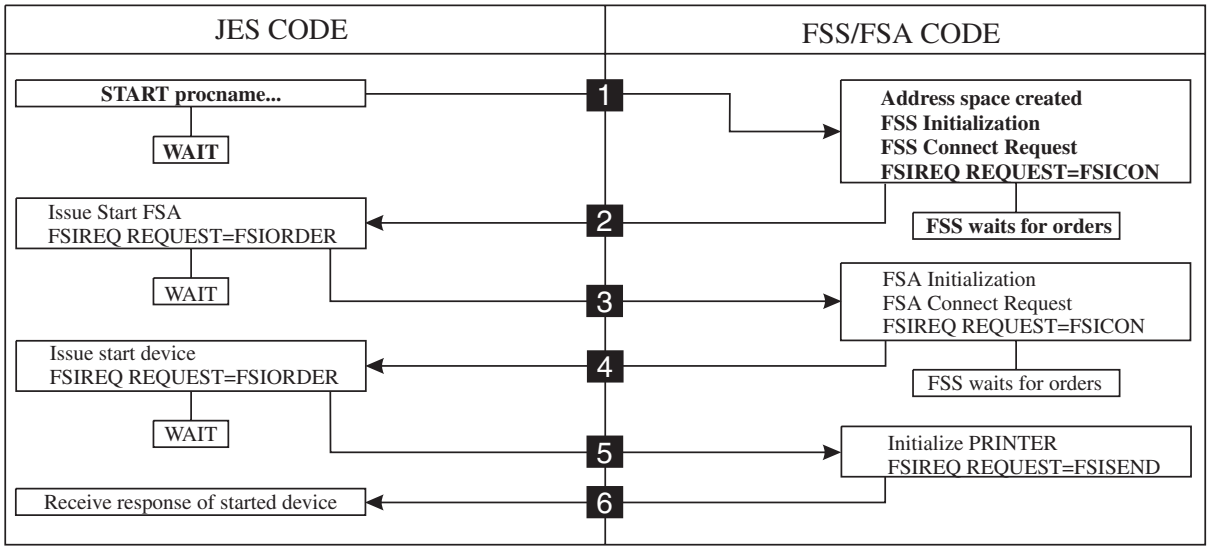

Figure 12. An Overview of FSI Startup Processing

Processing the START FSA Order

To start an FSA, JES issues the START FSA order to the FSS's FSI ORDER routine. During FSS CONNECT processing, the FSS places the address of the FSI ORDER routine into the CDFAD field of the CONNECT parameter list for use by JES. JES passes the address of the START FSA order parameter list in register 1. The parameter list contains the address of the order response area (IAZRESPA).

Refer to Chapter 5, "FSI Communication," on page 17 for general information about the responsibilities of the FSS's Order routine.

The START FSA order causes the FSS to attach an FSA task that will attempt to allocate and initialize a specific printer device. JES will not issue another order to the FSS until it receives a response to the START FSA order.

The START FSA order parameter list consists of the following sections:

- Common parameter header
- Common order header
- START order function dependent section
- Device initialization area
- Message routing information area (JES3 only)

The following figure shows the connection between the different sections of the FSIREQ parameter list.

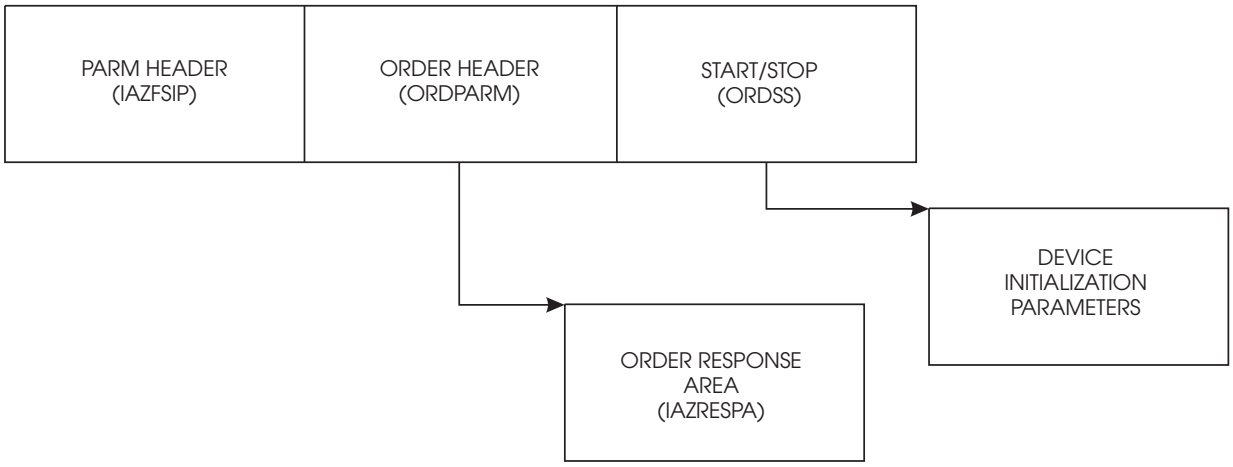

Figure 13. FSIREQ Parameter Lists for the START FSA Order

The following table shows the parameters that JES initializes for the START FSA order. The values that JES assigns are explained after the table.

Field Name	Length (bytes)	Value JES Assigned
Common Parameter Header (IAZFSIP)		
FSILEN	4	Length of START order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS identifier
Common Order Header (ORDPARM)		

Field Name	Length (bytes)	Value JES Assigned
ORDFDATA	4	Information supplied to JES in the FSS/FSA CONNECT parameter list (CDFFDATA)
ORDRSPAD	4	Address of the order response area
ORDID	2	ORDSTFSA
START Order Function Dependent Section (ORDSS)		
ORDSSSP	4	Address of device initialization area (ORDSSP1)
ORDSSID	4	FSS/FSA identifier of device to start
ORDSSAD4	4	Device address in 4-digit format
ORDSSAD	3	Device address in 3-digit format
ORDSSNA	8	Device name
ORDSSSP2	4	Address of message routing information
Device Initialization Area		
ORDSSPF1	1	Spacing flag byte
ORDSSPF2	1	Checkpoint flag byte
ORDSSPF3	1	NPRO timer flag byte
ORDSSKI	4	Initial checkpoint interval
ORDSSNI	4	Initial NPRO timer interval
Message routing information area		
ORDSS2LN	2	Length of the message routing information area
ORDSS2FL	1	Message routing flag
ORDSS2RC	16	MCS routing code mask
ORDSS2CN	4	Console ID in WTO format

FSILEN
The total length of the START order parameter list. The START order parameter list consists of the common parameter header, the common order header and the START order function dependent section.

Note: The device initialization area and the message routing information are not part of the total length. Field ORDSSSP contains the address of the device initialization area. Field ORDSSSP2 contains the address of the message routing area.

FSIFUNC
The ORDER ID number. JES assigns the value FSIORDER to this field.

FSIFSID
The FSS/FSA identifier.

FSIFSSID
The FSS identifier that JES assigned when it started the FSS.

FSIFSAID
This field is initialized to zero. The FSA sets this field to the FSA identifier when it issues the FSA-level CONNECT request. At this point in processing, the FSA identifier is contained in the ORDSSAI field of the START FSA order dependent section of the START FSA order parameter list.

ORDFDATA
The address of a control block containing FSS-related information. The FSS passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value in the START FSA order parameter list so that the FSS's FSI order routine can post the appropriate FSS task to process the order. This control block may contain the FSS or FSA ECB to be posted for processing. It may also be used to save the order parameter list for processing by the FSS/FSA or the QUERY order information for an immediate response.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The START FSA order ID number. JES assigns the value ORDSTFSA to this field. The order routine uses this value to determine what the order is and whether it should be responded to synchronously or asynchronously.

ORDSSSP
The address of the device initialization area. The device initialization area contains setup characteristics for the device.

ORDSSID
The FSS/FSA identifier of the device to start.

ORDSSSI
The FSS section of the identifier.

ORDSSAI
The FSA section of the identifier. Use this value to initialize the FSA portion of the FSIFSAID field.

ORDSSAD4
The 4-digit device address in printable form. This field will contain blanks if the printer is a non-channel attached device.

ORDSSAD
The 3-digit device address in printable form. This field will contain blanks if the printer is a non-channel attached device.

ORDSSNA
The device name in printable form. The device name is one of the keys that JES gives the FSS/FSA so that it can select some device default characteristics.

ORDSSPF1
This flag byte contains the JES spacing requirements for data sets printed on this device. The following indicators may be set:

ORDSSS1 B'10000000'
JES requires the FSA to single space the output.

ORDSSS2 B'01000000'
JES requires the FSA to double space the output.

ORDSSS3 B'00100000'
JES requires the FSA to triple space the output.

ORDSSSR B'00010000'
JES requires the FSA to space the output according to the requirements of the individual data set.

ORDSSPF2
This flag byte either specifies the type of JES checkpoint interval to be used for

output checkpointing on this device or specifies that the checkpoint feature should be disabled for this device. One of the following indicators may be set:

ORDSSKP B'10000000'
JES requires the FSA to take output checkpoints based on the page count specified in the ORDSSKI field.

ORDSSKT B'01000000'
JES requires the FSA take output checkpoints based on the time elapsed specified in the ORDSSKI field.

ORDSSKN B'00100000'
JES requires the FSA to disable the checkpoint feature.

ORDSSPF3
This flag byte specifies whether or not the FSA should use the NPRO (non-process runout) timer interval specified in ORDSSNI. The NPRO time interval is the interval during which output remains in the paper path but has not reached the stacker. This parameter is only valid for pipeline devices. After the NPRO timer specification has elapsed, the FSA forces the output to the stacker. One of the following indicators may be set:

ORDSSDN B'10000000'
JES requires that the NPRO timer be disabled.

ORDSSIN B'01000000'
JES requires that the NPRO timer interval value specified in the ORDSSNI field be used.

ORDSSKI
The initial checkpoint interval value.

ORDSSNI
The initial NPRO (non-process runout) time interval value.

ORDSS2LN
The length of the message routing information area (JES3 only).

ORDSS2FL
This flag byte specifies how JES3 wants FSA-related messages routed (JES3 only). The following indicator can be set:

ORDSS2CS B'10000000'
JES has specified a console ID in field ORDSS2CN.

ORDSS2RC
An MCS routing code mask for FSA-related messages (JES3 only).

ORDSS2CN
The console ID in WTO format where FSA-related messages are to be routed (JES3 only).

Initializing the FSA

The FSS decides if it is able to process the START FSA order. If it can, it attaches an FSA task which will then complete the initialization process. If the FSS cannot process the order, it must respond by using the FSIREQ SEND function call to indicate order processing was unsuccessful.

As the FSA initialization process continues, the FSA task uses the values passed in the device initialization area of the START FSA order. The initialization parameters

included in the device initialization are spacing requirements, checkpoint interval requirements, and NPRO (non-process runout) requirements. The preceding section describes these parameters in detail.

The FSA is now responsible for responding to the START FSA order. The proper asynchronous responses to this order are:

- If processing is successful - FSA level CONNECT
- If processing is unsuccessful - SEND with the RESPRETC field set to a non-zero value

FSA Successfully Started

If the FSA is successfully initialized, the FSA issues the FSA-level FSIREQ CONNECT request. This is the response to the START FSA order.

Preparing for FSA CONNECT

Before the FSA can issue the FSA level CONNECT, it must:

- Provide an 18-word save area
- Initialize the CONNECT parameter list.

Initializing the FSIREQ Connect Parameter List

The following figure shows the connection between the different sections of the FSIREQ parameter list.

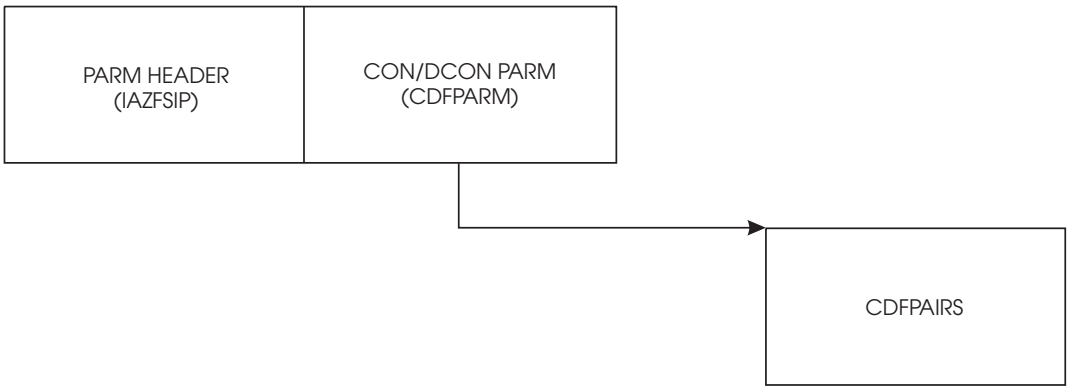

Figure 14. FSIREQ Parameter Lists for FSA CONNECT

The FSA needs to initialize the following parameters before it issues the FSIREQ CONNECT request.

Field Name	Value (bytes)	Value to be Assigned
Common Parameter Header (IAZFSIP)		
FSILEN	4	Length of CONNECT parameter list
FSIFUNC	4	FSICON
FSIFSID	4	The FSS/FSA identifier
CONNECT Function Dependent Area (CDFPARM)		
CDFFLGR2	1	Specifies functions that require operator intervention
CDFSTOR	4	Address of storage for SSOB/SSIB pair

Field Name	Value (bytes)	Value to be Assigned
CDFFDATA	4	Address of a control block containing FSA information
CDFIDNO	4	2 (Number of function ID/address pairs in pointed to by CDFIDNA)
CDFIDNA	4	Address of the function ID/address pair
CDFSSID	4	Name of the JES that the FSA is connected to

FSILEN
The length of the entire CONNECT parameter list. The CONNECT parameter list consists of the IAZFSIP common header section and the CONNECT function dependent section.

FSIFUNC
The CONNECT function ID number. The FSA assigns the symbolic value FSICON to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

FSIFSSID
This field contains the FSS portion of the FSS/FSA identifier.

FSSFSAID
This field contains the FSA portion of the FSS/FSA identifier. Use the value JES passed in the ORDSSAD field of the START FSA order parameter list to initialize this field.

CDFFLGR2
This flag byte specifies the operator intervention required functions that the device supports. If any of these bits are set, intervention orders for all of these functions are sent to the FSA. The FSA should only process the ones it can and ignore any others. One or more of the following indicators can be set:

CDFFL2BT B'10000000'
The device supports BTS (burster-trimmer-stacker) intervention.

CDFFL2FL B'01000000'
The device supports flash intervention.

CDFFL2FO B'00100000'
The device supports forms intervention.

CDFFL2CF B'00010000'
The device supports continuous forms intervention.

CDFFLGR3
This flag byte specifies the JES functions that the FSA supports. The following indicator can be set:

CDFFL3MS B'10000000'
The device supports extended message routing (JES3 only).

CDFSTOR
The address of the storage for the contiguous SSOB/SSIB pair.

CDFFDATA
The address of a control block containing FSA information that JES returns

when it invokes the FSI POST and FSI ORDER routines. This parameter enables the FSA to pass control information through FSIREQ to its POST and ORDER routines.

CDFIDNO
The number of function ID/address pairs pointed at by CDFIDNA. JES uses this number to determine how many pairs are contained in the CDFPAIRS portion of the CONNECT parameter list.

CDFIDNA
The address of the first pair of function ids and their respective addresses. The FSA level functions included in this section are FSIORDER and FSIPOST.

CDFSSID
Name of the JES that the FSA is connected to. If this parameter is not specified, the FSA is connected to the primary JES defined to your installation.

CDFFLGS1
Indicates functions supported by JES. This field is set in the FSS connect parameter list.

CDFS1INT B'10000000'
Unsolicited send for intervention conditions

CDFS1ETE B'01000000'
Support for environmental type errors

CDFS1A31 B'00100000'
Support for AMODE 31

CDFS1ESS B'00010000'
Support for ESS keywords

CDFS14DG B'00001000'
Support for 4-digit hexadecimal device numbers

Issuing the FSA Level FSIREQ CONNECT Request

When the FSA has completed initializing the CONNECT parameter list, it uses the FSIREQ macro to invoke the FSI CONNECT service. The format of this macro call is:

```
FSIREQ REQUEST=FSICON,PARM=CONNECT parm-list-address,
       TARGET=JES,FSID=fsid
```

Refer to Chapter 4, “The FSIREQ Macro,” on page 13 for a complete description of each operand on this macro and any defaults that you can take.

FSA CONNECT processing

When JES receives the FSA CONNECT request from the FSA, JES validates the FSA information and builds FSI-related control blocks for use by both the FSS and JES.

JES initializes the second FSA FSCT with the addresses of the FSI service routines that JES provides. When the FSA issues a request, the FSIREQ macro uses these addresses to branch into the appropriate JES-provided routines.

How JES Handles Logic Errors and Abends

JES may not be able to connect the FSA for one of the following reasons:

- The parameter list is incorrect
- The function code is invalid

- The FSS identifier or the FSA identifier is invalid
- The FSA is trying to connect before the FSA is fully connected
- The FSA is already connected

If JES could not connect the FSA, the value in register 15 is non-zero to indicate that the FSA should abnormally terminate. The FSS should correct whatever caused the error and reissue the FSA CONNECT request.

How JES Monitors Timing of FSA CONNECT

When JES issues the START FSA order to the FSS, it starts a timer. If the FSA does not respond with a FSA CONNECT within five minutes JES issues a STOP FSA order to the FSS. Refer to Chapter 12, "Stopping an FSA," on page 107 for more information about the STOP FSA order.

FSA Could Not Be Started

Depending on why the FSA could not be started, either the FSS or the FSA itself notifies JES. The FSS can decide in its order routine that the FSA START order is to be rejected. In this case, the FSS sets a non-zero return code in register 15 in response to the order. If the FSS does this, JES will destroy the address space that the FSS is running in.

If the FSS determines in its mainline code that it could not start the FSA, the FSS indicates this condition in the order response area (IAZRESPA). The FSS then issues an FSIREQ SEND request in response to the START FSA order to notify JES that the FSA could not be started.

If the FSA determines that it cannot issue the FSA CONNECT, it will notify JES by issuing a FSIREQ SEND function call with RESPRETC set to a non-zero value.

Chapter 8. Starting an FSS Device

Successful completion of FSA level CONNECT processing causes JES to issue the START device order to the FSA. When the FSA receives the START device order, it performs device processing and then responds to JES. IBM recommends that all device initialization be done by the START FSA routine. Therefore, the START device routine is the signal for the FSA to begin issuing GETDS requests. Once the device is started, it can begin requesting data sets from JES for output processing.

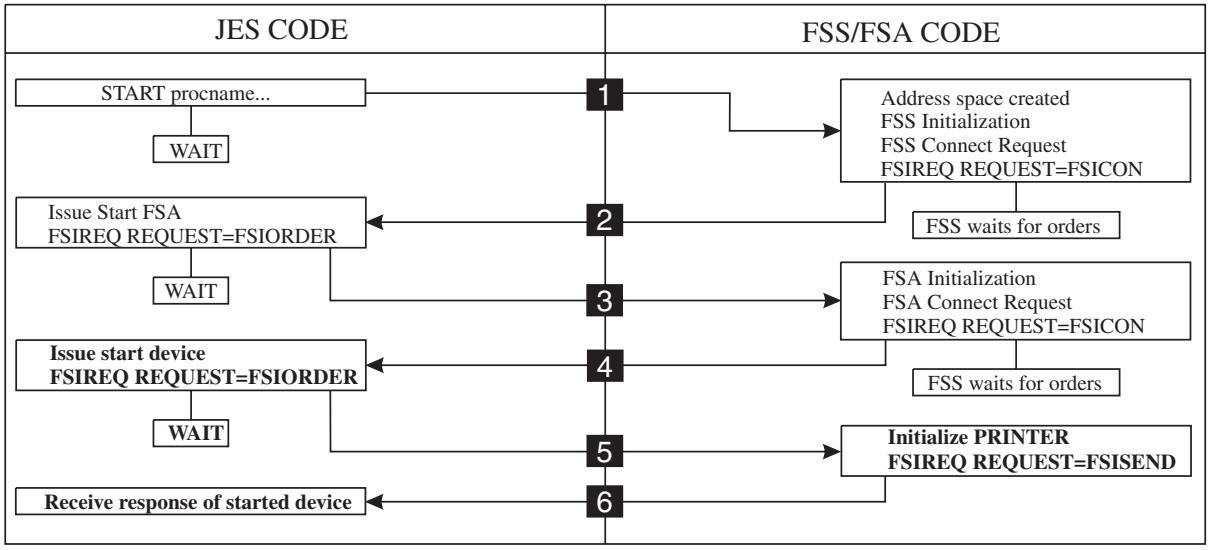

Figure 15. An Overview of FSI Startup Processing

The topics that follow explain how the FSA processes the START device order and responds to JES.

Processing the START Device Order

To start a device that is running under control of an FSS, JES issues the start device order to the FSA's FSI ORDER routine. JES passes the address of the START device order parameter list in register 1. Refer to Figure 16 on page 48 for a description of the START device parameter list.

When the FSI ORDER routine receives the order, it:

- Determines the type of order issued
- Either processes the order immediately or posts the appropriate FSA task to process the order.

The value of the ORDID field in the common order header section of the START device order parameter list represents the type of order the FSA needs to process.

Note: If the order is something that may take a while and therefore can be answered asynchronously, IBM recommends that the FSA order routine notify the FSA that there is an order to process and immediately return control to JES. The FSI order and post routines are part of the FSS and FSA. However, JES invokes the FSI order and post routines and they run under a JES TCB or SRB.

The START device order parameter list consists of the following sections:

- Common parameter header
- Common order header
- START order function dependent section.

The following figure shows the connection between the different sections of the FSIREQ parameter list.

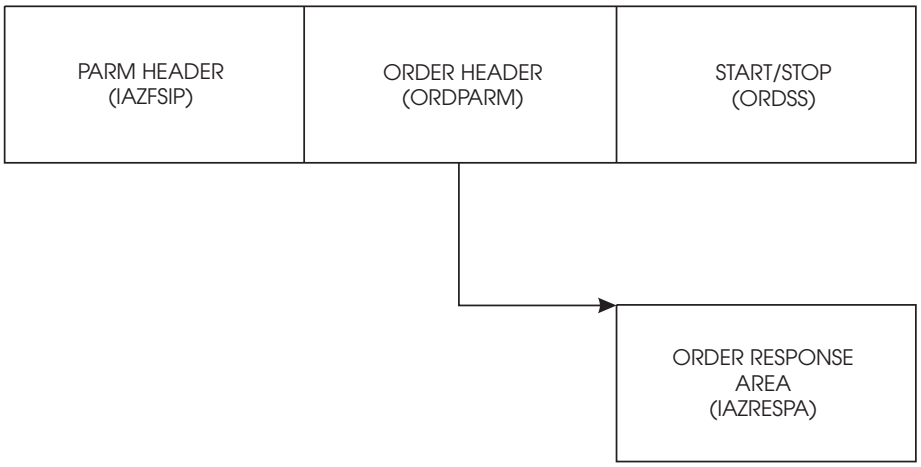

Figure 16. FSIREQ Parameter Lists for the Start Device Order

The following table shows the parameters that JES initializes for the START device order. The values that JES assigns are explained after the table.

Field Name	Length (bytes)	Value JES Assigned
Common Parameter Header		
FSILEN	4	Length of START order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS/FSA identifier
Common Order Header		
ORDFDATA	4	Information supplied to JES in the FSS/FSA CONNECT parameter list (CDFFDATA)
ORDRSPAD	4	Address of the order response area
ORDID	2	ORDSTDEV
START Order Function Dependent Section		
ORDSSSP	4	0
ORDSSID	4	FSA identifier of device to start
ORDSSAD4	4	Device address in 4-digit format
ORDSSAD	3	Device address in 3-digit format
ORDSSNA	8	Device name

FSILEN
The total length of the START order parameter list. The START order parameter list consists of the common parameter header, the common order header and the START order header.

FSIFUNC
The ORDER ID number. JES assigns the symbolic value FSIORDER to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSA-related information. The FSA passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value in the START device order parameter list so that the FSA's FSI ORDER routine can start the appropriate FSA task to process the order.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The START device order ID number. JES assigns the symbolic value ORDSTDEV to this field.

ORDSSSP
This field is set to zero. JES supplies the address of the device initialization area in this field for the START FSA order only.

ORDSSID
The FSS/FSA identifier of the device to start.

ORDSSSI
The FSS section of the FSA identifier.

ORDSSAI
The FSA section of the FSA identifier.

ORDSSAD4
The 4-digit device address in printable form. This field will contain blanks if the printer is a non-channel attached device.

ORDSSAD
The 3-digit device address in printable form. This field will contain blanks if the printer is a non-channel attached device.

ORDSSNA
The device name in printable form.

Notifying JES of Device Status

When the FSA's FSI order routine receives the START device order from JES, the FSA decides whether it can start the device immediately, or needs to perform additional processing before starting the device. Refer to Chapter 5, “FSI Communication,” on page 17 for information about responding to an order from JES.

SEND Processing

When JES receives the SEND request, it processes the return code set by the FSA in the RESPRETC field of the order response area. If the return code is zero, JES is ready to accept GETDS requests. If the return code is non-zero, JES issues a STOP FSA order. Refer to Chapter 12, “Stopping an FSA,” on page 107 for more information about the STOP FSA order.

Chapter 9. Issuing Data Requests to JES

After an FSA notifies JES (using the FSI SEND request) that it successfully started the associated device, it is ready to begin data set processing. As part of data set processing, the FSA invokes the FSI data access services (GETDS, GETREC, FREEREC, RELDS, and CKPT) to:

- Obtain a SYSOUT data set and its characteristics from JES, as described in "Getting a SYSOUT Data Set (GETDS)"
- Obtain logical records of an obtained data set, as described in "Getting SYSOUT Records from an Acquired Data Set" on page 70
- Release logical records for a data set to JES, as described in "Releasing a SYSOUT Record" on page 77
- Release an obtained data set to JES, as described in "Releasing a SYSOUT Data Set" on page 80
- Request JES to record checkpoint information for a JES spool data set currently being processed by the FSA device. as described in "Requesting a Checkpoint of Processing" on page 84.

The information provided for each of the FSI data access services on the following pages explain:

- The tasks required to invoke the FSI service
- The FSI service processing
- The information that JES returns in response to the FSIREQ request.

JES groups similar data sets together and prints them between a set of separator pages. A header separator page starts a group and a trailer separator page ends a group. Therefore, when the FSA receives a request from JES to process a data set, the request will include separator page information related to that data set's position within the group.

For example, the first data set in a group will have a header separator page and the last data set in a group will have a trailer separator page. Data sets between the first and the last will not have header or trailer separator pages.

Getting a SYSOUT Data Set (GETDS)

An FSA obtains a JES spool data set and its characteristics for output processing by invoking the FSI GETDS service. The following are places the FSA gets data set characteristics in addition to the characteristics it gets during GETDS processing:

- The job separator page area (JSPA) whose address is in the GETDS parameter list.
- The job management record (JMR) whose address is in the JSPA.
- The scheduler work blocks whose address is in the GETDS parameter list. This is the major place to find basic SYSOUT attributes from the end user's JCL and installation defaults.
- The device settings, set explicitly from the START FSA order and possible reset by using the SET order.
- The device setting that might be implicitly set by the FSA due to the device name or UCB name that JES passes.

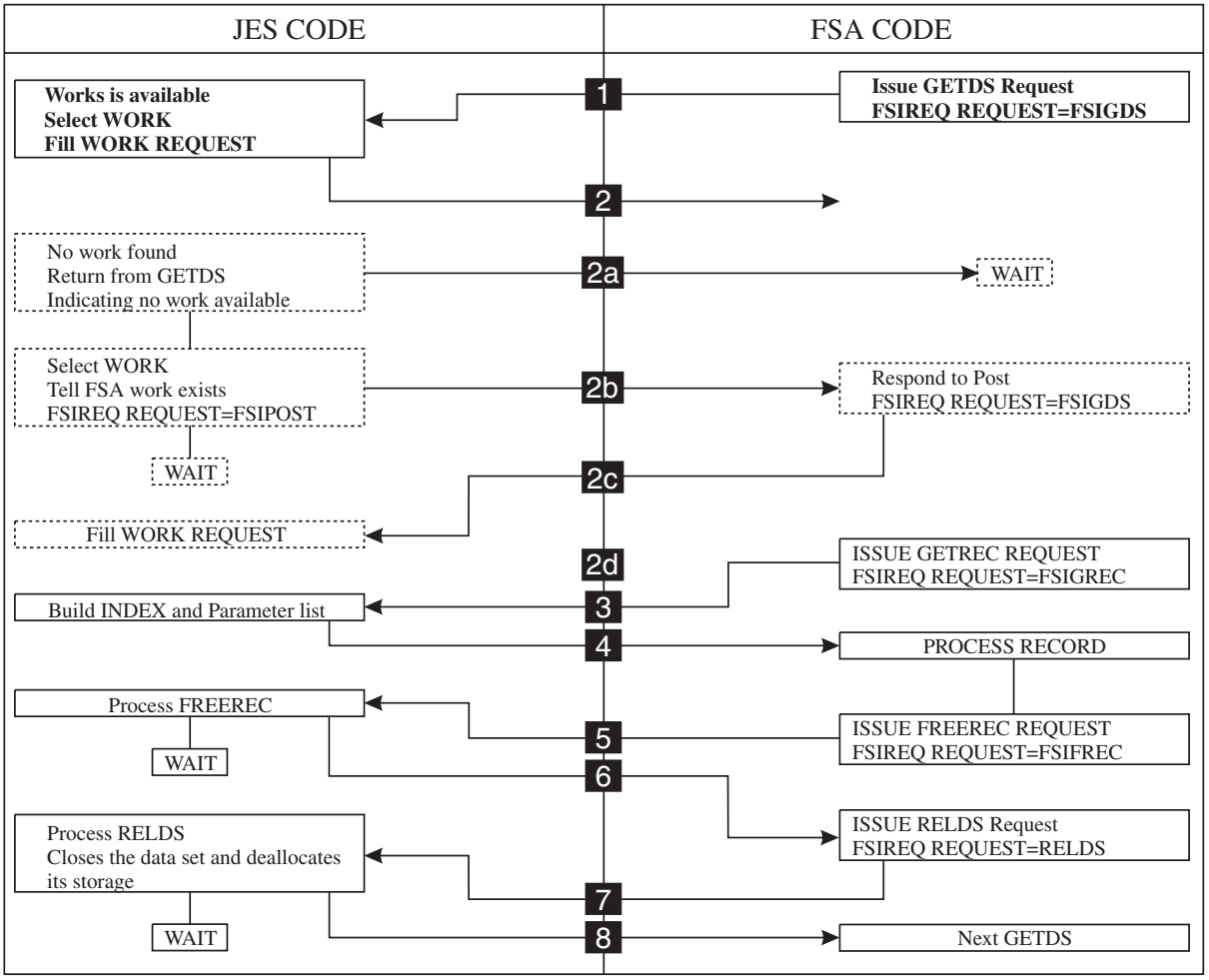

Figure 17. An Overview of FSI Data Set Processing

The FSI GETDS service is functionally equivalent to allocating and opening a SYSOUT data set. The FSA does not specify data set selection criteria in the GETDS request; it makes a request for the next available data set. JES uses its own work selection criteria to provide the most appropriate data set to the FSA. If no data set is available for processing, JES notifies the FSA that it could not satisfy the GETDS request. The FSA should not issue any more GETDS requests until the FSA is notified that work is available. When work becomes available, JES notifies the FSA via the FSIREQ POST function that it can reissue the GETDS request.

JES does not restrict the number of data sets that can be allocated to the FSA concurrently. Thus, the FSA can issue multiple GETDS requests without intervening RELDS requests. However, in a JES3 environment all GETREC requests will be satisfied from the last GETDS request.

The following figure shows the connection between the different sections of the FSIREQ parameter list for GETDS processing. The individual parts of this diagram are explained in this section.

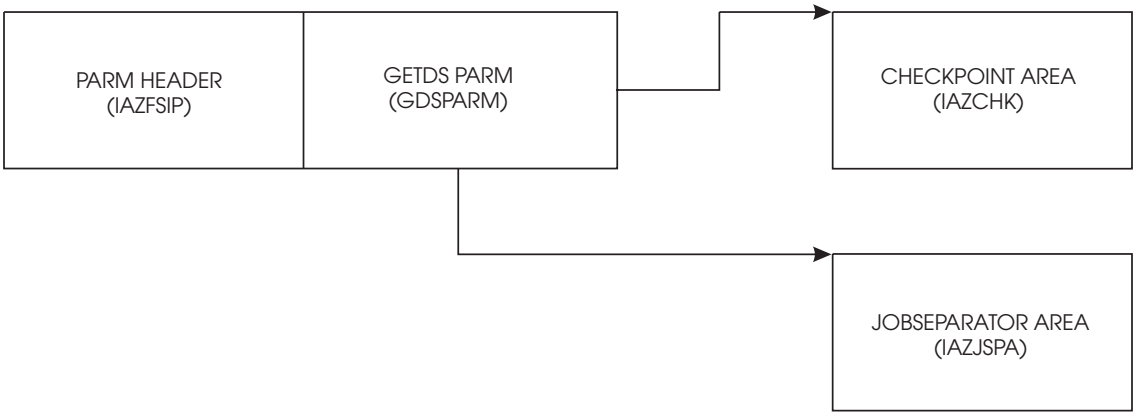

Figure 18. FSIREQ Parameter Lists GETDS Processing

The following sections explain the tasks the FSA must complete to invoke the FSI GETDS service.

Providing an FSA Checkpoint Area

The FSA must place the address of a checkpoint area in the GETDS parameter list. JES uses this area to return information that allows a previously interrupted data set to continue printing from the point indicated by the last valid checkpoint. When the FSA invokes the FSI GETDS service, JES retrieves any checkpoint information for the data set assigned to the FSA and moves that information into the FSA-provided checkpoint area. If JES has filled in the checkpoint area, the GDSCKP bit is turned on.

When the FSA establishes a checkpoint area, the address of the checkpoint area should be placed in the GDSCKPA field. The length of the checkpoint area should be stored in the GDSCKPL field.

IBM recommends that the size of this checkpoint area be large enough to accommodate both the IAZCHK FSI checkpoint record and any FSA device dependent checkpoint information. The IAZCHK macro is the JES base checkpoint information mapping. Any device dependent information must be placed at the end of the JES base. The length that is stored in GDSCKPL is the combination of both the JES base and device dependent sections.

The FSA should establish (GETMAIN) a unique checkpoint area for each concurrently active data set that it is processing. For example, if the FSA issues one GETDS, processes all the records, and then releases the data set, only one checkpoint area is needed. If the FSA issues several GETDS requests and processes them at the same time, several checkpoint areas are needed.

Initializing the GETDS Parameter List

Both the FSA and JES use the FSIREQ GETDS parameter list to pass information to one another. The FSA must initialize certain fields of the FSIREQ GETDS parameter list for each issuance of the GETDS request. The following table lists the required

fields, the lengths of these fields, and the values that the FSA must assign. Detailed descriptions of the value assignments follow this table.

Table 3. FSIREQ GETDS parameter values

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of GETDS parameter list
FSIFUNC	4	FSIGDS
FSIFSID	4	The FSS/FSA IDs
FSIPEXT	4	The extension area address
GETDS Function Dependent Section		
GDSCKPL	4	Length of FSA checkpoint area
GDSCKPA	4	Address of FSA checkpoint area
GDSDSID	12	0 (zero)
GETDS Function Dependent Extension Area Header		
FSIEXNUM	2	Number of extensions
FSIEXLEN	2	Length of all extensions
FSIEHID	4	Extension header ID
GETDS Function Dependent Extension Area		
FSIEGLEN	2	Extension area length
FSIEGFID	4	Extension function ID

FSILEN
The length of the entire GETDS parameter list. The GETDS parameter list consists of both the IAZFSIP common header section and the GETDS function dependent section.

FSIFUNC
The GETDS function ID number. The FSA assigns the symbolic equate value FSIGDS to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

FSIPEXT
If this field is non-zero, then there is an existing extension to this parameter list. The address of the extension is the contents of this field. See the Appendix for structure of extension area.

GDSCKPL
The length of the FSA checkpoint area.

GDSCKPA
The address of the FSA checkpoint area.

GDSDSID
The FSA must clear this field to zero before each issuance of the GETDS request because JES assigns the data set identifier to this field.

FSIEXNUM
The number of function dependent extension areas following this header.

FSIEXLEN
The length of the function dependent extension areas. This length does not include the length of this header.

FSIEHID
The extension header ID. This is the character string "EHID".

FSIEGLEN
The length of this function dependent extension. Set this field to the symbolic equate value FSIEASZE.

FSIEGFID
The function ID of this request. Set this field should the symbolic equate value FSIGDS.

Issuing the FSIREQ GETDS Request

When the FSA has completed initializing the GETDS parameter list, it issues the FSIREQ macro to invoke the FSI GETDS service. The format of this macro call is:

```
FSIREQ REQUEST=FSIGDS,TARGET=JES,PARM=GETDS parm-list-addr, FSID=value-addr
```

See Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

JES GETDS Processing

The JES-supplied GETDS routine in the FSS address space receives control when the FSA issues the FSIREQ GETDS macro. This routine communicates with the JES address space to process GETDS requests. The basic function of GETDS processing is to attempt to satisfy the GETDS request immediately by selecting a JES output data set and then despooling that data set to the FSA. JES uses its own data set selection criteria to select the appropriate data set.

If no errors occur during GETDS processing and a data set is available, JES retrieves any checkpoint information for the data set and moves that information into the checkpoint area provided by the FSA. JES also determines the JES characteristics and retrieves a pointer to the scheduler work blocks for this data set. The scheduler work blocks represent the data set's characteristics that were specified in the job's JCL. Finally, JES initializes the GETDS parameter list with the data set information and sets a return code of zero in register 15. JES then returns control to the FSA.

If no errors occur during GETDS processing but a data set is not available, JES sets the GDSNALLC flag on in the GETDS parameter list and sets a return code of zero in register 15. JES then returns control to the FSA. See "No Work Exists for Printing" on page 66 for more information.

If an error occurs during GETDS processing (for example, JES detects that the length of the GETDS parameter list is invalid), JES sets the GDSNALLC flag on in the GETDS parameter list and sets a non-zero return code in register 15. JES then returns control to the FSA. When the FSA receives a non-zero return code it should abnormally terminate and take a dump.

Information Returned from GETDS Processing

On return from successful GETDS processing, the GETDS parameter list contains the following information:

Field Name	Length (bytes)	Value Assigned
Common Parameter Header Section		
FSILEN *	4	Length of the GETDS parameter list
FSIFUNC *	4	FSIGDS
FSIFSID *	4	The FSS/FSA IDs
FSIPEXT*	4	The extension area address
GETDS Function Dependent Section		
GDSFLGR1	1	JES printing requirements for the data set
GDSFLGR2	1	SWB requirements for job header/trailer pages
GDSFLGS1	1	GETDS processing status information
GDSCKPL *	4	Length of FSA checkpoint area
GDSCKPA *	4	Address of the FSA checkpoint area
GDSJSPA	4	A pointer to the JSPA
GDSOUTK	8	The OUTPUT SWB token
GDSJDVTN	8	The JDVT name used at data set creation
GDSDSID *	12	The data set identifier
GDSRECFM	1	The data set record format
GDSMRECL	2	The data set record length
GDSSJMSG	80	The SJF error message. This field is initialized only if the GDSFLGS1 flag byte indicates that an error occurred in SJF processing.
GETDS Function Dependent Extension Area		
FSIEGLEN*	2	Extension area length
FSIEGVSN	2	Version number field
FSIEGFID*	4	Extension function ID
FSIEGUTK	80	User token
FSIEGRTK	80	Resource token
FSIEGOGT	20	Output group token

The fields with an asterisk (*) contain values set by the FSA when it issued the GETDS request. The fields that JES set or reset during GETDS processing are described in detail below:

GDSFLGR1
This flag byte contains the JES printing requirements for the data set returned to the FSA. The following indicators may be set:

GDSJHDR B'10000000'
JES requires the FSA to print a job header page for the data set.

GDSJTRL B'01000000'
JES requires the FSA to print a job trailer page for the data set.

Note: JES may optionally issue a SYNCH order to request a job trailer page for the data set.

GDSHDR B'00100000'
JES requires the FSA to print a data set header page.

GDSHTDS B'00010000'
JES requires the FSA to print the data set on the same page as the job header or trailer page. If this flag is set, either the job header or job trailer flag is also set, but never both. JES sets this flag only if it has assigned the JESNEWS data set to the FSA.

GDSFRMRK B'00001000'
JES requires a form mark on the separator page.

GDSCMC B'00000100'
JES requires the FSA to change the copy mark for each data set. For a stacking machine, a change of the copymark is equivalent to an offset of the paper. For a machine without a stacker, the copymark is a black tickmark on the bottom of the page.

GDSCMCPY B'00000010'
JES requires the FSA to change the copy mark for each copy.

GDSTRKDS B'00000001'
JES requires the FSA to track the data set and issue an FSIREQ SEND request when the data set reaches the operator observation point. See "Notifying JES that the Data Set Reached the OOP" on page 68 for information about handling this requirement.

GDSFLGR2
This flag byte contains the installation defined printing requirements for the data set returned. The following indicators may be set:

GDSJHSWB B'10000000'
The FSA is to use FSA header defaults, if they exist, defined for the job header page when printing the data set. JES sets this flag only for the JESNEWS data set.

GDSJTSWB B'01000000'
The FSA is to use FSA trailer defaults, if they exist, defined for the job trailer page when printing the data set.

GDS2EOG B'00100000'
End of output group.

GDSFLGS1
This flag byte contains status information related to GETDS processing. The following indicators may be set:

GDSCKP B'10000000'
The checkpoint area contains valid information that the FSA may use to restart the processing of a previously interrupted data set. See "Information Contained in the FSA Checkpoint Area" on page 65 for a description of each field.

GDSALLOC B'01000000'
JES successfully allocated a data set to the FSA.

GDSRSTCT B'00000100'
JES requires the FSA to reset the group page and record counts that the FSA keeps track of for the QUERY order. See "The Query Order" on page 89 for more information about information returned to JES for a QUERY order.

GDSSJERR B'00000010'
The JES GETDS service routine detected an error in scheduler JCL facility (SJF) processing. The GDSSJMSG field contains a detailed error message that the FSA is to display.

GDSJSPA
A pointer to a job separator page data area (JSPA). The JSPA contains job and data set related information that the FSA may use to generate header and trailer pages (if required), and SMF Type 6 records. The section "Information Contained in the JSPA" shows the possible settings for each JSPA field.

GDSOUTPK
The OUTPUT SWB token. The FSA uses this token to interface with the scheduler JCL facility (SJF) to acquire the data set's characteristics specified on the JCL OUTPUT statement. "The Scheduler JCL Facility" on page 117 describes how to invoke SJF services and retrieve JCL data set characteristics.

GDSJDVTN
The JCL definition vector table (JDVT) name used at data set creation. The FSA uses this table to let the Scheduler JCL Facility (SJF) know what JCL to use for the SJF RETRIEVE service. See "Using SJF Services" on page 118 for more information about the SJF RETRIEVE service.

GDSDSID
The data set identifier. The FSA uses this identifier in subsequent FSI service requests (GETREC, FREEREC, RELDS, CHKPT, and ORDER) to uniquely identify the data set.

GDSRECFM
The data set record format as defined in the JFCB.

GDSMRECL
The data set record length. This is the largest record length with which the data set was opened.

GDSSJMSG
The message text describing the SJF error that occurred. JES initializes this field only if the GDSSJERR indicator is set in the GDSFLGS1 flag byte indicating that an error occurred in SJF processing. The FSA is to print this error message with the data set.

FSIEGLEN
The length of the extension area.

FSIEGVSN
The version number of the extension area.

FSIEGFID
The function ID for which this extension is created.

FSIEGUTK
This field contains the security token for the data set's. creator

FSIEGRTK
This field contains the security token of the data set.

FSIEGOGT
This field contains a number that uniquely identifies an output group.

Information Contained in the JSPA

When JES returns control to the FSA, it indicates in the GETDS parameter list the job header, job trailer, and data set header requirements. It also provides a pointer

in field GDSJSPA to the JSPA created for the assigned data set regardless of whether or not a separator page is requested by JES.

The JSPA contains job and data set related information that the FSA may use to generate the header and trailer pages. JES does not make requirements as to what information from the JSPA should be included on these pages.

The FSA determines how it will create the separator pages, and may freely use any or all fields passed to it in the JSPA for those pages. The FSA may also use the job related information in the JSPA to generate an SMF type 6 record for the assigned data set.

The JSPA consists of:

- A common JES section,
- A JES dependent section, and
- A user dependent section.

The IAZJSPA is returned to the caller above or below the 16 megabyte line based on the connect parameters supplied by the FSS (Bit CDFFL331 in byte CDFFLGR3 is supplied by the FSS during FSS connect to indicate if the FSS is running AMODE(31). The structure of the IAZJSPA control block may be seen in Figure 19 on page 60.

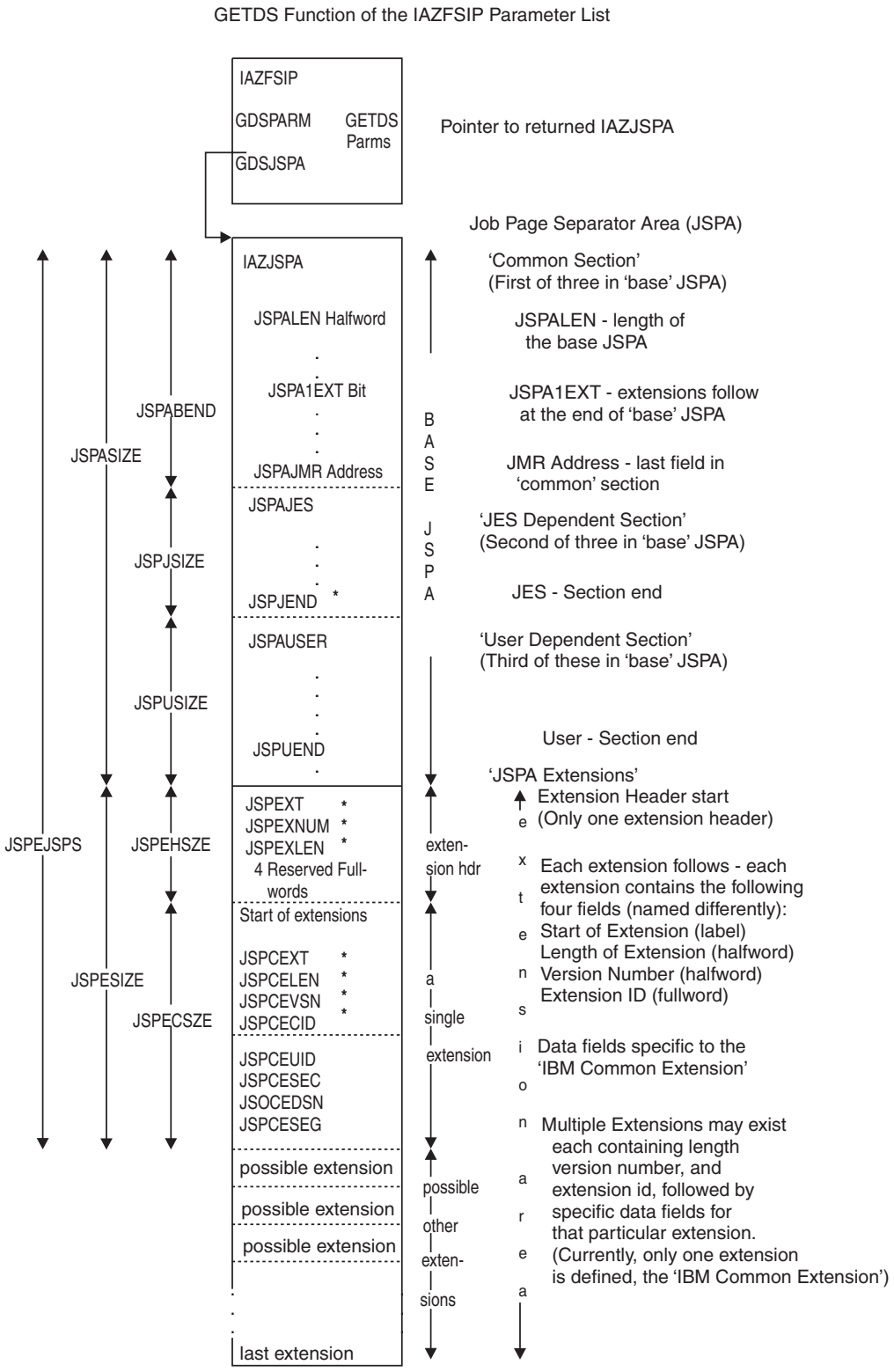

Figure 19. The IAZJSPA (Job Separator Page Area)

It also may contain one or more data extensions which contain additional information about the returned data set.. The fields in the JES dependent section may or may not be set depending on whether the JES connected to this FSA scans

for their associated information and whether the information was provided (for example, the programmer name may not have been specified).

In addition, the JES3 user exit IATUX45 allows the user to modify the information in the JES sections and/or expand the JSPA with user defined information, while the JES2 user exit 23 allows the user to modify or expand the user section of the JSPA. If the FSA is to take advantage of information in the user dependent section, it must provide its own user exits. Otherwise it is concerned only with the information in the JES sections. For more information about IATUX45 or user exit 23, see the appropriate JES Customization book.

The pointer out of the GDSPARM area is shown, along with the possible 3 sections of the BASE JSPA, the Extension Header, and the mapping of the 'IBM Common Extension'. The equated sizes are shown, but are only accurate at assembly-time.

To obtain the start of the extension area (JSPEXT), the user should add the contents of field JSPALEN to the starting address of the JSPA. Then, the extension header is a fixed size, and field JSPEXLEN contains the lengths of all extensions (not including the extension header).

The following table lists the individual JSPA fields and the lengths of these fields, and the values that may have been set. Any fields not containing values are set to binary zeroes unless otherwise noted.

Table 4. IAZJSPA Macro

Field Name	Length (bytes)	Assigned Value
Common Section (All fields are set by either JES2 or JES3)		
IAZJSPA (JSPA)	0	DSECT mapping name
JSPAID	4	JSPA
JSPALEN	2	Length of the IAZJSPA base section
JSPAFLG1	1	Flag byte
JSPAJBNM	8	Job name
JSPAJBID	8	Job id
JSPADEVN	8	Device name assigned to the FSA that is associated with the device being used to process the returned data set.
JSPADEVA	4	The 3-character or 4-character device address in EBCDIC of the device named in JSPADEVN. See explanation below.
JSPAJMR	4	Address of the JMR (job management record)
JES Dependent Section (The values are determined by JES)		
JSPAJES	0	Start of JES dependent data area
JSPJGRPN	8	Output group name from the job output element (JOE)
JSPJGRP1	2	Output group first ID from the job output element (JOE)
JSPJGRP2	2	Output group second ID from the job output element (JOE)
JSPJGRPD	8	Output group's DESTID
JSPJRMNO	4	Room number from the JCT associated with the owning job

Table 4. IAZJSPA Macro (continued)

Field Name	Length (bytes)	Assigned Value
JSPJPNAM	20	Programmer name from the JOB statement
JSPJDSNM	24	Three part DD name assigned by JES to the returned SYSOUT data set
JSPJDSPN	8	Procedure name component of JSPJDSPN
JSPJDSSN	8	Step name component of JSPJDSPN
JSPJDSDD	8	DD name component of JSPJDSPN
JSPJSOCL	1	SYSOUT class of the data set
JSPJPRIO	1	Priority of the data set
User Dependent Section		
JSPAUSR1	4	Reserved for the user
JSPAUSR2	4	Reserved for the user
Mappings of the extensions		
JSPEXT	0	Start of the extension area
JSPEXNUM	2	Number of extensions
JSPEXLEN	2	Length of all extensions
Mappings of IBM Common Extension		
JSPCEXT	0	Start of the IBM Common Extension
JSPCELEN	2	Length of IBM Common Extension
JSPCEVSN	2	Version number of this extension
JSPCECID	4	Id of this extension
First data field of the extension at offset +8		
JSPCEUID	8	Userid associated with this data set
JSPCESEC	8	Security label (SECLABL) of this data set.
JSPCEDSN	53	Fully-qualified (including node name) data set entity name
JSPCESEG	4	Segment number associated with this data set

IAZJSPA (JSPA)
The DSECT mapping name.

JSPAID
JSPA

JSPALEN
The length of the IAZJSPA base section not including the extension header or any extensions. If JSPA1EXT is set, the value in this field added to the starting address of IAZJSPA is the address of the extension header. This value is used to obtain the address of JSPEXT.

JSPAFLG1
The flag byte is defined as follows:

- JSPA1CON identifies this data set as a continuation of a previously-passed output group. The bit may be used to signify that this output group has may have portion(s) previously returned to a print device and that this particular returned data set might not start at the beginning of the data set.
- JSPA1EXT signals that one or more extensions follow the base IAZJSPA.

- JSPA1UND signals that the user id contained in field JSPCEUID is undefined and is not a valid user id.
- JSPA4DG signals that the device number is in 4-digit format.

JSPAJBNM
The job name that is assigned to the job that created this data set.

JSPAJBID
The job id assigned to the job that created this data set.

JSPADEVN
The device name assigned to the FSA that is associated with the device being used to process the returned data set.

In JES2, it is the PRT(nnnn) name of the device as assigned on the local printer initialization statement.

In JES3, it is the JNAME of the device as assigned on the DEVICE initialization statement.

JSPADEVA
The device address in EBCDIC of the device named in JSPADEVN. If JSPA4DG in JSPAFLG1 is ON, this field contains the 4-digit device number; otherwise, it contains the 3-digit device number in the first 3 bytes.

JSPAJMR
The address of the JMR (job management record) associated with the job that created this data set. It is mapped by IEFJMR.

JSPAJES
The start of JES dependent data area.

JSPJGRPN
JES2 supplies the output group name from the job output element (JOE). JES3 does not use this field.

JSPJGRP1
JES2 supplies the output group first ID from the job output element (JOE). JES3 does not use this field.

JSPJGRP2
JES2 supplies the output group second ID from the job output element (JOE). JES3 does not use this field.

JSPJGRPD
JES2 supplies the output group's DESTID.

JSPJRMNO
JES2 supplies the room number from the JCT associated with the owning job. JES3 does not use this field.

JSPJPNAM
The programmer name from the JOB statement, if available, or blanks (X'40's).

JSPJDSNM
The three part DD name assigned by JES to the returned SYSOUT data set (a combination of JSPJDSPN, JSPJDSSN, and JSPJDSDD).

JSPJDSPN
The procedure name component of JSPJDSPN.

JSPJDSSN
The step name component of JSPJDSPN.

JSPJDSDD
The DD name component of JSPJDSPN.

JSPJSOCL
The SYSOUT class of the data set.

JSJPRIO
The priority of the data set.

JSPAUSR1
This field is reserved for the user.

JSPAUSR2
This field is reserved for the user.

JSPEXT
The start of the extension area. The extension header follows. The header and any extensions are only present if JSPA1EXT has been set in the base section. See the description for field, JSPALEN in determining where the extension header exists in storage.

JSPEXNUM
The number of extensions following this header.

JSPEXLEN
The length of all extensions following this header not including the extension header itself.

JSPCEXT
The start of the IBM Common Extension. Extensions must have the same first three fields containing the extension length, version, and id.

JSPCELEN
The length of IBM Common Extension. All extensions must have a 2 byte field containing the length at this offset (+0) into an extension.

JSPCEVSN
The version number of this extension. All extensions must have a 2 byte field containing the version at this offset (+2) into an extension. The following versions exist:

1 - Reserved

2 - The following data fields are defined in the extensions: JSPCEUID, JSPCESEC, and JSPCEDSN.

3 - All version 2 data fields exist plus JSPCESEG are defined as well.

JSPCECID
The id of this extension JSPCEXTI indicates this particular extension is the IBM Common Extension. All extension's data fields at offset +8 and beyond are uniquely defined for any particular extension and begin after the id of the extension. All extensions must have a 4 byte field containing the id at this offset (+4) into an extension.

JSPCEUID
The userid associated with this data set. This value is available only if JSPCEVSN is greater than or equal to 2.

JSPCESEC
The security label (SECLABL) of this data set. This value is available only if JSPCEVSN is greater than or equal to 2.

JSPCEDSN
The fully-qualified (including node name) data set entity name in the format

nodename.userid.jobname.jobid.number.name. This value is available only if JSPCEVSN is greater than or equal to 2.

JSPCESEG
The segment number associated with this data set. This value is available only if JSPCEVSN is greater than or equal to 3. Segment id is valid only when using the SEGMENT= DD JCL keyword. JES3 does not support the SEGMENT= keyword on the DD JCL statement thus this field is set to binary zeroes in JES3.

Information Contained in the FSA Checkpoint Area

If valid checkpoint information exists for the data set assigned to the FSA, JES moves this information into the FSA checkpoint area during GETDS processing. The specific information provided depends on whether the data set was previously being processed by an FSS- or JES-controlled device.

- If the data set was previously being processed by a JES-controlled device, JES converts its own checkpoint data into the FSI checkpoint record format (IAZCHK) and moves the record into the FSA checkpoint area.
- If the data set was previously being processed by an FSS-controlled device, JES retrieves FSA-supplied checkpoint information that it recorded during FSI CHKPT processing and moves that into the FSA checkpoint area. In this case, the FSA checkpoint area contains the FSI checkpoint record (IAZCHK) (whose fields were set by an FSA) and any FSA device dependent checkpoint information.

The following table lists the fields of the IAZCHK record, the length of each field, and the values that may have been assigned.

Field Name	Length (bytes)	Assigned Value
CHKID	4	'CHK' (FSI checkpoint record identifier)
CHKLNGTH	2	Length of FSI checkpoint record
CHKJESWK	64	JES dependent checkpoint information for the data set. The FSA does not use this information.
CHKRBA	8	The JES equivalent of a relative block address (RBA). The FSA may use this address in a subsequent GETREC request to cause JES to begin accessing records at this address.
CHKDEV	4	The device type
CHKMOD	4	The model number of the device
CHKCOPY	4	The number of copies that have been printed
CHKTRNC	4	The transmission count
CHKREC	4	The count of spool records processed (line mode records with a length of zero or machine immediate carriage controls are not counted).
CHKPAGE	4	The physical page count
CHKPROD	8	The product that created the checkpoint record
CHKVER	4	The version of the product

Field Name	Length (bytes)	Assigned Value
CHKRELS	4	The release of the product
CHKMODF	4	The modification level of the product
CHKSERV	4	The service level of the product

No Work Exists for Printing

If JES cannot allocate a data set during GETDS processing, it sets the GDSNALLC flag on in the GETDS parameter list and then returns control to the FSA. The GDSNALLC flag indicates that no work is currently available and that JES will notify the FSA, via the FSIREQ POST function, when it can satisfy the GETDS request.

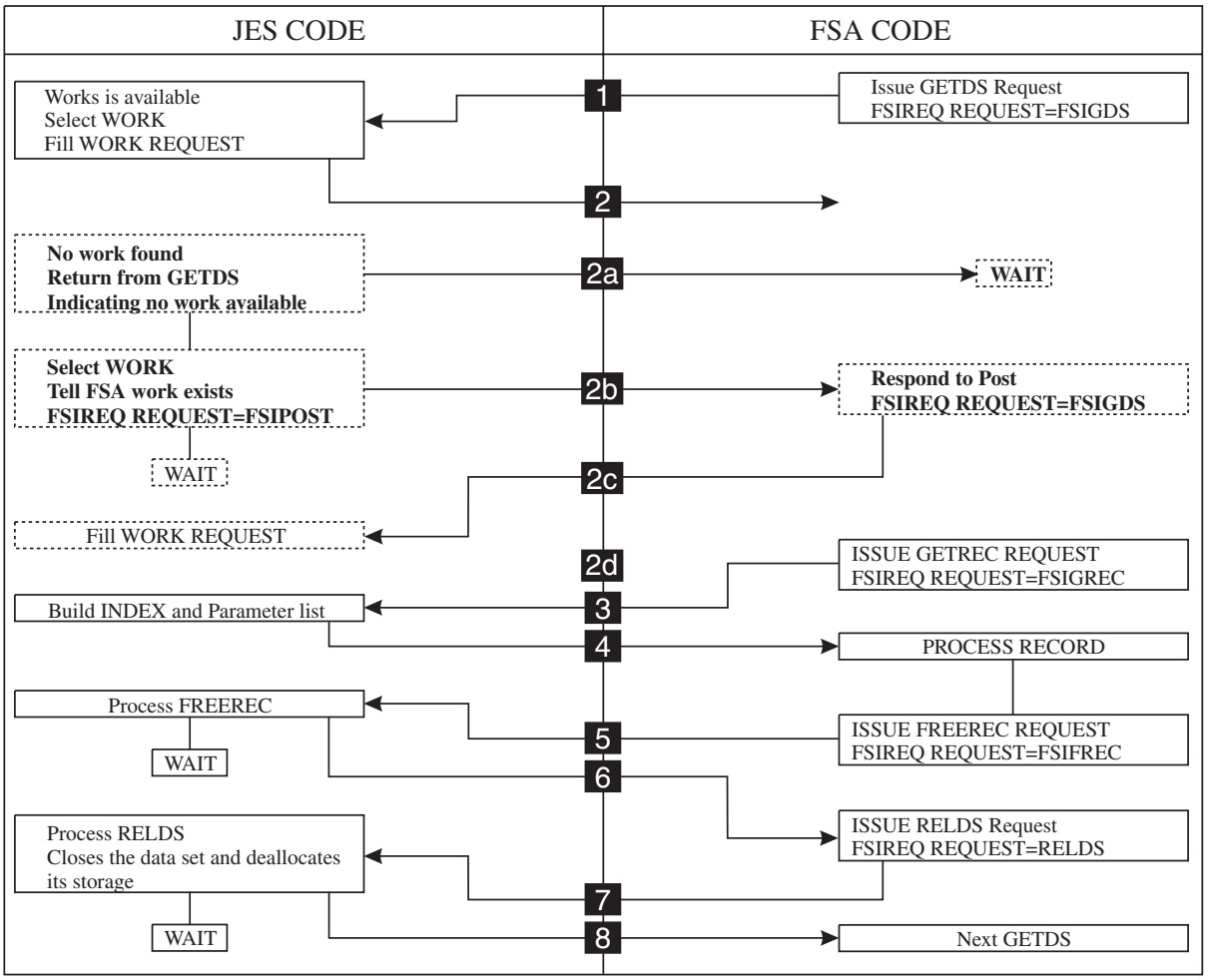

Figure 20. An Overview of Data Set Processing

The GETDS parameter list contains the following information:

Field Name	Length (bytes)	Value Assigned
Common Parameter Header Section		
FSILEN *	4	Length of the GETDS parameter list
FSIFUNC *	4	FSIGDS

Field Name	Length (bytes)	Value Assigned
FSIFSID *	4	The FSS/FSA IDs
GETDS Function Dependent Section		
GDSFLGS1	1	GDSNALLC (indicator for data set not allocated)
GDSCKPL *	4	Length of FSA checkpoint area
GDSCKPA *	4	Address of the FSA checkpoint area

The fields with an asterisk(*) contain values set by the FSA when it issued the GETDS request.

Notifying the FSA When Work Becomes Available

When JES determines that work is available, it notifies all FSAs that are waiting for a data set and are eligible to process the work. Specifically, for each FSA, JES issues an FSIREQ POST request to the FSA-supplied POST routine indicating that GETDS requests can now be satisfied and should be reissued. When the FSA POST routine receives the request, it is responsible for alerting the FSA which will cause it to issue another GETDS request.

The following figure shows the connection between the different sections of the FSIREQ parameter list for POST processing.

Figure 21. FSIREQ Parameter Lists for POST Processing

In the POST parameter list, JES passes the following information:

Field Name	Length (bytes)	Value Assigned
Common Parameter Header Section		
FSILEN	4	Length of the POST parameter list
FSIFUNC	4	FSIPOST
FSIFSID	4	The FSS/FSA IDs
POST Function Dependent Section		
POSTFLS1	1	POSTGDS
POSFDATA	4	CDFFDATA

FSILEN
The length of the POST parameter list, which consists of the common header section and the POST function dependent section.

FSIFUNC
The POST function ID number. The symbolic equate FSIPOST represents this value.

FSIFSID
The FSS/FSA IDs that JES assigned to the FSS/FSA during start up.

POSTFLS1
This status flag byte indicates the reason for the POST request. The following indicator is set:

POSTGDS B'10000000'
GETDS requests can now be satisfied.

POSFDATA
This field contains the value that the FSA passed to JES in the CDFFDATA field of the CONNECT parameter list.

Processing the FSIREQ POST Request

The FSA POST routine uses the information passed in the POST parameter list to activate the appropriate FSA. This information is pointed to by the POSFDATA field. This field is filled in from the CDFFDATA field during connect processing. If the POST processing is successful, the FSA POST is expected to return control to JES with a zero return code in register 15. If an error occurs during processing, the FSA POST routine is expected to set a non-zero return code in register 15 and then return control to JES. Upon receiving a non-zero return code, JES will abnormally terminate the FSS address space.

Notifying JES that the Data Set Reached the OOP

If JES sets the GDSTRKDS indicator on in the GDSFLGR1 flag byte in the GETDS parameter list, the FSA is required to track the processing of the data set and then notify JES when the data set reaches the operator observation point (OOP). JES expects the FSA to issue an unsolicited FSIREQ SEND request and provide status information in a response area. In this instance however, JES has **not passed** the address of the response area (IAZRESPA). The FSA must format its own response area according to the IAZRESPA mapping macro and provide its address in the FSIREQ SEND parameter list.

Initializing the Order Response Area

The following table lists the IAZRESPA fields that require initialization, the length of each field, and the values that the FSA must assign to those fields.

Field Name	Length (bytes)	Value to be assigned
Response Area Mapped by IAZRESPA		
RESPID	4	'RESP' (ID of response area)
RESPLEN	4	Length of response area
RESPOOPI	12	The identifier of the data set at the OOP

Initializing the SEND Parameter List

The following figure shows the connection between the different sections of the FSIREQ parameter list for Send processing.

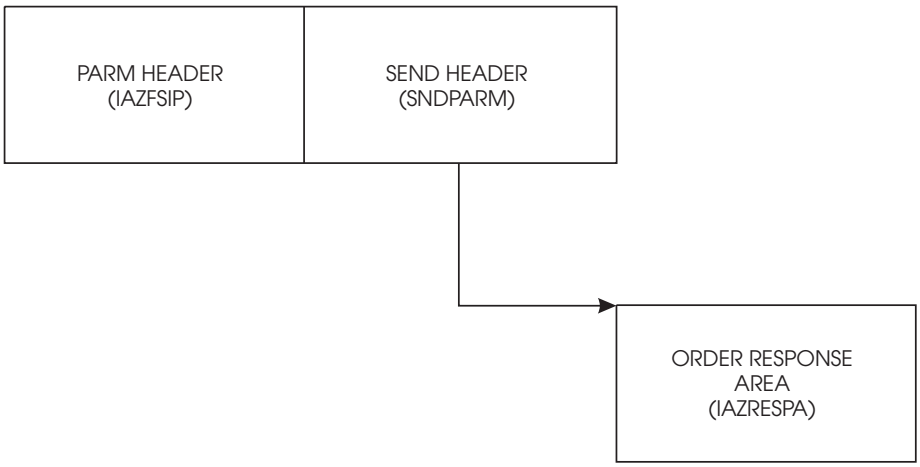

Figure 22. FSIREQ Parameter Lists for Send Processing

The following table below lists the fields in the SEND parameter list that require initialization, the length of each field, and the values that the FSA must assign to those fields. Detailed value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of SEND parameter list
FSIFUNC	4	FSISEND
FSIFSID	4	The FSS/FSA IDs
SEND Function Dependent Section		
SNDTYPE	1	SNDTYTDS
SNDRSPTR	4	The address of the response area

FSILEN
The length of the entire SEND parameter list. The SEND parameter list consists of both the IAZFSIP common header section and the SEND function dependent section.

FSIFUNC
The SEND function ID number. The FSA assigns the symbolic equate value FSISEND to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

SNDTYPE
The FSA uses this flag byte to indicate to JES the type of information being sent. For this issuance of the SEND request, the FSA is expected to set the following indicator:

SNDTYTDS B'01000000'
The FSA is satisfying JES's request for notification (GDSTRKDS) when the data set reaches the OOP.

SNDRSPTR
The address of the FSA-provided response area.

Issuing the FSIREQ SEND Request

When the FSA has completed initializing the response area and SEND parameter list, it issues the FSIREQ macro to invoke the FSI SEND communication service. The format of this macro call is:

```
FSIREQ REQUEST=FSISEND,TARGET=JES,PARM=SEND
parm-list-addr,FSID=value-addr
```

Note: See Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

On return from SEND processing, register 15 contains either a zero return code indicating success or a non-zero return code indicating an error occurred during processing.

Getting SYSOUT Records from an Acquired Data Set

Once an FSA has obtained a data set with a GETDS request, it can use the data set identifier (GDSDSID) returned to invoke the FSI GETREC service. The FSI GETREC service acquires one or more logical records for the specified data set and returns a pointer to the variable length index (IDX) to the FSA.

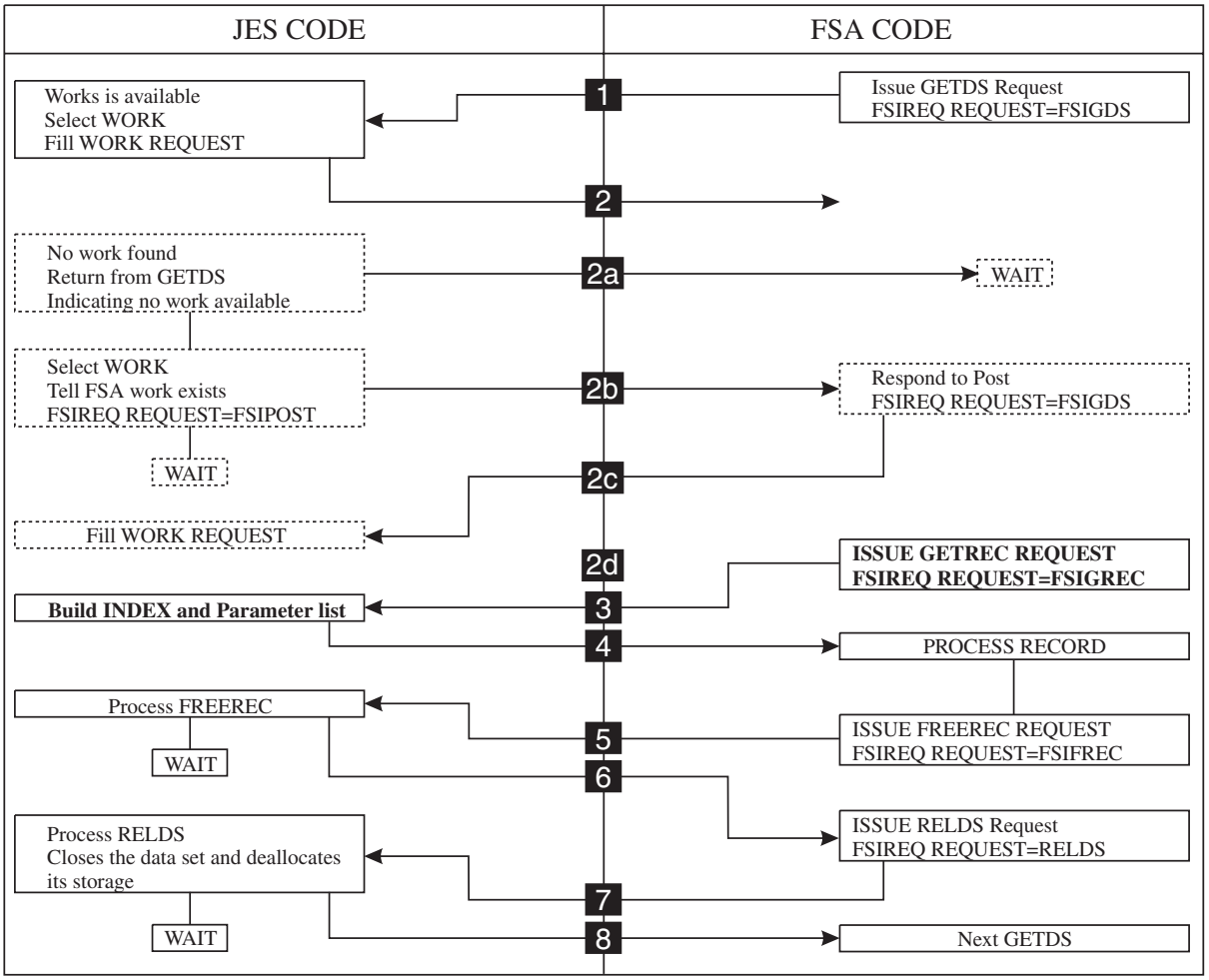

Figure 23. An Overview of Data Set Processing

The index (mapped by IAZIDX) contains one or more entries. Each entry normally represents one logical record. Entries may correspond to a partial record if it is a spanned record. Each entry contains a pointer to the data portion of the record. The FSA is responsible for accessing each of the individual record entries contained in the index. The number of entries in the table is provided in the IDXNUM field of the fixed index header (IAZIDX).

The storage associated with the logical records is assigned to the FSA and may not be reused by JES until the FSA issues a FREEREC request for the index representing those records.

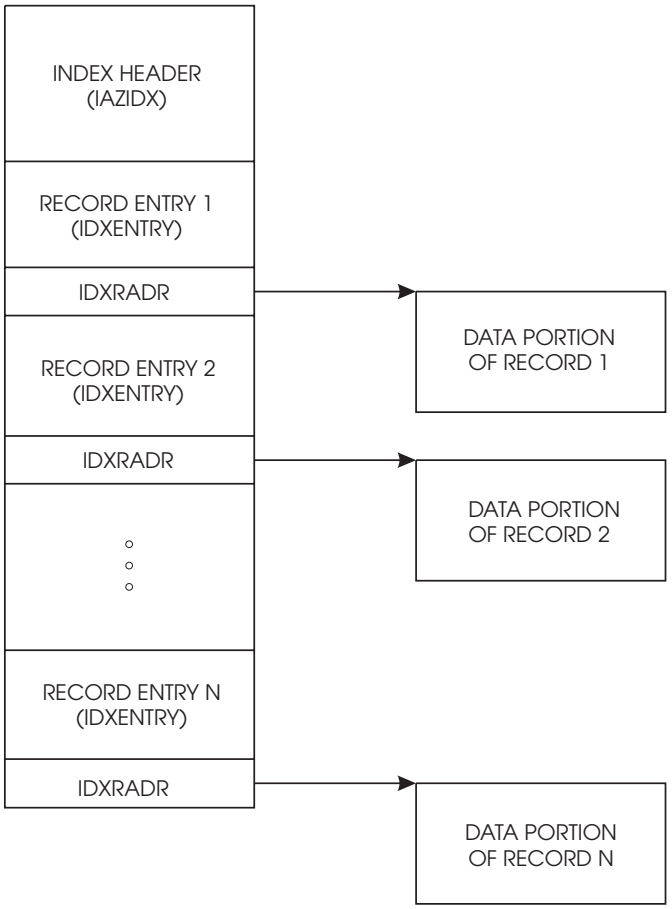

Figure 24. The Index (Mapped by IAZIDX) Returned From the GETREC Request

The FSI GETREC service supports both sequential and specific record retrieval. The FSA can specify the type of record retrieval desired in the GETREC parameter list. The FSA may request JES to begin record access at the beginning of the data set, at the next sequential record, or at a specific record (if the record id is known to the FSA). If this is the first GETREC request for a data set, JES automatically begins accessing records at the beginning of the data set, unless the FSA specifically indicates otherwise in the GETREC parameter list. Specific record retrieval is described in more detail below.

The FSI GETREC service supports multiple GETREC requests against a single data set without intervening FREEREC requests. This allows the FSA to perform "read-ahead" processing and therefore, obtain adequate despooling performance. The FSA, however, must be sensitive regarding storage limitations. When the FSA finishes processing the records pointed to by an index, it should issue a FREEREC

request for that index. If the FSA issues too many GETREC requests without issuing FREEREC requests, GETREC processing may eventually not be able to continue because of a buffer shortage. See “Releasing a SYSOUT Record” on page 77 for more information about the FSI FREEREC service.

FSI GETREC processing is asynchronous. JES does not require the FSA to finish getting all the records from one data set before it will accept another GETDS request by the FSA.

FSI GETREC Service Restriction: GETREC requests for a data set's records must be made from the same task that issued the GETDS request for that data set.

Specific Record Retrieval

The FSA may desire specific record retrieval for several reasons. Two examples are:

- After processing a SYNCH order, the FSA may need to reaccess data records from the resultant point of synchronization or repositioning. If the FSA desires to support repositioning, the FSA is responsible for collecting the required information.

  Note: The GETREC service also allows the FSA to re-access data records from the beginning of the data set.
- If the data set was previously interrupted, the FSA can restart the processing of the data set at the point indicated by the last data set checkpoint.

If the FSA desires specific record retrieval, it must indicate so in the GETREC parameter list and it must supply the identifier of the record at which JES is to begin record access. If this is not the first GETREC request for this data set, the FSA may use the record identifier that is incorporated into each index entry (IDXRECID) returned from a previous GETREC request. If this is the first GETREC request for this data set and valid checkpoint information exists, the FSA may use the record identifier that is included in the checkpoint information (the CHKRBA field of the IAZCHK record).

If the record has had trailing X'40' characters truncated by JES and the data set was created at z/OS Version 1 Release 4 JES2 (or higher) or z/OS Version 1 Release 5 JES3 (or higher) and was not sent from, stored and forwarded at, offloaded on, or dumped on any lower level JES2 or JES3, then the IDXORECL field contains the original record length before the truncation.

Initializing the GETREC Parameter List

The FSIREQ GETREC parameter list is used by both the FSA and JES to pass information. The FSA must initialize certain fields of the FSIREQ GETREC parameter list for each issuance of the GETREC request. The following table lists the required fields, the offsets and lengths of these fields, and the values that the FSA must assign. Detailed descriptions of the value assignments follow this table.

Note: The GLRECID field requires initialization only if the GETREC request is for specific record retrieval.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of GETREC parameter list
FSIFUNC	4	FSIGREC

Field Name	Length (bytes)	Value to be assigned
FSIFSID	4	The FSS/FSA IDs
GETREC Function Dependent Section		
GLRFLGR1	1	The type of record request
GLRECID	8	The record identifier
GLRDSID	12	The data set identifier

FSILEN
The length of the entire GETREC parameter list. The GETREC parameter list consists of both the IAZFSIP common header section and the GETREC function dependent section.

FSIFUNC
The GETREC function ID number. The FSA assigns the symbolic equate value FSIGREC to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

GLRFLGR1
The FSA uses this flag byte to specify the type of record request. The FSA may set one of the following indicators:

GLRREC1 B'10000000'
The FSA requests record access to begin at the first record in the data set.

GLRRECN B'01000000'
The FSA requests record access to begin at the next sequential record in the data set.

GLRRECS B'00100000'
The FSA requests record access to begin at the record specified in the GLRECID field.

GLRECID
The identifier of a specific record. This identifier is the JES equivalent of a relative block address (RBA). It was passed to the FSA either in the checkpoint area returned by a previous GETDS request or in an index returned from a previous GETREC request for this data set. The FSA needs to initialize this field only if it has set the GLRRECS indicator indicating JES is to begin record access at this record.

GLRDSID
The data set identifier.

Issuing the FSIREQ GETREC Request

When the FSA has completed initializing the GETREC parameter list, it issues the FSIREQ macro to invoke the FSI GETREC service. The format of this macro call is:

```
FSIREQ REQUEST=FSIGREC,TARGET=JES,PARM=GETREC
parm-list-addr,FSID=value-addr
```

See Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

JES GETREC Processing

The JES-supplied GETREC routine in the FSS address space receives control when the FSA issues the FSIREQ GETREC macro. The basic function of GETREC processing is to provide the FSA access to data records from a data set previously assigned to the FSA.

The GETREC service uses the data set identifier passed in the GETREC parameter list to locate the correct data records. It then determines the type of record retrieval requested (sequential or specific) and uses this information to begin assigning records to the FSA. If no errors occur during processing, the GETREC service fills in an index with pointers to the record(s) assigned and record status information. It then returns control to the FSA with a zero return code in register 15.

If an error occurs during GETREC processing, the GETREC service does the following:

- Indicates the error condition in the GETREC parameter list
- Indicates in the GETREC parameter list that no index was returned

If the error is the result of an invalid parameter list passed by the FSA, the GETREC service sets a **non-zero** return code in register 15 and then returns control to the FSA. The FSA should correct the error in the GETREC parameter list and then reissue the GETREC request. For other types of errors, the GETREC service sets a **zero** return code in register 15, indicating that processing can continue. See the specific error indicators in the GETREC parameter list for more information.

Information Returned in GETREC Parameter List

On return from successful GETREC processing, the GETREC parameter list contains the information listed below. If GETREC processing was not successful, the GLRINDX field in the GETREC parameter list does not contain a pointer to an index.

The following figure shows the connection between the different sections of the FSIREQ parameter list for GETREC processing.

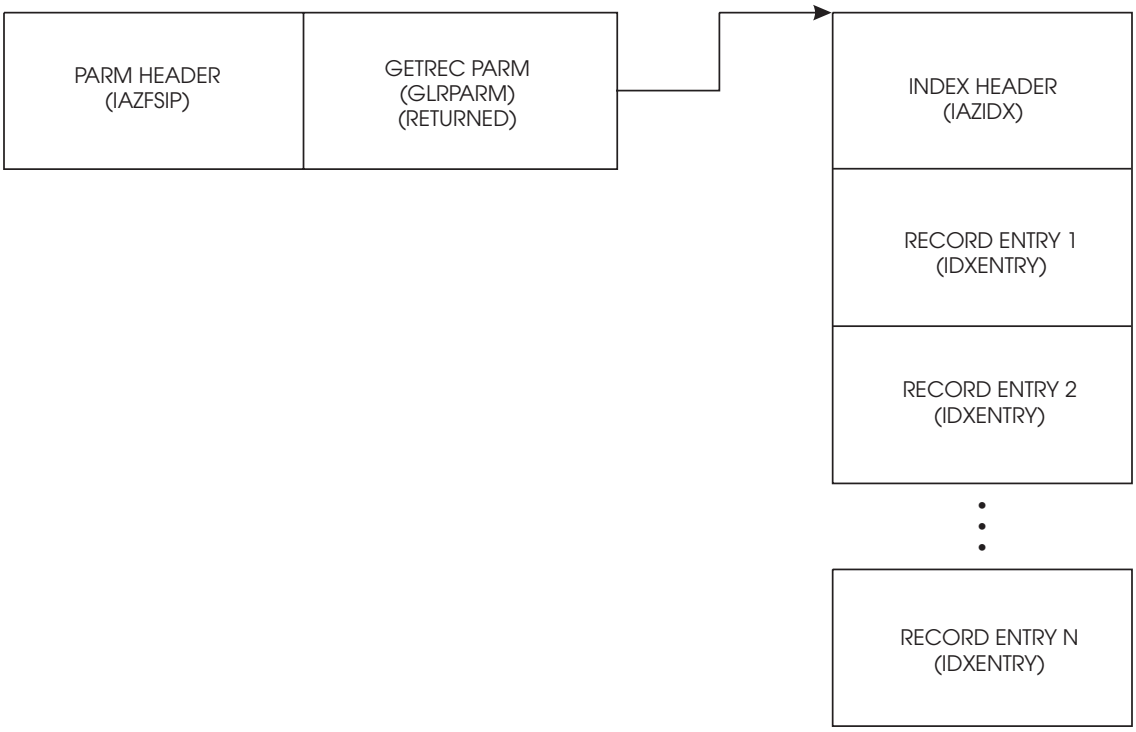

Figure 25. FSIREQ Parameter Lists for GETREC Processing

Field Name	Length (bytes)	Value assigned
Common Parameter Header Section		
FSILEN *	4	Length of GETREC parameter list
FSIFUNC *	4	FSIGREC
FSIFSID *	4	The FSS/FSA IDs
GETREC Function Dependent Section		
GLRFLGR1 *	1	The type of record request
GLRFLGS1	1	GETREC processing status information
GLRINDX	4	A pointer to the index returned
GLRECID *	8	The spool record ID
GLRDSID *	12	The data set identifier

The fields with an asterisk (*) contain values set by the FSA when it issued the GETREC request. The GLRECID field may or may not be set depending on whether the request was for specific record retrieval. The fields that JES set during GETREC processing are described in detail below:

GLRFLGS1
This flag byte contains GETREC processing status information. The following indicators may be set:

GLREOF B'10000000'
JES has reached the end of file (EOF) for the data set. If JES reaches the end of file without encountering any additional records for the GETREC request, JES does not return an index, and sets the GLRNOI indicator.

GLRNBA B'01000000'
No buffers are available to satisfy the GETREC request. This condition may occur when the FSA makes several GETREC requests without subsequent FREEREC requests. The FSA can recover from this error by issuing FREEREC requests to release the storage resources. The FSA can then retry the GETREC request.

GLRIPL B'00100000'
The parameter list passed by the FSA was invalid. Possible reasons for this error are: 1) the FSA specified an invalid type of record request (GLRFLGR1), 2) the FSA specified an invalid record ID (GLRECID) and specified specific record retrieval, 3) the FSA specified an invalid data set identifier (GLRDSID).

GLRIOE B'00010000'
The GETREC service detected a permanent hardware I/O error on the JES spool device during processing of the current data set.

The FSA should not attempt further processing of the data set. It should issue a RELDS request for the data set indicating in the RELDS parameter list that data set processing is complete. The FSA can then continue processing the next data set.

GLRLGE B'00001000'
Either a logic error (for example, an incorrect spool record format) or an ABEND has occurred during processing of the current data set. This is probably a JES error for which the JES has already provided the diagnostic data (for example, trace and a dump).

The FSA should not attempt further processing of the data set. It should issue a RELDS request for the data set indicating in the RELDS parameter list that data set processing is complete. The FSA can then continue processing the next data set.

GLRNOI B'00000100'
JES did not return an index. This indicator is always set when one of the previous indicators (except the GLREOF indicator) is set. If the GLREOF indicator is set, this indicator may or may not be set.

GLRINDX
This field contains a pointer to the index returned by GETREC processing. If an index was not returned, this field will be zero.

Information Contained in Index

The GLRINDX field points to the index returned from GETREC processing. The index contains a header section and an index entry area. The index entry area is of variable length depending on how many records were assigned to the FSA. The fields of the index are described below. Only one index entry is shown.

Field Name	Length (bytes)	Value assigned
Fixed Header of Index Table		
IDXID	4	'IDX ' (ID of Index table)
IDXNUM	2	Number of entries in the table. Each entry refers to a specific logical record.
IDXTOK	2	A JES-supplied token that JES uses for validation purposes
RESERVED	4	
Index Entry Area		

Field Name	Length (bytes)	Value assigned
IDXENTRL	2	Length of the index entry
IDXRECL	2	Length of the data portion of the logical record
IDXFLAG1	1	Status information for the record
IDXRADR	4	Address of the data portion of the logical record
IDXRECID	8	The identifier of this logical record

IDXFLAG1
This flag byte contains status information for the logical record identified by IDXRECID. The following indicators may be set in this flag byte:

IDXDSR B'10000000'
The record contains stream mode data.

IDXLMR B'01000000'
The record contains line mode data.

IDXANSI B'00100000'
The record contains ANSI control characters.

IDXMACH B'00010000'
The record contains machine control characters.

IDXSRS B'00001000'
This entry is actually the start of split record.

IDXSRM B'00000100'
This entry is the middle of a split record.

IDXSRE B'00000010'
This entry is the end of a split record.

IDXOPJ B'00000001'
The OPTCODE=J was used for the record.

IDXRECID
The identifier of this logical record. The FSA may use this identifier to request JES to begin access to a data set at this logical record by specifying this value on a GETREC request and specifying this value as the GLRECID.

Releasing a SYSOUT Record

An FSA invokes the FSI FREEREC service to release logical records previously obtained with a GETREC request. The FSA provides a pointer to an index and JES releases the storage associated with the record index entries. Releasing logical records allows JES to reuse the associated storage.

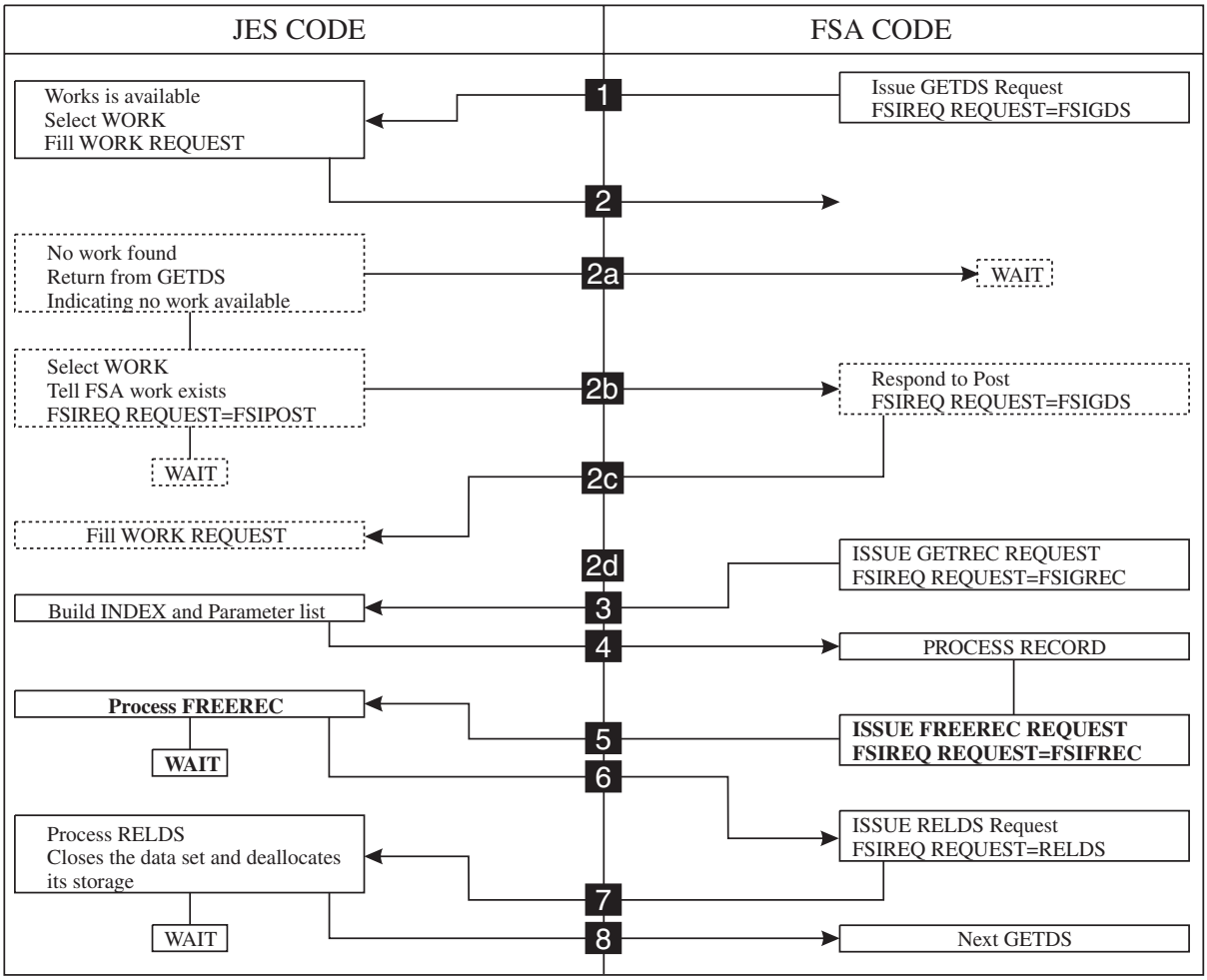

Figure 26. An Overview of Data Set Processing

FSI FREEREC processing is asynchronous. JES does not require the FSA to finish releasing all the records from one data set before it will accept another GETDS request by the FSA.

FSI FREEREC Service Restriction: FREEREC requests for a data set's records must be made from the same task that issued the GETDS request for that data set.

Initializing the FREEREC Parameter List

For each FREEREC request, the FSA must initialize certain fields of the FSIREQ FREEREC parameter list.

The following figure shows the connection between the different sections of the FSIREQ parameter list for FREEREC processing.

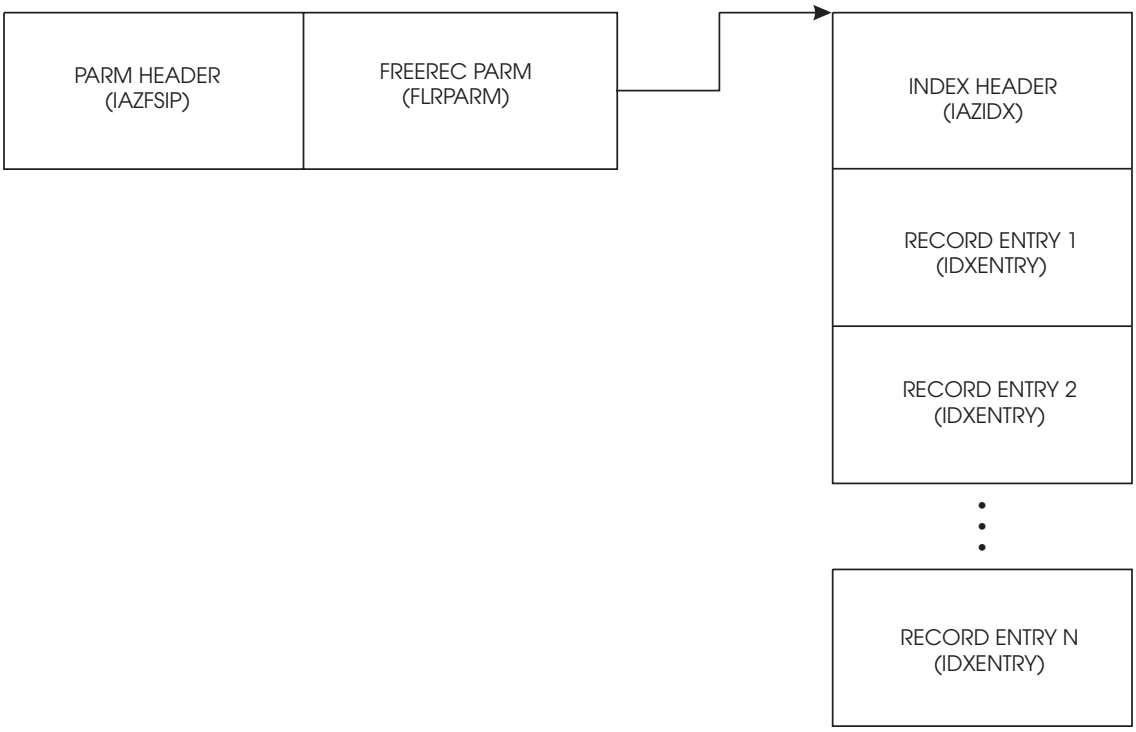

Figure 27. FSIREQ Parameter Lists for FREEREC Processing

The following table lists the required fields, the offsets and lengths of these fields, and the values that the FSA must assign. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of FREEREC parameter list
FSIFUNC	4	FSIFREC
FSIFSID	4	The FSS/FSA IDs
FREEREC Function Dependent Section		
FLRINDX	4	The pointer to the index to be freed.
FLRDSID	12	The data set identifier

FSILEN
The length of the entire FREEREC parameter list. The FREEREC parameter list consists of both the IAZFSIP common header section and the FREEREC function dependent section.

FSIFUNC
The FREEREC function ID number. The FSA assigns the symbolic equate value FSIFREC to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

FLRINDX
The pointer to the index to be freed. JES returned this pointer on a previous GETREC request in the GLRINDX field of the GETREC parameter list.

FLRDSID
The identifier of the data set to which the record(s) belong. This identifier was returned from GETDS processing in the GDSDSID field in the GETDS parameter list.

Issuing the FSIREQ FREEREC Request

When the FSA has completed initializing the FREEREC parameter list, it issues the FSIREQ macro to invoke the FSI FREEREC service. The format of this macro call is:

```
FSIREQ REQUEST=FSIFREC,TARGET=JES,PARM=FREEREC
parm-list-addr,FSID=value-addr
```

See Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

JES FREEREC Processing

The JES-supplied FREEREC routine receives control when the FSA issues the FSIREQ FREEREC macro. The FREEREC service uses the data set identifier and the index pointer passed in the FREEREC parameter list to de-allocate the storage areas associated with the data set's records referenced by the index. The storage is then available for subsequent GETREC processing.

Status of Request Returned by JES

If no errors occur during FREEREC processing, JES returns control to the FSA with a zero return code in register 15. If an error does occur that prevents FREEREC processing from continuing, JES indicates this to the FSA by passing a non-zero return code in register 15. The error can be one of the following:

- An invalid parameter list
- The IDXTOK is invalid
- The IDX has already been freed
- The data set has already been released (RELDS)

The FSA should correct the problem and reissue the request.

Releasing a SYSOUT Data Set

The FSA invokes the FSI RELDS service to:

- Return a data set that was previously obtained with a GETDS request to JES
- Notify JES of the data set's processing status.

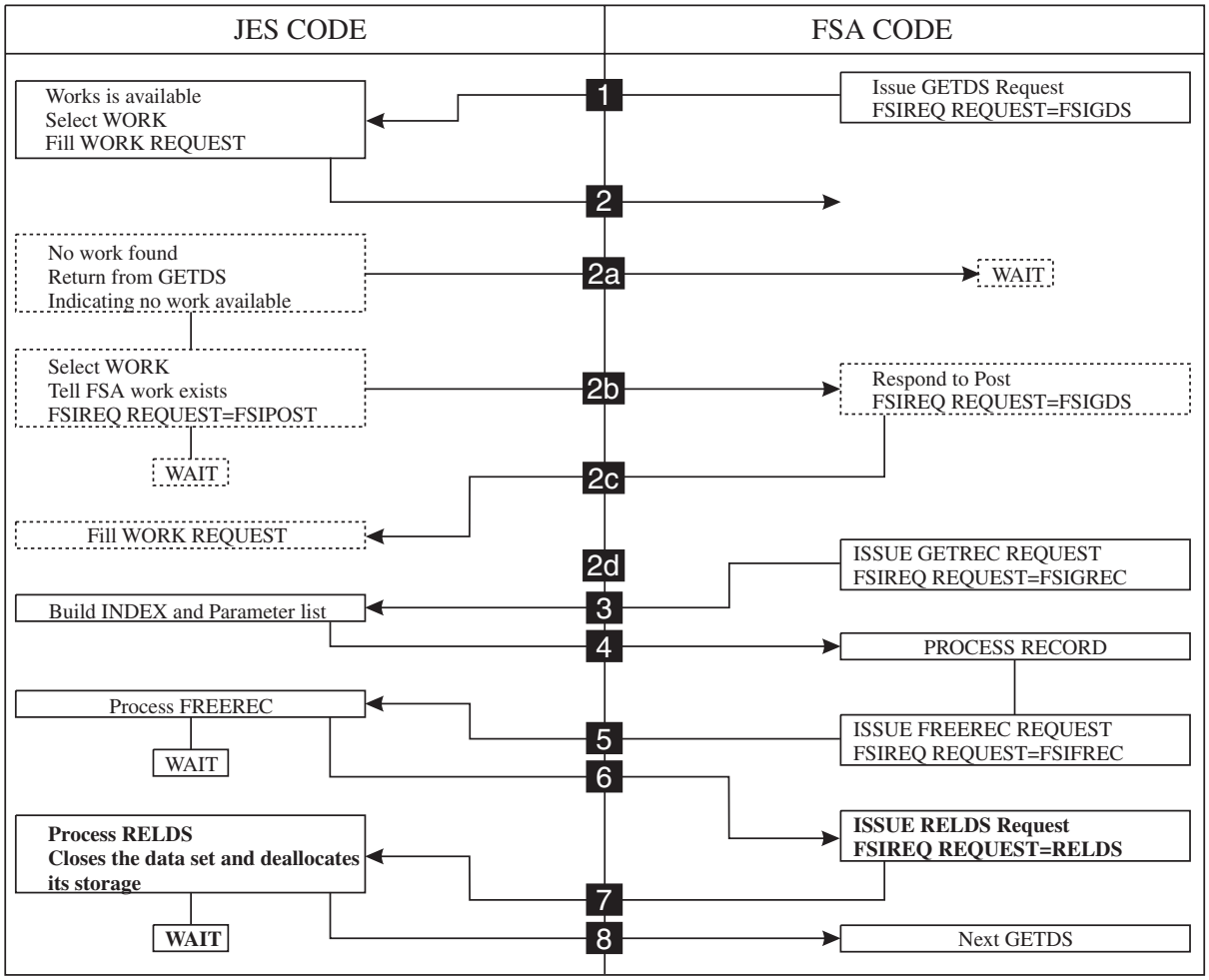

Figure 28. An Overview of Data Set Processing

The FSI RELDS service is functionally equivalent to closing and de-allocating the data set. The storage associated with the data set is made available to JES for reuse. If the FSA indicates that valid checkpoint information exists for the data set, JES writes the final checkpoint record to spool. If the FSA issues a RELDS request for a data set before it releases all of its records (using the FREEREC request), the FSI RELDS service also frees the storage for all outstanding records for that data set.

Data Set Processing Status

In the RELDS parameter list, the FSA indicates the data set's processing status, as follows:

- The data set has been completely processed.
- The data set has not been completely processed. Its checkpoint information is:
  - valid
  - invalid
- The data set is unprintable.

The descriptions of the specific indicators that may be set in the status flag byte (RDSFLGS1) explain how JES reacts to each processing status.

Initializing the RELDS Parameter List

The FSA must initialize specific fields of the FSIREQ RELDS parameter list for each issuance of the RELDS request.

The following figure shows the connection between the different sections of the FSIREQ parameter list for RELDS processing.

Figure 29. FSIREQ Parameter Lists for RELDS Processing

The following table lists the required fields, the offsets and lengths of these fields, and the values that the FSA must assign. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of RELDS parameter list
FSIFUNC	4	FSIRDS
FSIFSID	4	The FSS/FSA IDs
RELDS Function Dependent Section		
RDSFLGS1	1	The processing status of the data set to be released
RDSDSID	12	The data set identifier
RDSMIDSE	8	Message ID indicating data set error

FSILEN
The length of the entire RELDS parameter list. The RELDS parameter list consists of both the IAZFSIP common header section and the RELDS function dependent section.

FSIFUNC
The RELDS function ID number. The FSA assigns the symbolic equate value FSIRDS to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

RDSFLGS1
This flag byte indicates the processing status of the data set to be released. The FSA may set the following indicators:

RDSDONE B'10000000'
The FSA has completely processed the data set. This indicator indicates that the entire data set has passed the data integrity point (DIP) of the device and JES may purge the data set from spool.

RDSINC B'01000000'
The FSA has not completely processed the data set. Processing was interrupted because either the FSA processed a SYNCH order that specified

an interrupt action or an error occurred on the device. This indicator causes JES to re-queue the data set for processing.

RDSCKPI B'00100000'
The checkpoint information for the data set is invalid and should not be used when the data set is again selected for processing. The data set should be printed from the beginning. JES ignores this indicator if the FSA also sets the RDSDONE or RDSUNPR indicator.

RDSUNPR B'00010000'
The data set is unprintable. During processing, the FSA detected an error in the data set that prevents it from completely printing the data set. This indicator causes JES to re-queue the data set for processing, but mark it as held. This prevents the data set from being selected until the error is corrected and the data set is released from hold.

RDSDSID
The identifier of the data set that is to be released. This identifier was previously returned by JES during GETDS processing.

RDSMIDSE
If FSA has encountered an error when processing the data set, this field contains a message ID that describes the error. See the FSA message manual for details.

Issuing the FSIREQ RELDS Request

When the FSA has completed initializing the RELDS parameter list, it issues the FSIREQ macro to invoke the FSI RELDS service. The format of this macro call is:

```
FSIREQ REQUEST=FSIRDS,TARGET=JES,PARM=RELDS
parm-list-addr,FSID=value-addr
```

See Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

JES RELDS Processing

The JES-supplied RELDS routine receives control when the FSA issues the FSIREQ RELDS request. This routine closes the data set and de-allocates the storage resources associated with it. The RELDS routine uses the data set processing status passed by the FSA to determine what additional actions are required by JES. When the data set is processed, JES invokes the Scheduler JCL Facility (SJF) to release the SWBs associated with the data set passed from the GETDS request. This data set is no longer available for use by the FSA. If the data set was incompletely processed, JES updates the checkpoint data according to the completion status provided by the FSA and writes the final checkpoint record to spool.

Status of Request Returned by JES

If no errors occur during RELDS processing, JES returns control to the FSA with a zero return code in register 15. If an error does occur that prevents RELDS processing from continuing, JES passes a non-zero return code in register 15 to the FSA.

SMF Record Writing

After the FSA issues a RELDS request for a data set, it is expected to write an SMF type 6 record for that data set. The JSPA provided by JES at GETDS processing contains SMF record information. The FSA uses this information and its own information to generate the SMF record for the assigned data set. See "Information

Contained in the JSPA" on page 58 for more information about the JSPA. Refer to *z/OS MVS System Management Facilities (SMF)* for more information about type 6 SMF records.

Requesting a Checkpoint of Processing

The FSA invokes the FSI CHKPT service to request JES to record checkpoint data for a spool data set currently being processed by the FSA. The FSA passes the address of a checkpoint record containing data set information to JES and JES writes the checkpoint record to spool. The FSA is responsible for ensuring output checkpoints are taken at appropriate points in processing. It needs to be able to handle checkpoint intervals specified on a data set basis using SWB information. If a checkpoint interval was not specified in the data set's SWBs, the FSA uses the default passed by JES in the START FSA order parameter list.

Checkpointing is not a mandatory function that must be provided by the FSA. If your FSA will only process small data sets (1 or 2 pages), the FSA can decide not to support checkpointing.

Note: Even if your FSA does not support checkpointing, it is still responsible for providing the checkpoint data area and checkpoint area length in the GETDS parameter list.

Purpose of the FSI CHKPT Service

The FSI CHKPT service supports data set checkpointing for restart. The checkpoint information recorded during FSI CHKPT processing may later be used by the FSA to restart the printing of a previously interrupted data set from the point indicated by the data set's last checkpoint. For example, if the processing of a data set is interrupted due to a SYNCH order, the FSA returns the data set to JES (using the RELDS request) with an incomplete processing status. If the FSA also indicates in the RELDS parameter list that valid checkpoint information exists for the data set, JES saves the information for future processing. If on a future GETDS request, this same data set is again assigned to an FSA, JES will fill in the checkpoint area provided by the FSA. JES will also indicate in the GETDS parameter list that valid checkpoint information exists for the data set. The FSA then uses this information to restart the printing of the data set from the point indicated by the last checkpoint.

Preparing for Checkpointing

When an FSA determines an output checkpoint needs to be taken for a data set, it must:

1. Establish and initialize a checkpoint area. This checkpoint area must begin with the FSI checkpoint record (IAZCHK). If the FSA wants to provide additional device dependent checkpoint information, that information immediately follows IAZCHK.
2. Initialize the FSIREQ CHKPT parameter list.
3. Issue the FSIREQ CHKPT request to invoke the FSI CHKPT service.

Initializing the FSI Checkpoint Record

The following table lists the fields contained in the IAZCHK checkpoint record. The CHKID field is the only field that JES requires the FSA to initialize. The FSA may initialize the remaining fields on a discretionary basis.

Note: JES uses the CHKJESWK field. The FSA does not initialize this area. It is shown in the table only to provide a complete record format.

Field Name	Length (bytes)	Assigned Value
CHKID	4	'CHK' (FSI Checkpoint record identifier)
CHKLNGTH	2	Length of FSI checkpoint record
CHKJESWK	64	JES dependent checkpoint information for the data set. The FSA does not use this area.
CHKRBA	8	The identifier of the record currently being processed
CHKDEV	4	The device type
CHKMOD	4	The model number of the device
CHKCOPY	4	The number of copies that have been printed
CHKTRNC	4	The transmission count
CHKREC	4	The logical record count
CHKPAGE	4	The physical page count
CHKPROD	8	The product that created the checkpoint record
CHKVER	4	The version of the product
CHKRELS	4	The release of the product
CHKMODF	4	The modification level of the product
CHKSERV	4	The service level of the product

Initializing the CHKPT Parameter List

The FSA must initialize certain fields of the FSIREQ CHKPT parameter list for each CHKPT request.

The following figure shows the connection between the different sections of the FSIREQ parameter list for checkpoint processing.

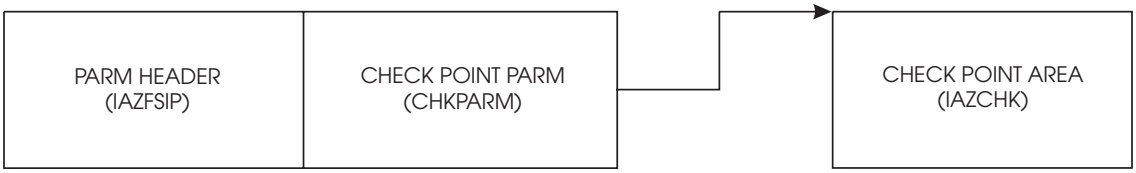

Figure 30. FSIREQ Parameter Lists for CHKPT Processing

The following table lists the required fields, the offsets and lengths of these fields, and the values that the FSA must assign. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of CHKPT parameter list
FSIFUNC	4	FSICKPT
FSIFSID	4	The FSS/FSA IDs
CHKPT Function Dependent Section		

Field Name	Length (bytes)	Value to be assigned
CHKADR	4	The pointer to the FSA-supplied checkpoint area
CHKFLGR1	1	CHKFCWRT
CHKDSID	12	The data set identifier

FSILEN
The length of the entire CHKPT parameter list. The CHKPT parameter list consists of both the IAZFSIP common header section and the CHKPT function dependent section.

FSIFUNC
The CHKPT function ID number. The FSA assigns the symbolic equate value FSICKPT to this field.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

CHKADR
The pointer to the FSA checkpoint area which contains the initialized FSI checkpoint record (IAZCHK) and optionally, any device dependent information to be checkpointed. On return from CHKPT processing, the FSA may reuse this area.

CHKFLGR1
This flag byte indicates the type of checkpoint request. The FSA may set the following indicator:

CHKFCWRT B'10000000'
The FSA requires a forced write of the checkpoint record. During checkpoint processing, if I/O is not yet complete for the checkpoint buffer and CHKFCWRT is set, JES waits for I/O completion and then writes the record to spool before returning to the FSA. If CHKFCWRT is not set, JES returns to the FSA without waiting for I/O completion.

For optimum performance, the FSA should set this indicator only for a checkpoint request made immediately prior to releasing the data set with a RELDS request.

CHKDSID
The identifier of the data set that is being checkpointed. This identifier was returned to the FSA on a previous GETDS request.

Issuing the FSIREQ CHKPT Request

When the FSA has completed initializing the CHKPT parameter list, it issues the FSIREQ macro to invoke the FSI CHKPT service. The format of this macro call is:

```
FSIREQ REQUEST=FSICKPT,TARGET=JES,PARM=CHKPT parm-list-addr,
FSID=value-addr
```

See Chapter 4, “The FSIREQ Macro,” on page 13 for a complete description of each operand on this macro and the defaults that may be taken.

JES CHKPT Processing

The JES-supplied CHKPT routine receives control when the FSA issues the FSIREQ CHKPT request. The basic function of the CHKPT service is to write the

checkpoint records to the JES spool data set. JES writes the record directly from the FSS address space. The CHKPT service does not require JES address space functions.

Before writing the checkpoint record to spool, JES copies it to its own buffer. If I/O is not yet complete from a previous checkpoint write and a forced write was not specified, JES sets a zero return code in register 15 and returns control to the FSA without the checkpointing of the record completed. If a previous checkpoint I/O is not outstanding and forced write was not specified then the checkpoint write is initiated but control will return to the FSA before the write completes.

If a forced write was specified and a previous checkpoint I/O is outstanding, JES will wait for the outstanding I/O to complete, issue a write for the current checkpoint and wait for that I/O to complete before returning control to the FSA. If a previous I/O is not outstanding, JES initiates a write for the current checkpoint and waits for it to complete before returning control to the FSA.

If, during processing, JES detects an error other than a bad checkpoint record (for example, invalid parameter list length), it sets a non-zero return code in register 15 and returns control to the FSA.

If JES detects a checkpoint write I/O error, it sets the CHKFCERR flag bit on in the CHKPT parameter list indicating a permanent I/O error and then returns control to the FSA with a non-zero return code in register 15.

Bad Checkpoint Record Detected by JES

If JES determines that the checkpoint record is bad, it initializes the CHKFLGS1 flag byte in the CHKPT parameter list before returning to the FSA.

CHKFLGS1	1	CHKFCERR

CHKFCERR B'10000000'
A permanent I/O or processing error occurred while JES was attempting a write of the checkpoint record. JES ignores the checkpoint request and stops checkpointing the current data set. The FSA should retry the request (resume checkpointing) for the next data set.

Chapter 10. Responding to Device Orders From JES

When JES determines that an operator command requires participation of an FSA, JES converts the command into an FSI order. JES then issues an FSIREQ ORDER request to the FSA's FSI ORDER routine. The FSA supplied the address of this routine to JES at FSA CONNECT time in the CDFAD field of the CONNECT parameter list.

When the FSI ORDER routine receives the order, it is responsible for determining the type of order issued and then either posting the appropriate FSA task to process the order or processing the order directly. When order processing is complete, the FSA responds to JES with the required data. JES will not send another order to the FSA until it receives a response for the outstanding order.

This chapter describes the processing for orders that:

- request a change in device or data set characteristics
- affect the flow of data through the device
- request information about a data set currently being processed by an FSA device.

Notes:

1. Refer to Chapter 5, "FSI Communication," on page 17 for restrictions on responding to orders.
2. This chapter explains the tasks involved in processing orders, it does not explain how the FSI order routine should be coded to satisfy those tasks.

The Query Order

JES issues a query order to an FSA's FSI ORDER routine when an operator command requests information concerning the data set at the operator observation point (OOP). Because this order pertains to the data set at the OOP, JES requires an immediate response. The query order is unique in that respect.

Note: For the 3800-3, the OOP is the point at which the output can be seen by the operator.

The following topics describe the commands resulting in a query order and the FSA processing required for this order.

Examples of JES Commands Resulting in a Query Order

Both JES2 and JES3 issue the query order for various commands. Examples of these commands are:

- JES2
  - $N PRTnnnn - repeat device.
  - $DU,PRTnnnn - display device status.
- JES3
  - *START,devname,P - display pending pages and records for current data set.

Processing the Query Order

When JES issues an FSI query order, it passes the address of the query order parameter list in register 1 to the FSA's FSI order routine. The query order parameter list consists of the following sections:

- Common parameter header
- Common order header (which contains a pointer to the JES-provided order response area (IAZRESPA))

Note: There is no variable order data section for the query order.

The following figure shows the connection between the different sections of the FSIREQ parameter list for the QUERY order.

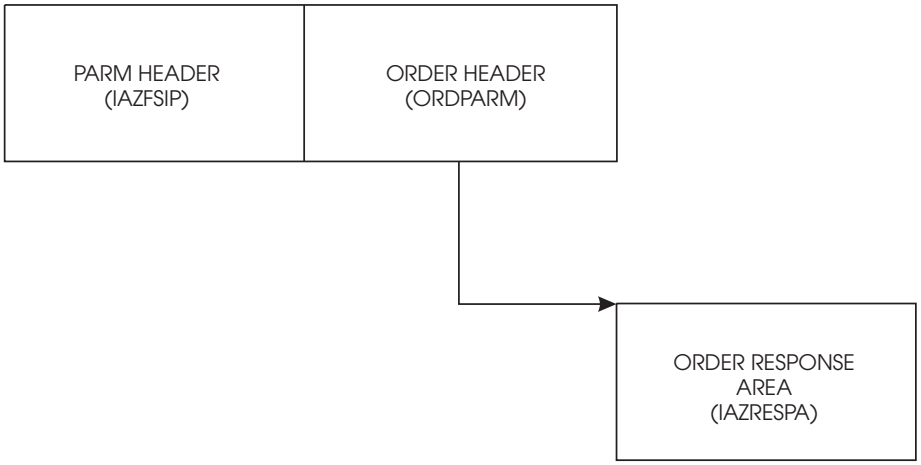

Figure 31. FSIREQ Parameter Lists for the QUERY Order

The table below lists the initialized fields, the lengths of these fields, and the values that JES has assigned. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value assigned
Common Parameter Header Section		
FSILEN	4	Length of query order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS/FSA IDs
Common Order Header Section		
ORDFDATA	4	A value supplied to JES by the FSA as a CONNECT parameter (CDFFDATA)
ORDRSPAD	4	Address of the order response area (IAZRESPA)
ORDID	2	ORDQUERY

FSILEN
The length of the entire query order parameter list. The query order parameter list consists of both the IAZFSIP common parameter header section and the common order header section.

FSIFUNC
The ORDER function ID (FSIORDER).

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSA-related information. The FSA passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value to the FSI's ORDER routine so that it can start the appropriate FSA.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The query order ID number. ORDQUERY is the symbolic equate.

The FSA's FSI ORDER routine uses the ORDID value to determine that the JES order requests a query action. The FSI ORDER routine is then responsible for obtaining information about the data set currently at the OOP and immediately returning that information to JES in the JES provided order response area (IAZRESPA). If the FSI ORDER routine determines that no data set is currently active at the OOP, it indicates this condition to JES in the order response area. Chapter 5, “FSI Communication,” on page 17 explains the IAZRESPA fields that the FSA needs to initialize.

The query order information can be kept in a control block whose address JES passes to the FSI order routine in the ORDFDATA field.

Note: Because the query order requires an immediate response, it is recommended that the FSI ORDER routine process the order directly rather than posting an FSA task to process the order.

The Set Order

JES issues a set order to an FSA's FSI ORDER routine to set or change device characteristics unrelated to data set processing specifications. JES specifically issues the set order to set or change the non-process runout (NPRO) timer interval. The non-process runout (NPRO) time interval is that time interval during which output remains in the paper path but has not reached the stacker. After the NPRO time interval has elapsed, the FSA directs the device to force the output to the stacker. The new NPRO values goes into effect the next time the device goes idle.

The following topics describe the JES commands that result in a set order and the FSA processing required for this order.

Examples of JES Commands Resulting in a Set Order

Examples of JES commands resulting in a set order are:

- JES2
  - $T PRTnnnn,NPRO=nnnn
- JES3
  - *S,devname,NPRO=nnnn
  - *R,devname,NPRO=nnnn

Processing the Set Order

When JES issues an FSI set order, it passes the address of the set order parameter list in register 1 to the FSA's FSI order routine. The set order parameter list consists of the following sections:

- Common parameter header
- Common order header (which contains a pointer to the JES provided order response area (IAZRESPA))
- set order dependent section.

The following figure shows the connection between the different sections of the FSIREQ parameter list for SET order processing.

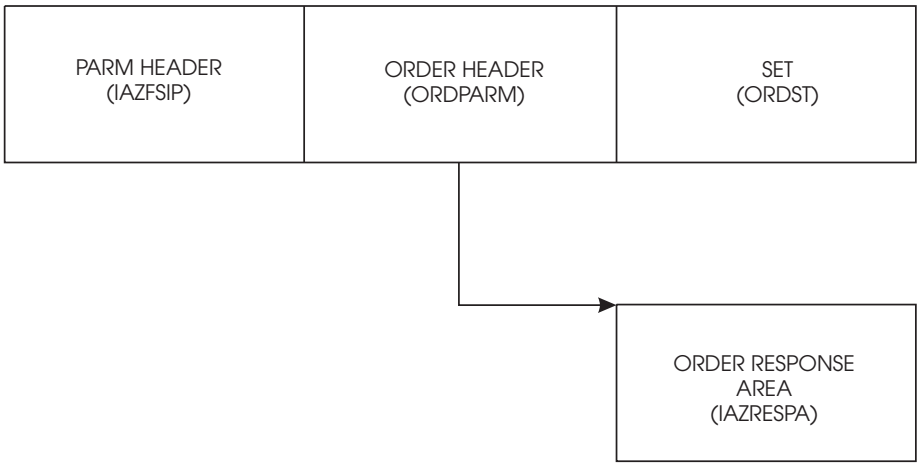

Figure 32. FSIREQ Parameter Lists for SET Order Processing

The table below lists the initialized fields, the lengths of these fields, and the values that JES has assigned. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of set order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS/FSA IDs
Common Order Header Section		
ORDFDATA	4	A value supplied to JES by the FSS/FSA as a CONNECT parameter (CDFFDATA)
ORDRSPAD	4	Address of the order response area (IAZRESPA)
ORDID	2	ORDSET
Set Order dependent Section		
ORDSTR1	1	Type of set order
ORDSTNI	4	The NPRO interval value (in seconds)

FSILEN

The length of the entire set order parameter list. The set order parameter list

consists of the IAZFSIP common parameter header section, the common order header section, and the set order dependent section.

FSIFUNC
The ORDER function ID (FSIORDER).

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSA-related information. The FSA passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value to the FSA ORDER routine so that it can start the appropriate FSA.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The set order ID number. ORDSET is the symbolic equate.

ORDSTR1
This flag byte indicates the set action to be performed by the FSA. JES sets one of the following indicators:

ORDSTSN B'10000000'
The FSA is to set the NPRO timer interval. The ORDSTNI field contains the NPRO interval value.

ORDSTDN B'01000000'
The FSA is to disable the NPRO timer interval.

ORDSTNI
The NPRO interval value, in seconds. If the set order requests the FSA to disable the NPRO timer interval (ORDSTDN is set), this field is set to zero.

The FSA's FSI ORDER routine uses the ORDID value to determine that the JES order requests a set action. The FSI ORDER routine then either processes the set order directly or posts an FSA task to process the order. If a response to the order cannot be immediately provided to JES, the FSI ORDER routine sets the ORDFLGS1 field in the common order header section equal to ORDARESP. This notifies JES that the response to the order will be returned at a later time by means of an FSIREQ SEND request.

When set order processing is complete, the FSA responds to JES indicating whether the order was processed successfully. Chapter 5, “FSI Communication,” on page 17 explains how the FSA responds to JES.

The Synch Order

JES issues a synch order to an FSA's FSI ORDER routine when an action needs to be performed against the data set currently at the operator observation point (OOP). The synch order requests that FSA processing be synchronized to the point of actual printing.

The following topics describe the JES commands that result in a synch order and the FSA processing required for this order.

Examples of JES Commands Resulting in a Synch Order

Examples of JES commands resulting in a synch order are:

- JES2
  - $B PRTnnnn,m - backward space
  - $F PRTnnnn,m - forward space
  - $Z PRTnnnn - halt device.
- JES3
  - *RESTART,devname - restart data set at beginning
  - *CANCEL,devname - terminate data set
  - *START,devname,CP=+2, - increment copy count by 2 for current data set.

Processing the synch order

When JES issues an FSI synch order, it passes the address of the synch order parameter list in register 1 to the FSA's FSI order routine. The synch order parameter list consists of the following sections:

- Common parameter header
- Common order header
- Synch order dependent section.

The following figure shows the connection between the different sections of the FSIREQ parameter list for SYNCH order processing. From left to right, the PARM HEADER (IAZFSIP), ORDER HEADER (ORDPARM) and SYNCH (ORDSY) sections are directly linked, with the ORDER HEADER (ORDPARM) section connecting separately to the ORDER RESPONSE AREA (IAZRESPA) section.

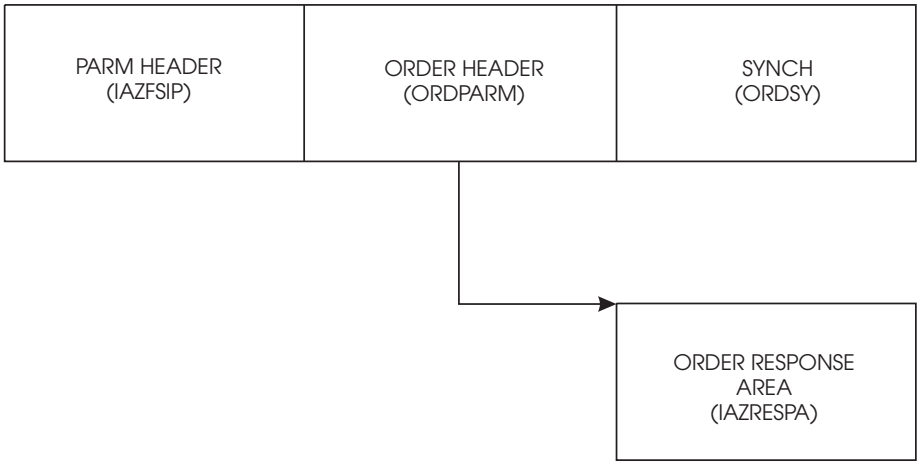

Figure 33. FSIREQ Parameter Lists for synch order processing

The table below lists the initialized fields, the offsets and lengths of these fields, and the values that JES has assigned. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of synch order parameter list
FSIFUNC	4	FSIORDER

Field Name	Length (bytes)	Value to be assigned
FSIFSID	4	The FSS/FSA IDs
Common Order Header Section		
ORDFDATA	4	A value supplied to JES by the FSA as a CONNECT parameter (CDFFDATA)
ORDRSPAD	4	Address of the order response area (IAZRESPA)
ORDID	2	ORDSYNC
Synch Order Dependent Section		
ORDSYR1	1	Synch action to be performed
ORDSYR2	1	Reposition action to be performed
ORDSYR3	1	Device update action to be performed
ORDSYR4	1	Data set update action to be performed
ORDSYR5	1	Interrupt action to be performed
ORDSYR6	1	Miscellaneous action to be performed
ORDSYNP	4	Number of pages to reposition
ORDSYKI	4	Checkpoint interval (seconds or pages)
ORDSYCP	2	Copy count value
ORDSYMSG	120	Message text for users output

FSILEN
The length of the entire synch order parameter list. The synch order parameter list consists of the IAZFSIP common parameter header section, the common order header section, and the synch order dependent section.

Note: If a pointer to the set order parameter list is provided in the synch order dependent section, the length of the set order parameter list is not included in the FSILEN value.

FSIFUNC
The ORDER function ID number. FSIORDER is the symbolic equate.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSA related information. The FSA passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value to the FSA ORDER routine so that it can start the appropriate FSA.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The synch order ID number. ORDSYNC is the symbolic equate.

ORDSYR1
This flag byte indicates the type of synch action that the FSA must perform. If it equals zero and the device is a buffered device, the FSA is to release all the data sets in its buffer. If it equals zero and the device is not a buffered device, the FSA is to continue processing. One of the following indicators may be set:

ORDSYBCP B'10000000'
The FSA is to synchronize the data set to the previous checkpoint.

ORDSYFCP B'01000000'
The FSA is to synchronize the data set to the next checkpoint.

ORDSYBTM B'00100000'
The FSA is to synchronize the data set to the beginning of the current transmission.

ORDSYETM B'00010000'
The FSA is to synchronize the data set to the end of the current transmission.

ORDSYBDS B'00001000'
The FSA is to synchronize the data set to the beginning of the data set.

ORDSYEDS B'00000100'
The FSA is to synchronize the data set to the end of the data set.

ORDSYR2
This flag byte indicates the type of reposition action that the FSA must perform. One of the following indicators may be set:

ORDSYRI B'10000000'
The FSA is to increment the page position.

ORDSYRD B'01000000'
The FSA is to decrement the page position.

ORDSYNR B'00100000'
The FSA is not to reposition past the end of the data set at the OOP.

If the reposition request causes the FSA to go beyond the end of the data set, it must:

1. Stop the reposition.
2. Respond to the SYNCH order with the RESP2EOD bit on in the order response area. Chapter 5, “FSI Communication,” on page 17 provides information about responding to JES.
3. Wait for another SYNCH order from JES. If the ORDSYR1 field of the SYNCH order parameter list is zero, the FSA should continue RELDS processing for that data set. If the ORDSYR1 field is non-zero, the FSA should process the SYNCH order normally.

ORDSYR3
This flag byte indicates the changes that the FSA is to make to the device characteristics. One or more of the following indicators may be set:

ORDSYS1 B'10000000'
The device is to single space the output.

ORDSYS2 B'01000000'
The device is to double space the output.

ORDSYS3 B'00100000'
The device is to triple space the output.

ORDSYSR B'00010000'
The device is to use data set specified spacing.

ORDSYKP B'00001000'
The FSA is to take checkpoints based on page count.

ORDSYKT B'00000100'
Output checkpointing is to be based on time.

ORDSYKN B'00000010'
The FSA is to disable checkpointing.

ORDSYRL B'00000001'
The FSA is to reload electronic resources (for example, font libraries and overlays) for the data set at the OOP.

ORDSYR4
This flag byte indicates the changes the FSA is to make to the characteristics of the data set at the OOP. JES has rules that the maximum amount of copies that can be printed of a dataset is 255. The FSA is responsible for checking that this rule is enforced. One of the following indicators may be set:

ORDSYCI B'10000000'
The FSA is to increment the copy count of the data set.

ORDSYCD B'01000000'
The FSA is to decrement the copy count of the data set.

ORDSYCR B'00100000'
The FSA is to replace the copy count for the data set.

ORDSYR5
This flag byte indicates the data set processing status that is to be assigned to the data set currently at the OOP. The status depends on the type of interruption performed. The following indicators may be set:

ORDSYDC B'10000000'
The data set at the OOP is complete.

ORDSYDI B'01000000'
The data set at the OOP is incomplete.

ORDSYVA B'00100000'
The checkpoint data for the data set is valid.

ORDSYNV B'00010000'
The checkpoint data for the data set is invalid.

ORDSYR6
This flag byte indicates miscellaneous actions that the FSA is to perform. The following indicators may be set:

ORDSYMV B'10000000'
The FSA is to print the ORDSYMSG message on the output of the data set being synched.

ORDSYDS B'01000000'
The FSA is to reject the synch order if a data set is not currently active at the OOP.

The FSA must respond with the RESP2NDS bit on in the order response area. Chapter 5, “FSI Communication,” on page 17 provides information about responding to JES.

ORDSYSP B'00100000'
The FSA is to print a job trailer page for the data set at the OOP.

ORD6EOG B'00010000'
End of output group.

ORD6CLP B'00010000'
The FSA clears the pipeline when this flag is on.

The FSA issues an FSI RELDS request with the data set processing status of RDSINC for all data sets up to but not including the one active at the operator observation point (OOP).

ORDSYNP
The number of pages that the FSA is to reposition the data set. If this value is zero, the FSA should ignore the reposition by pages request.

ORDSYKI
The checkpoint interval that is to be used for checkpointing. This interval indicates a number of pages or seconds depending on whether the FSA is to perform checkpointing based on page count or elapsed time.

ORDSYCP
The copy count value that is used by the FSA to change the copy count of the data set.

ORDSYSMX
A pointer to the set order parameter list. If this pointer is present, the length of the set order parameter list is not included in the FSILEN value.

ORDSYMSG
The message text that the FSA is to print on the user's output.

Determining Synch Action to be Performed

The FSI ORDER routine uses the ORDID value to determine that the JES order requires actions against the data set at the OOP. The synch order specifically requests the FSA to perform one to four actions against the data set at the OOP. These are:

1. To synchronize the data set to a specified point (indicated by the ORDSYR1 flag byte)
2. To reposition the data set from the point of synchronization (indicated by the ORDSYR2 flag byte)
3. To interrupt printing of the data set (indicated by the ORDSYR5 flag byte)
4. To update device and/or data set characteristics (indicated by the ORDSYR3 and ORDSYR4 flag bytes).

The FSA performs the actions in the order listed above. If an interrupt action is not specified, the FSA updates the data set characteristics along with any synchronization and repositioning actions. If a response to the order cannot be immediately provided to JES, the FSI ORDER routine sets the ORDFLGS1 field in the common order header section equal to ORDARESP. This notifies JES that the response to the order will be returned at a later time by means of an FSIREQ SEND request.

When synch order processing is complete, the FSA responds to JES with the required data. Chapter 5, “FSI Communication,” on page 17 explains how the FSA responds to JES.

The Intervention Order

JES issues an intervention order to the FSA's FSI ORDER routine when a change in forms, flash, or burster-trimmer-stacker specifications is required for the data set that JES is currently assigning to the FSA in response to a GETDS request. The intervention order requires the FSA to process the device buffered data and then ready the device for operator intervention.

Processing the Intervention Order

When JES issues an FSI intervention order, it passes the address of the intervention order parameter list in register 1 to the FSA's FSI ORDER routine. The intervention order parameter list consists of the following sections:

- Common parameter header
- Common order header
- Intervention order dependent section

The following figure shows the connection between the different sections of the FSIREQ parameter list for intervention order processing.

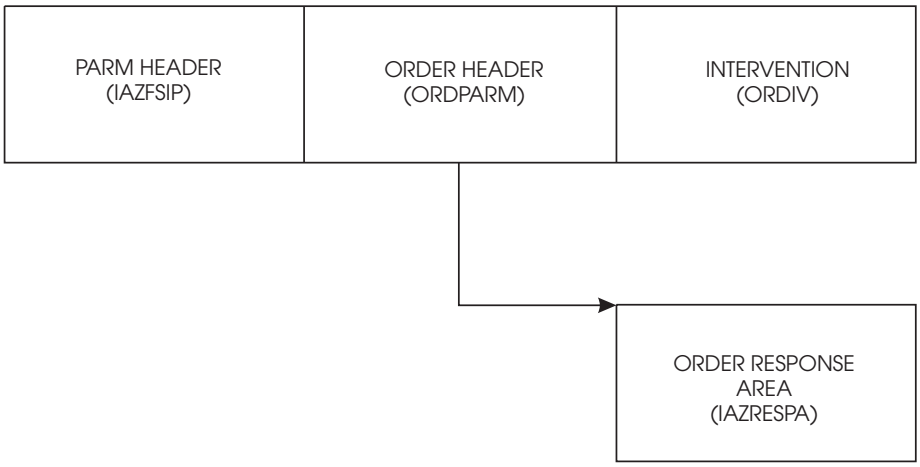

Figure 34. FSIREQ Parameter Lists for Intervention Order Processing

The table below lists the initialized fields, the offsets and lengths of these fields, and the values that JES has assigned. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
Common Parameter Header Section		
FSILEN	4	Length of intervention order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS/FSA IDs
Common Order Header Section		
ORDFDATA	4	A value supplied to JES by the FSS/FSA as a CONNECT parameter (CDFFDATA)
ORDRSPAD	4	Address of the order response area (IAZRESPA)
ORDID	2	ORDINTV

Field Name	Length (bytes)	Value to be assigned
Intervention Order Variable Data Section		
ORDIVF1	1	Intervention type
ORDIVF2	1	Update type
ORDIVBTT	8	BTS intervention token
ORDIVFLT	8	Flash intervention token
ORDIVFOT	8	Forms intervention token
ORDIVCFT	8	CFS intervention token

FSILEN
The length of the entire intervention order parameter list. The intervention order parameter list consists of the IAZFSIP common parameter header section, the common order header section, and the intervention order dependent section.

FSIFUNC
The ORDER function ID number. FSIORDER is the symbolic equate.

FSIFSID
The FSS/FSA IDs that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSA related information. The FSA passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value to the FSA ORDER routine so that it can notify the appropriate FSA.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The intervention order ID number. ORDINTV is the symbolic equate.

ORDIVF1
This flag byte indicates the type of intervention required. The following indicators may be set:

ORDIVRBT B'10000000'
Burster-trimmer-stacker (BTS) intervention is required.

ORDIVRFL B'01000000'
Flash intervention is required.

ORDIVRFO B'00100000'
Forms intervention is required.

ORDIVRCF B'01000000'
Continuous forms stacker (CFS) intervention is required.

ORDIVF2
This flag byte indicates the type of updates required. The following indicators may be set:

ORDIVUBT B'10000000'
A BTS token update is required.

ORDIVUFL B'01000000'
A flash token update is required.

ORDIVUFO B'00100000'
A forms token update is required.

ORDIVUCF B'00010000'
A CFS token update is required.

ORDIVBTT
The token for BTS intervention ('Y' or 'N').

ORDIVFLT
The token for flash intervention (a user-supplied name).

ORDIVFOT
The token for forms intervention (a user-supplied name).

ORDIVCFT
The token for CFS intervention ('Y' or 'N').

The FSI ORDER routine uses the ORDID value to determine that the JES order requests an intervention action. The FSI ORDER routine then either processes the order directly or posts an FSA task to process the order. If the intervention order is for a change in forms or BTS, the FSA needs to ensure all data in the pipeline has reached the data integrity point (DIP) before responding to JES. If the order is for a change in forms, the FSA needs to ensure all data in the buffer has reached the OOP. If a response to the order cannot be immediately provided to JES, the FSI ORDER routine sets the ORDFLGS1 field in the common order header section equal to ORDARESP. This notifies JES that the response to the order will be returned at a later time by means of an FSIREQ SEND request.

When intervention order processing is complete, the FSA responds to JES with the required data. Chapter 5, “FSI Communication,” on page 17 explains how the FSA responds to JES.

Note: When JES receives the response to the intervention order, it issues a setup message to the operator. When the operator replies to the message that the setup is correct, JES issues an FSIREQ POST request to the FSA indicating that GETDS requests can now be satisfied. The FSA should then reissue the GETDS request for the data set.

Notifying JES of Order Completion

When the FSA complete the processing for an order it responds to JES with the required response data. If the FSA is responding to a query, synch, or intervention order, it needs to initialize the RESPRETC field of the order response area (IAZRESPA) with a return code and provide information about the data set at the OOP. For the set order, the FSA needs to initialize only the RESPRETC field indicating whether the order was processed successfully. Refer to Chapter 5, “FSI Communication,” on page 17 for information about responding to a JES order.

SEND Processing

The FSIREQ SEND request causes control to be passed to the FSI SEND service. This service processes the return code in the RESPRETC field of the order response area. If the return code is zero, JES continues processing the order. If the response is to a query, synch, or intervention order, JES retrieves the data set information from the order response area and uses this information to respond to the JES operator command.

INTERVENTION

If the return code in the order response area is not zero, JES issues an error message to the JES operator.

Chapter 11. Stopping an FSS Device

When an operator issues a command to stop a specific device, JES issues a STOP device order to the FSA. When the FSA receives the STOP device order, it completes processing the current data set and then performs its device termination processing. When the device finishes processing the current data set, the FSA does not request any more work.

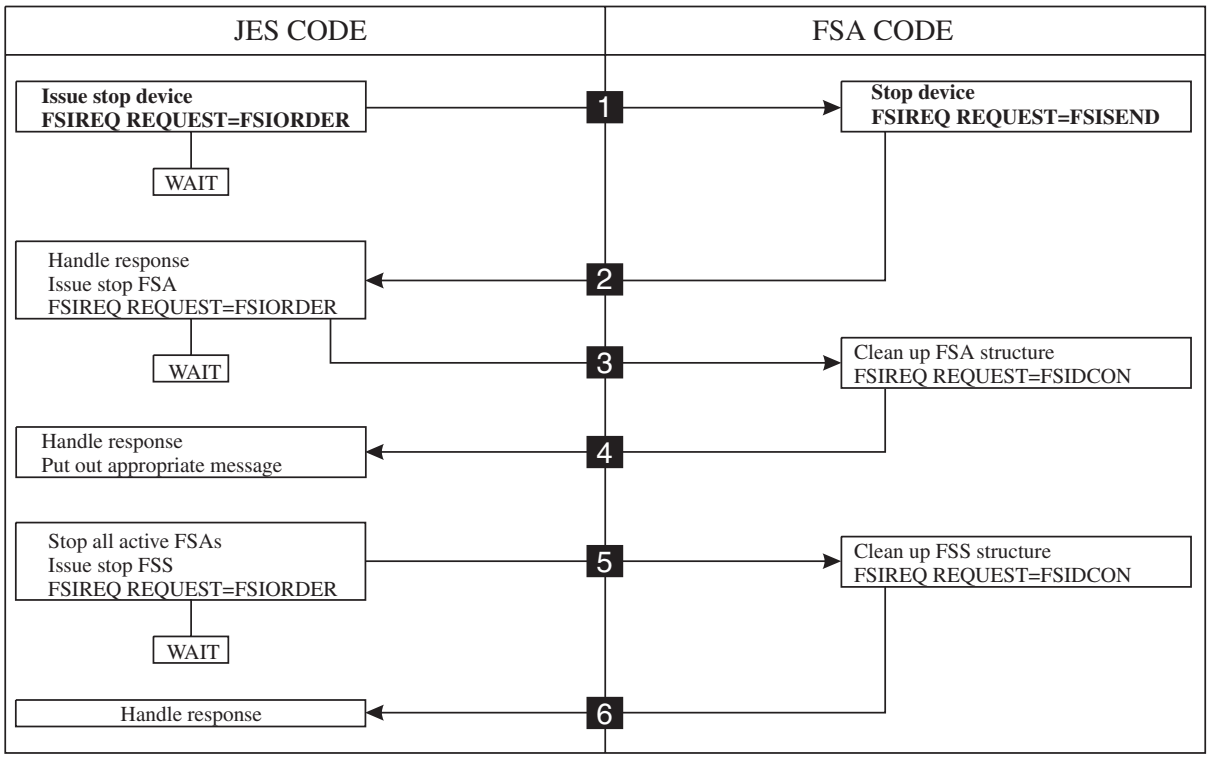

Figure 35. An Overview of FSI Shutdown Processing

Processing the STOP Device Order

To stop a device that is running under an FSS, JES issues the STOP device order to the FSA' FSI order routine. JES passes the address of the STOP device order parameter list in register 1. The parameter list points to the address response area (IAZRESPA). When the FSI ORDER routine receives the order, it is responsible for determining the type of order issued and either posting the appropriate FSA task to process the order or processing the order directly. The value of the ORDID field in the common order header section of the STOP device order parameter list represents the type of order the FSA needs to process.

The STOP device order parameter list consists of the following sections:

- Common parameter header
- Common order header
- STOP order function dependent section

The following figure shows the connection between the different sections of the FSIREQ parameter list for STOP device processing.

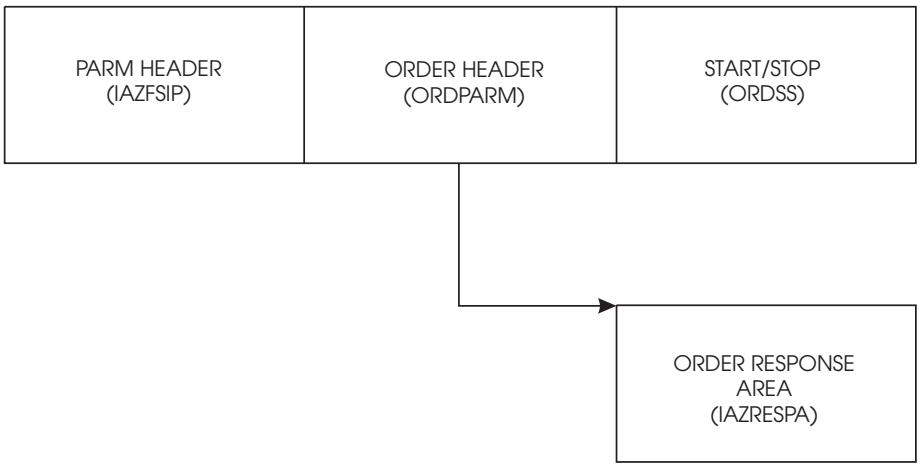

Figure 36. FSIREQ Parameter Lists for STOP Device Processing

The following table shows the parameters that JES initializes for the STOP device order. The values that JES assigns are explained after the table.

Field Name	Length (bytes)	Value JES Assigned
Common Parameter Header		
FSILEN	4	Length of STOP order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS/FSA identifier
Common Order Header		
ORDFDATA	4	Information supplied to JES in the FSA CONNECT parameter list (CDFFDATA)
ORDRSPAD	4	Address of the order response area
ORDID	2	ORDSPDEV
STOP Order Function Dependent Section		
ORDSSSP	4	0
ORDSSF1	1	Type of termination requested
ORDSSID	4	FSA identifier of device to stop
ORDSSAD4	4	Device address in 4-digit format
ORDSSAD	3	Device address in 3-digit format
ORDSSNA	8	Device name

FSILEN
The total length of the STOP order parameter list. The STOP order parameter list is composed of the common parameter header, the common order header and the STOP order function dependent section.

FSIFUNC
The ORDER ID number. JES assigns the symbolic value FSIORDER to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSA-related information. The FSA supplied this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value in the STOP device order parameter list so that the FSA's FSI ORDER routine can activate the appropriate FSA task to process the order.

ORDRSPAD
The address of the order response area (IAZRESPA).

ORDID
The STOP device order ID number. JES assigns the symbolic value ORDSPDEV to this field.

ORDSSSP
This field is set to zero. JES supplies the address of the device initialization area in this field for the START FSA order only.

ORDSSF1
This flag byte indicates the type of device termination requested by JES.

ORDSSNO B'10000000'
The FSA is to terminate the device normally.

ORDSSAB B'01000000'
The FSA is to abnormally terminate the device.

ORDSSDU B'00001000'
The FSA is to take a dump when the device terminates.

ORDSSID
The FSA identifier of the device to stop.

ORDSSSI
The FSS section of the FSA identifier.

ORDSSAI
The FSA section of the FSA identifier.

ORDSSAD4
The 4-digit device address in printable form. If the printer is a non-channel attached device, this field will contain blanks.

ORDSSAD
The 3-digit device address in printable form. If the printer is a non-channel attached device, this field will contain blanks.

ORDSSNA
The device name in printable form.

Notifying JES When the Device is Stopped

When the FSA receives the STOP device order from JES, the FSA decides whether it can respond immediately, or needs to perform additional processing before it can respond. Before responding to the stop device order, the FSA should wait for the device to finish printing the data set the printer is currently working on and push that data set to the stacker.

Refer to Chapter 5, “FSI Communication,” on page 17 for information about responding to the STOP device order.

SEND Processing

When JES receives the SEND request, it processes the return code set by the FSA in the RESPRETC field of the order response area. If the return code is zero, JES issues a STOP FSA order. Refer to Chapter 12, “Stopping an FSA,” on page 107 for more information about the STOP FSA order. If the return code is greater than zero, JES issues another STOP device order. This second STOP device order requests the FSA to abnormally terminate the device and take a dump.

Chapter 12. Stopping an FSA

JES normally issues a STOP FSA order to the FSS in response to the FSA's notification that the device has stopped. After the FSS receives the STOP FSA order from JES, the FSA performs whatever cleanup processing needs to be done and then responds to JES by issuing an FSA level DISCONNECT.

There are other situations where JES issues a STOP FSA order. JES issues a STOP FSA order when the FSA notifies JES (in response to a START device order) that the device could not be started. JES also issues a STOP FSA order if it determines that the FSA must terminate before JES had a chance to stop the device (that is, unrecoverable JES error).

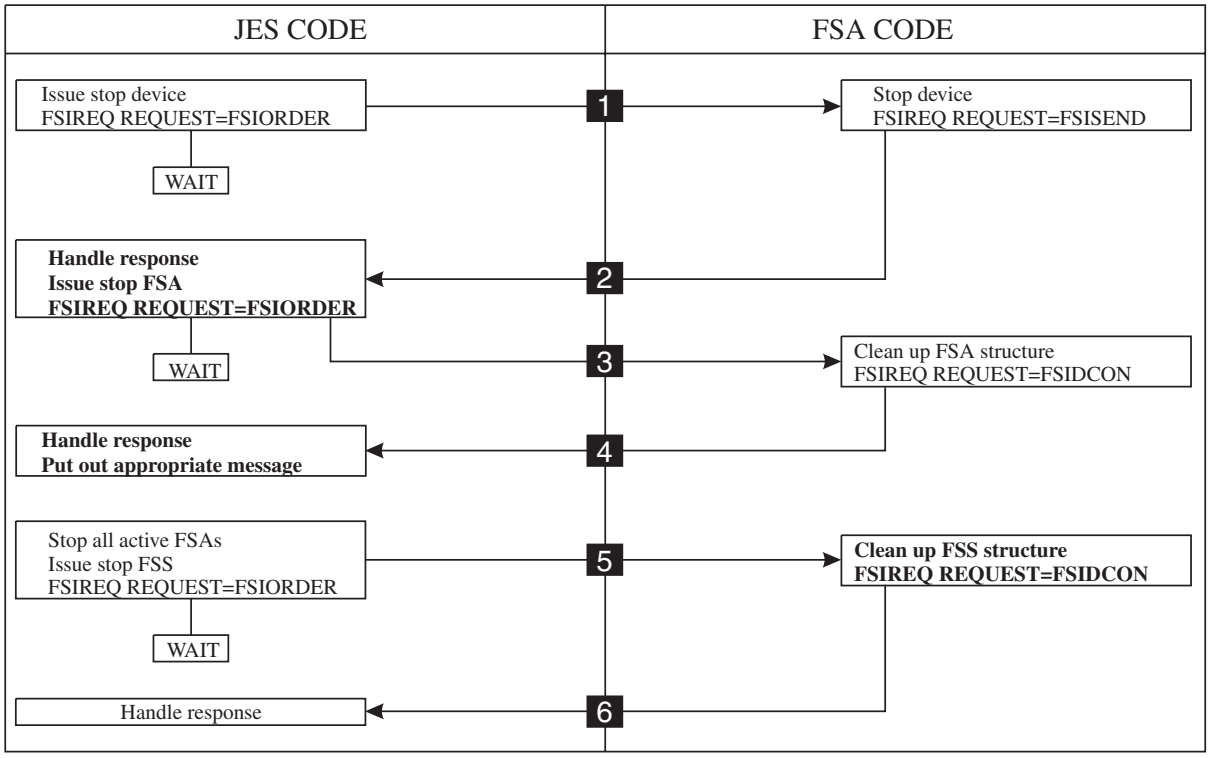

Figure 37. An Overview of FSI Shutdown Processing

Processing the STOP FSA Order

To stop an FSA that is running under an FSS, JES issues the STOP FSA order to the FSS's FSI order routine. Since there can be multiple FSAs running under an FSS, JES issues a STOP FSA order for each FSA in the FSS address space. JES passes the address of the STOP FSA order parameter list in register 1. The parameter list contains the address of the order response area (IAZRESPA). When the FSS' FSI ORDER routine receives the order, it is responsible for determining the type of order issued and then either posting the appropriate FSA task to process the order or processing the order directly. The value of the ORDID field in the common order header section of the STOP FSA order parameter list represents the type of order the FSS needs to process.

The STOP FSA order parameter list consists of the following sections:

- Common parameter header
- Common order header
- STOP FSA order dependent section

The following figure shows the connection between the different sections of the FSIREQ parameter list for STOP FSA processing.

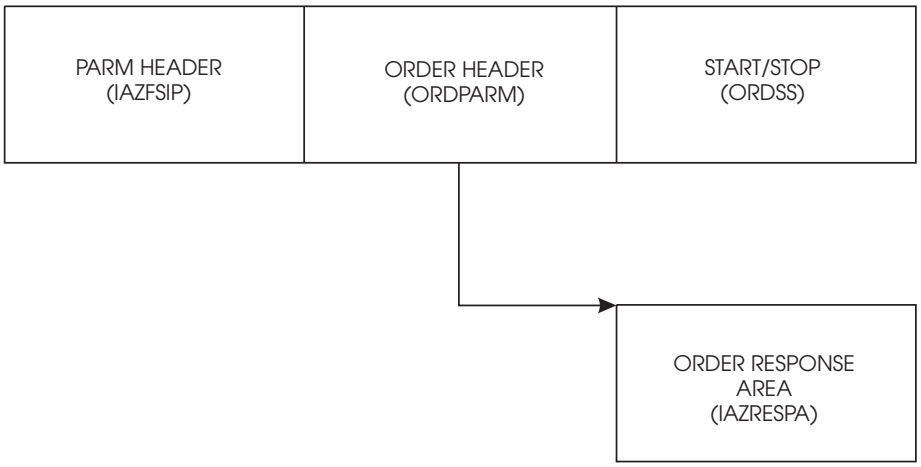

Figure 38. FSIREQ Parameter Lists for STOP FSA Processing

The following table shows the parameters that JES initializes for the STOP FSA order. The values that JES assigns are explained after the table.

Field Name	Length (bytes)	Value JES Assigned
Common Parameter Header		
FSILEN	4	Length of STOP order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS/FSA identifier
Common Order Header		
ORDFDATA	4	Information supplied to JES in the FSS CONNECT parameter list (CDFFDATA)
ORDID	2	ORDSPFSA
START/STOP Order		
ORDSSSP	4	0
ORDSSF1	1	Type of termination requested

FSILEN
The total length of the STOP order parameter list. The STOP order parameter list is composed of the common parameter header, the common order header and the STOP order dependent section.

FSIFUNC
The ORDER ID number. JES assigns the symbolic value FSIORDER to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

ORDFDATA
The address of a control block containing FSS-related information. The FSS passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value in the STOP FSA order parameter list so that the FSS' FSI ORDER routine can activate the appropriate FSA task to process the order.

ORDID
The STOP FSA ID number. JES assigns the symbolic value ORDSPFSA to this field.

ORDSSSP
This field is set to zero. JES supplies the address of the device initialization area in this field for the START FSA order only.

ORDSSF1
This flag byte indicates the type of termination requested. One or more of the following indicators can be set:

ORDSSNO B'10000000'
The FSA is to terminate normally.

ORDSSAB B'01000000'
The FSA is to abnormally terminate.

ORDSSDU B'00001000'
The FSA is to take a dump when it terminates.

Preparing for FSA Disconnect

JES notifies the FSS task when an FSA is to be stopped. The FSS is responsible for notifying the appropriate FSA that it is to terminate. The FSA performs cleanup processing and then issues the FSA level DISCONNECT. Cleanup processing includes issuing RELDS requests for any data sets not yet released, freeing any storage, unallocating the device, etc. Refer to “Releasing a SYSOUT Data Set” on page 80 for more information about RELDS processing.

Before the FSA can issue the DISCONNECT, it must:

- Provide an 18 word save area
- Initialize the DISCONNECT parameter list.

Initializing the FSIREQ DISCONNECT Parameter List

The FSA needs to initialize the following parameters before it issues the FSIREQ DISCONNECT request.

Field Name	Value (bytes)	Value to be Assigned
General Header		
FSILEN	4	Length of DISCONNECT parameter list
FSIFUNC	4	FSIDCON
FSIFSID	4	The FSS/FSA identifier
DISCONNECT Function Dependent Area		
CDFFLGR1	1	Specifies a normal or abnormal termination
CDFSTOR	4	Address of storage for SSOB/SSIB pair
CDFSSID	4	Name of the JES to which the FSA is connected

FSILEN
The length of the entire DISCONNECT parameter list. The DISCONNECT parameter list consists of both the IAZFSIP common header section and the DISCONNECT function dependent section.

FSIFUNC
The DISCONNECT function ID number. The FSA assigns the symbolic value FSIDCON to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

FSIFSSID
This field contains the FSS portion of the FSS/FSA identifier.

FSSFSAID
This field contains the FSA portion of the FSS/FSA identifier.

CDFFLGR1
Specifies the type of termination requested by the FSA. One of the following indicators can be set:

CDFNORM B'10000000'
Specifies a normal DISCONNECT. The FSA is disconnecting in response to a STOP FSA order or as a result of FSA-initiated termination.

CDFABNOR B'01000000'
Specifies an abnormal DISCONNECT. The FSA is disconnecting because of an unrecoverable error.

CDFSTOR
Address of the storage for the SSOB/SSIB pair.

CDFSSID
Name of the JES to which the FSA is connected. If this parameter is not specified, the FSA is disconnected from the primary JES defined to your installation.

Issuing the FSIREQ DISCONNECT Request

When the FSA has completed initializing the DISCONNECT parameter list, it uses the FSIREQ macro to invoke the FSI DISCONNECT service. The format of this macro call is:

```
FSIREQ REQUEST=FSIDCON,PARM=DISCONNECT parm-list-address,
       TARGET=JES,FSID=fsid
```

Refer to Chapter 4, “The FSIREQ Macro,” on page 13 for a complete description of each operand on this macro and any defaults that may be taken.

FSA-Initiated Termination

If the FSA becomes aware that it needs to initiate termination (for example, VTAM lines come down), it issues a SEND request to JES to inform JES about the termination. JES determines the current stage of FSA processing and then attempts to shut down the FSA as normally as possible. JES issues a STOP device order and then a STOP FSA order. The FSA is expected to respond normally to these orders.

Initializing the FSIREQ SEND Parameter List

The FSA needs to initialize the following parameters before it issues the FSIREQ SEND request.

Field Name	Value (bytes)	Value to be Assigned
General Header		
FSILEN	4	Length of SEND parameter list
FSIFUNC	4	FSISEND
FSIFSID	4	The FSS/FSA identifier
SEND Function Dependent Area		
SNDTYPE	1	Special types of SEND

FSILEN
The length of the entire SEND parameter list. The SEND parameter list consists of both the IAZFSIP common header section and the SEND function dependent section.

FSIFUNC
The SEND function ID number. The FSA assigns the symbolic value FSISEND to this field.

FSIFSID=
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

FSIFSSID
This field contains the FSS portion of the FSS/FSA identifier.

FSSFSAID
This field contains the FSA portion of the FSS/FSA identifier.

SNDTYPE
The SNDTYPE ID number. The FSA assigns the symbolic value SNDTYFIT to this field. SNDTYFIT indicates that the SEND request is a request for FSA-initiated termination.

SNDTYRSP
Response to an order.

SNDTYTDS
Send requested via GDSFLGR1 indicating DS reached OOP.

SNDTYFIT
Request for FSA term.

SNDTYINT
Unsolicited device intervention detected from the FSA.

SNDTYICL
Unsolicited device intervention cleared from the FSA.

SNDTYDNR
Unsolicited device not responding received from the FSA.

SNDTYDCL
Unsolicited device not responding cleared from the FSA.

SNDTYEXT
If one of the extended send types in SNDTYP2 is being used, this value must be set in SNDTYPE.

SNDTYP2
This is an extended send type. If this send type is used, SNDTYEXT must be set in SNDTYPE. SNDTYP2 can be set to one of the following values:

SNDE580K
Unsolicited request to issue an EOD-OK ENF58 signal.

SNDE58ER
Unsolicited request to issue an EOD-Error ENF58 signal.

Issuing the FSIREQ SEND Request

When the FSA has completed initializing the SEND parameter list, it uses the FSIREQ macro to invoke the FSI SEND service. The format of this macro call is:

```
FSIREQ REQUEST=FSISEND,PARM=SEND parm-list-address,
       TARGET=JES,FSID=fsid
```

Refer to Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and any defaults that may be taken.

SEND Processing

If JES receives the SEND request for an FSA-Initiated termination:

- Before JES has issued the STOP device order, JES issues a STOP device order to the FSA.
- After JES has issued the STOP device order but before it has issued the STOP FSA order, JES issues a STOP FSA order to the FSA.
- After JES has issued the STOP FSA order, JES awaits the FSA's response from the STOP FSA order (FSA-level DISCONNECT).

DISCONNECT FSA Processing

When JES receives the FSA DISCONNECT request from the FSA, JES validates the FSA information and decides whether it can normally terminate the FSA. As part of FSA disconnect processing, JES issues RELDS requests for all data sets not yet released.

How JES Handles Logic Errors and Abends

During the validation of the FSA information, JES may determine that it can not disconnect the FSA. If JES could not disconnect the FSA, the value of the SSOBRETN field of the SSOB will be non-zero. This non-zero value indicates that the FSS should abnormally terminate the FSA.

How JES Monitors Timing of FSA DISCONNECT

When JES issues the STOP FSA order to the FSS, JES starts a timer. If the FSS does not respond with a FSA DISCONNECT within a specific length of time, JES issues a message to the operator.

Chapter 13. Stopping an FSS

JES issues a STOP FSS order to the FSS when an operator command requests that JES be shut down. JES will also issue the STOP FSS order when all printers are shut down at the installation anticipates that the FSS will no longer be required. After the FSS receives the STOP FSS order from JES, the FSS performs its cleanup processing and then responds to JES by issuing an FSS level DISCONNECT.

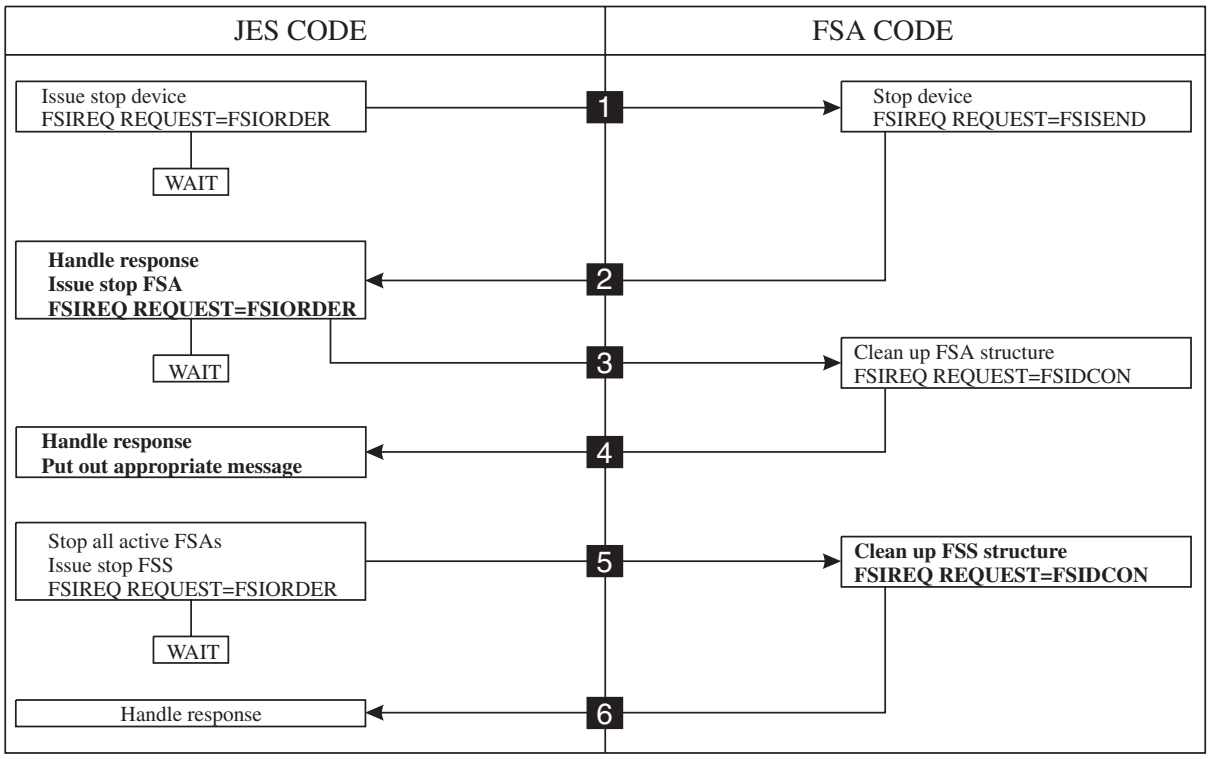

Figure 39. An Overview of FSI Shutdown Processing

Processing the STOP FSS Order

To stop an FSS, JES issues the STOP FSS order to the FSS' FSI ORDER routine. The FSS supplied the address of its FSI ORDER routine to JES during FSS CONNECT processing in the CDFAD field of the CONNECT parameter list. JES passes the address of the STOP FSS order parameter list. The parameter list contains a pointer to the JES provided order response area (IAZRESPA).

When the FSI ORDER routine receives the order, it is responsible for determining the type of order issued and then either posting an FSS task to process the order or processing the order directly. The value of the ORDID field represents the type of order the FSS needs to process.

The STOP FSS order parameter list consists of the following sections:

- Common parameter header
- Common order header
- STOP FSS order dependent section

The following figure shows the connection between the different sections of the FSIREQ parameter list for STOP FSS processing.

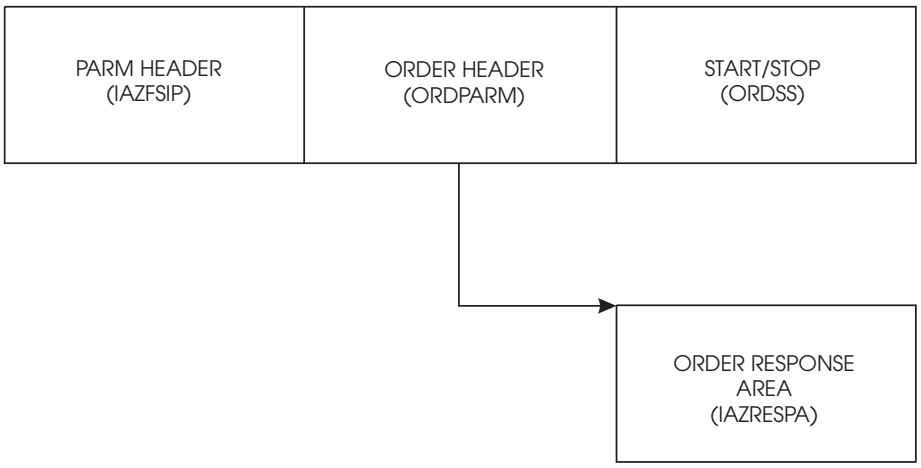

Figure 40. FSIREQ Parameter Lists for STOP FSS Processing

The following table shows the parameters that JES initializes for the STOP FSS order. The values that JES assigns are explained in the table.

Field Name	Length (bytes)	Value JES Assigned
Common Parameter Header		
FSILEN	4	Length of STOP order parameter list
FSIFUNC	4	FSIORDER
FSIFSID	4	The FSS identifier
Common Order Header		
ORDFDATA	4	Information supplied to JES in the FSS CONNECT parameter list (CDFFDATA)
ORDID	2	ORDSPFSS
STOP Order		
ORDSSSP	4	0
ORDSSF1	1	Type of termination requested

FSILEN
The total length of the STOP order parameter list. The STOP order parameter list consists of the common parameter header, the common order header and the STOP order header.

FSIFUNC
The ORDER ID number. JES assigns the symbolic value FSIORDER to this field.

FSIFSID
The FSS identifier that JES assigned when it started the FSS.

ORDFDATA
The address of a control block containing FSS-related information. The FSS passed this address to JES in the CDFFDATA field of the CONNECT parameter list. JES returns this value in the STOP FSS order parameter list so that the FSS' FSI ORDER routine can cause the appropriate FSS task to process the order.

ORDID
The STOP FSS ID number. JES assigns the symbolic value ORDSPFSS to this field.

ORDSSSP
JES supplies this area for the START FSA order only.

ORDSSF1
This flag byte contains the type of termination JES is requesting.

ORDSSNO B'10000000'
The FSS should terminate normally.

ORDSSAB B'01000000'
The FSS should abnormally terminate because of an unrecoverable JES error.

ORDSSDU B'00001000'
The FSS will take a dump when it terminates.

Preparing for FSS Disconnect

When the FSS receives the STOP FSS order from JES, the FSS performs cleanup processing and then issues the FSS level DISCONNECT. Cleanup processing includes issuing FSA disconnects for all FSAs running under the FSS that are still connected, freeing storage, deleting ESTAEs, etc. Refer to "Preparing for FSA Disconnect" on page 109 for more information about FSA level disconnects.

Before the FSS can issue the DISCONNECT, it must:

- Provide an 18 word save area
- Initialize the DISCONNECT parameter list.

Initializing the FSIREQ DISCONNECT Parameter List

The FSS needs to initialize the following parameters before it issues the FSIREQ DISCONNECT request.

Field Name	Value (bytes)	Value to be Assigned
General Header		
FSILEN	4	Length of DISCONNECT parameter list
FSIFUNC	4	FSIDCON
FSIFSID	4	The FSS/FSA identifier
DISCONNECT Function Dependent Area		
CDFFLGR1	1	Specifies a normal or abnormal termination
CDFSTOR	4	Address of storage for SSOB/SSIB pair
CDFSSID	4	Name of the JES to which the FSS is connected

FSILEN
The length of the entire DISCONNECT parameter list. The DISCONNECT parameter list consists of both the IAZFSIP common header section and the DISCONNECT function dependent section.

FSIFUNC
The DISCONNECT function ID number. The FSS assigns the symbolic value FSIDCON to this field.

FSIFSID
The FSS/FSA identifier that JES assigned when it started the FSS and FSA.

FSIFSSID
This field contains the FSS portion of the FSS/FSA identifier.

FSSFSAID
This field contains the FSA portion of the FSS/FSA identifier.

CDFFLGR1
Specifies the type of termination requested by the FSS.

CDFNORM B'10000000'
Specifies a normal DISCONNECT. The FSS is disconnecting in response to a STOP FSS order.

CDFABNOR B'01000000'
Specifies an abnormal DISCONNECT. The FSS is disconnecting because of an unrecoverable error.

CDFSTOR
Address of the storage for the SSOB/SSIB pair.

CDFSSID
Name of the JES to which the FSS is connected. If this parameter is not specified, the FSS is disconnected from the primary JES defined to your installation.

Issuing the FSIREQ DISCONNECT Request

When the FSS has completed initializing the DISCONNECT parameter list, it uses the FSIREQ macro to invoke the FSI DISCONNECT service. The format of this macro call is:

```
FSIREQ REQUEST=FSIDCON,PARM=DISCONNECT parm-list-address,
       TARGET=JES,FSID=fsid
```

Refer to Chapter 4, "The FSIREQ Macro," on page 13 for a complete description of each operand on this macro and any defaults that may be taken.

DISCONNECT FSS Processing

When JES receives the FSS DISCONNECT request from the FSS, JES validates the FSS information and decides whether it can normally terminate the FSS address space.

How JES Handles Logic Errors and Abends

During the validation of the FSS information, JES may determine that it can not disconnect the FSS. If JES could not normally disconnect the FSS, the value of the SSOBRETN field of the SSOB will be non-zero. This non-zero value indicates that the FSS should abnormally terminate. If the FSS abnormally terminates before it disconnects the FSAs running under it, JES will disconnect the FSAs.

How JES Monitors Timing of FSS DISCONNECT

When JES issues the STOP FSS order to the FSS, JES starts a timer. If the FSS does not respond with a FSS DISCONNECT within a specific length of time, JES terminates the FSS address space.

Chapter 14. FSS Output Descriptor Support

FSS output descriptor support allows a functional subsystem to use JCL that is not known to JES. An FSS can use this JCL to support sophisticated printers. Users can specify complex printing requirements for data sets that are processed by FSS devices.

Data set printing requirements are specified on the OUTPUT JCL statement. This statement supports several FSS-specific parameters such as FORMDEF and PAGEDEF. *z/OS MVS JCL Reference* describes each of the parameters on the OUTPUT JCL statement.

The following topics describe:

- The scheduler JCL facility and how it interfaces with JES and the FSS to provide FSS output descriptor support.
- An overview of output descriptor processing as it relates to the FSS.
- The tasks involved in retrieving output descriptor information.

The Scheduler JCL Facility

The scheduler JCL facility (SJF) controls output descriptor information processing. This facility consists of a set of service routines that interface with JES and the FSS/FSA to:

- Store JCL keyword subparameter information in the scheduler work block. For each JCL statement, SJF builds one or more scheduler work blocks and then chains these scheduler work blocks together.
- Update the information in the scheduler work blocks if an operator command changes a keyword value.
- Retrieve JCL information from scheduler work blocks.

An Overview of OUTPUT Processing

At a job level, a user can specify output processing specifications on the OUTPUT JCL statement. During the job's execution, the converter/interpreter verifies the JCL and invokes SJF to save the JCL keyword information in scheduler work blocks. Although JES does not use the FSS-specific keyword information, it passes the scheduler work block token to the FSA when it assigns the corresponding data set to the FSA.

The FSI GETDS service invokes the SJF PUT scheduler work block or SJF UPDATE service to obtain a token for the data set assigned to the FSA. The FSA can use this to retrieve data set characteristics specified on the OUTPUT JCL statement. Figure 41 on page 118 illustrates OUTPUT JCL Processing for the FSS.

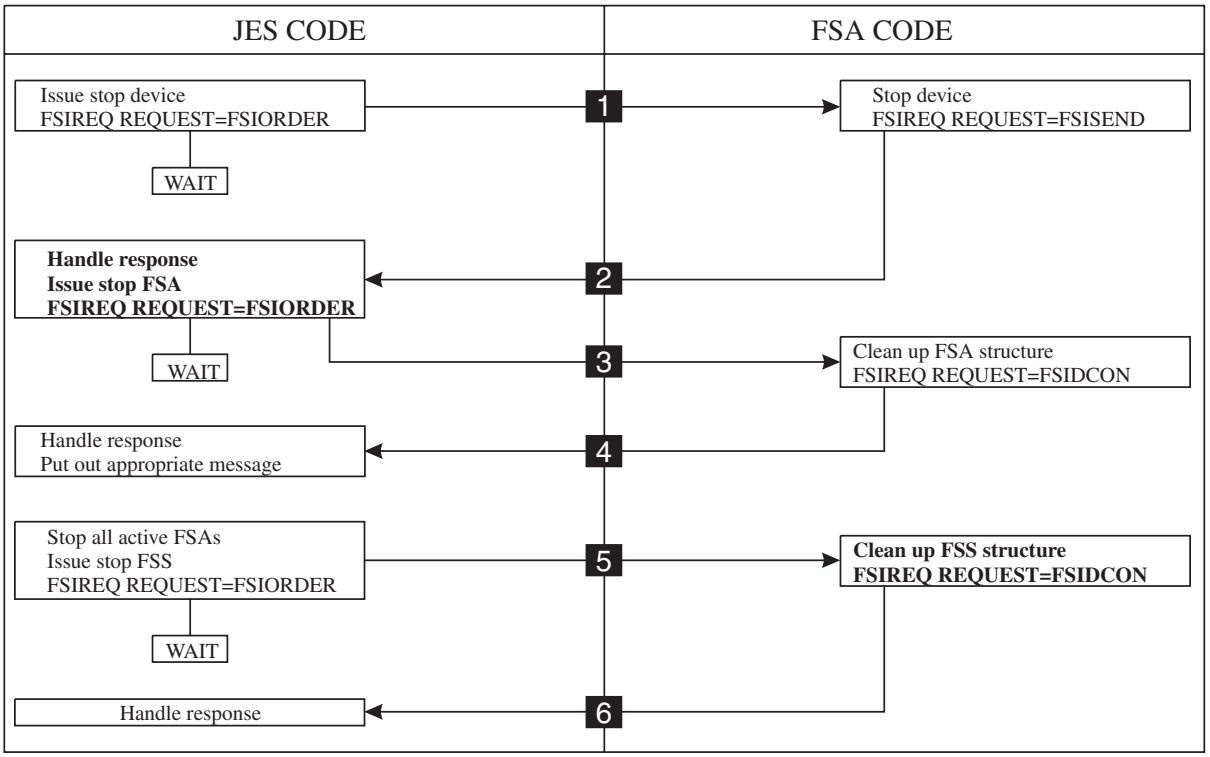

Figure 41. OUTPUT JCL Processing

Using SJF Services

The FSA invokes SJF services to retrieve JCL keyword subparameter information specified on the OUTPUT JCL statement.

To retrieve JCL keyword information specified on the OUTPUT JCL statement, the FSA uses the token obtained from GETDS processing to invoke the SJF RETRIEVE service.

Requirements for Using SJF Services

The following are rules and restrictions for using the SJF RETRIEVE service.

- The FSS/FSA must be in supervisor state, and run in key 0-7.
- The FSS/FSA must provide an 18-word save area.
- If the caller's storage area is referenced by SJF, it must not be fetch protected.
- SJF services are not available in cross-memory mode.
- Use of the multiple invocation facility of SJF is limited to under one task.

The Scheduler JCL Facility RETRIEVE Request

The FSS/FSA invokes the SJF RETRIEVE service to retrieve the JCL keyword subparameter information. The FSA supplies the token to the service to identify the correct scheduler work block chain.

The SJF mapping macros related to the RETRIEVE service are:

- IEFSJREP - maps the SJF RETRIEVE parameter list

- IEFZB4D0 - maps the text unit pointer list and text units
- IEFSJRC - maps the SJF reason codes.

The following topics describe the tasks the FSA performs to invoke the SJF RETRIEVE service and the associated SJF processing.

Initializing the Keyword List

The FSA needs to provide a keyword list (SJRELIST) to the SJF RETRIEVE service. The keyword list contains paired fields, each pair consists of a keyword field and a pointer field. In the list, the FSA specifies the JCL keywords for which information is to be retrieved. For each keyword specified, the SJF RETRIEVE service returns a pointer to the text unit pointer list associated with the keyword.

The following table shows the SJRELIST paired fields and their lengths. The fields that the FSA initializes are indicated.

Field Name	Length (bytes)	Value to be assigned
SJRELIST		
SJREKEYW	8	keyword 1 (supplied by FSA)
SJRETPAD	4	pointer (supplied by SJF)
SJREKEYW	8	keyword 2 (supplied by FSA)
SJRETPAD ⋮	4 ⋮	pointer (supplied by SJF) ⋮

Establishing a Storage area

For each SJF RETRIEVE request, the FSA needs to establish a storage area in which SJF is to return the output descriptor information. The size of this storage area depends on the number of keywords for which the FSA is requesting information. See Figure 42 on page 121 for a graphical representation of the SJF control blocks returned in the storage area.

The FSA specifies the address and size of this storage area in the SJF RETRIEVE parameter list.

Initializing the SJF RETRIEVE Parameter List

The FSA needs to initialize certain fields of the SJF RETRIEVE parameter list (IEFSJREP). The table below lists the required fields, the lengths of these fields, and the values that the FSA must assign. Detailed descriptions of the value assignments follow this table.

Field Name	Length (bytes)	Value to be assigned
IEFSJREP Parameter List		
SJREID	4	'SJRE'
SJREVERS	1	SJRECVER
SJRELEN	2	SJRELGTH
SJRESTOR	4	Local storage pointer or zero
SJREJDVT	8	GDSJDVTN or zeroes

Field Name	Length (bytes)	Value to be assigned
SJRETOKN	8	output descriptor block chain token (GDSOUTPT or SJFNTOKN)
SJREAREA	4	Storage area address
SJRESIZE	2	Size of storage area
SJRENKWD	2	Number of keywords passed
SJREKWDL	4	Keyword list address

SJREID
The identifier 'SJRE' of the RETRIEVE parameter list.

SJREVERS
The current version number of the SJF RETRIEVE service. The FSS/FSA assigns the symbolic equate SJRECVER to this field.

SJRELEN
The length of the RETRIEVE parameter list. The FSS/FSA assigns the symbolic equate SJRELGTH to this field.

SJRESTOR
If this is the first SJF RETRIEVE request, the FSS/FSA sets this field to zero. If this is not the first request, the FSS/FSA provides the local storage pointer returned from the previous FIND output descriptor information request.

SJREJDVT
The FSA initializes this field with the name of the JCL definition vector table (JDVT) returned from FSI GETDS processing in the GDSJDVTN field of the GETDS parameter list.

SJRETOKN
The scheduler work block chain token. The FSA initializes this field with the token returned from GETDS processing in the GDSOUTPT field of the GETDS parameter list.

SJREAREA
The address of the storage area in which SJF is to return the output descriptor information in the form of text units.

SJRESIZE
The amount of storage allocated for the output descriptor information.

SJRENKWD
The number of keywords passed in the keyword list (SJRELIST).

SJREKWDL
The address of the keyword list (SJRELIST).

Issuing the SJFREQ RETRIEVE Request

When the FSS has completed initializing the IEFSJREP parameter list, it issues the SJFREQ macro to invoke the SJF RETRIEVE service.

When the FSS has completed initializing the IEFSJREP parameter list, it issues the SJFREQ macro to invoke the SJF RETRIEVE service. The format of this macro call is:

```
SJFREQ REQUEST=RETRIEVE
```

SJF RETRIEVE Processing

The SJF RETRIEVE service uses the token provided in the IEFSJREP parameter list to locate the indicated scheduler work block chain. When the scheduler work block chain is found, the service retrieves the text units associated with each keyword specified in SJRELIST.

The SJF RETRIEVE service next builds a text unit pointer list for each keyword and supplies a pointer to the list in the keyword's corresponding pointer field (SJRETPAD) of SJRELIST. The text unit pointer list contains pointers to the individual text units associated with the keyword.

Information Returned from SJF RETRIEVE Processing

On return from SJF RETRIEVE processing, the FSA-provided storage area contains the text unit pointers list and the individual text units associated with each keyword. The keyword list (SJRELIST) contains paired fields, each pair consisting of a JCL keyword and a pointer to the text unit pointers lists for that keyword. Figure 42 shows the SJF control blocks returned from SJF RETRIEVE processing and their relationships.

Support for ESS Keywords

JES sets the CDFS1ESS bit of the FSS CONNECT parameter list to indicate that JES supports ESS keywords. These keywords specify the name, address, room and department associated with the OUTPUT.

Writing Information into SMF Records

If your system uses JES2, output descriptor information can be written into the SMF records that the FSS/FSA produces. These records include types 6, 24 and 57.

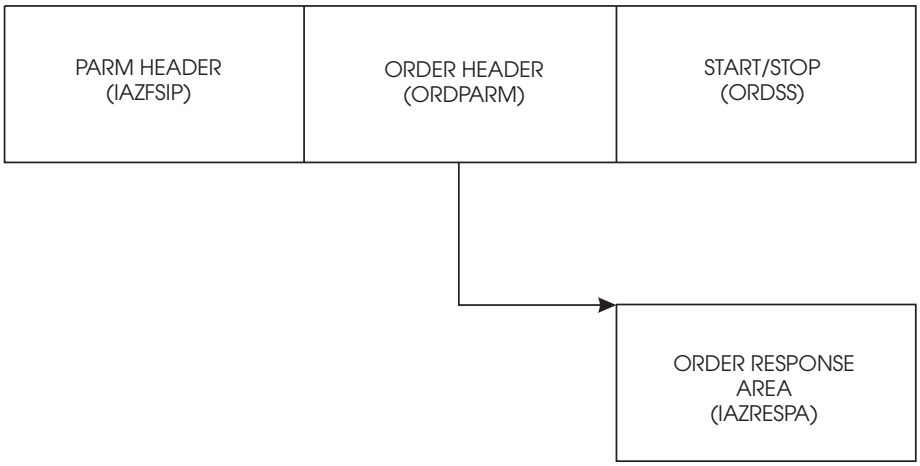

Figure 42. SJF Control Blocks Returned from SJF RETRIEVE

Chapter 15. FSI Trace

This section provides Diagnosis, Modification or Tuning Information.

Through the use of the generalized trace facility (GTF), the FSI provides a method of diagnosing problems that might occur in the FSS address space. Because FSI tracing may slow system performance, you should request FSI tracing only when a problem has occurred.

Using GTF to Trace FSI Communication

The generalized trace facility (GTF) collects trace data about events occurring during the interaction between JES and the FSS or FSA. You can tailor the type of tracing you want performed by specifying what FSI function calls and/or what FSS-driven devices are to be traced.

Tracing FSI communications consists of the following steps:

- Starting GTF
- Specifying GTF trace options
- Recreating the problem
- Stopping GTF.

Starting GTF

To request FSI tracing, ask the operator to enter the following command to start GTF:

```
S GTF.identifier,devname,volserial,(time=yes)
```

where:

identifier
specifies the user-specified name identifying this specific GTF session.

devname
specifies the device number or device type of an output device to contain the trace data set.

volserial
indicates the serial number of a magnetic tape or direct access volume that is to contain the trace data set.

time=yes
requests that every logical trace record is to be time-stamped in addition to the block time stamp associated with every block of data.

Note: For more information about the parameter values on the START GTF command, refer to *z/OS MVS Diagnosis: Tools and Service Aids*.

In response to the START GTF command, GTF issues the following message that asks the operator to enter trace operations:

```
xx AHL100A SPECIFY TRACE OPTIONS
```

Specifying GTF Trace Options

In response to the SPECIFY TRACE OPTIONS message, you can decide to trace the FSI communications by the specific FSI function call or the specific FSS-driven device, or both. To ask GTF to prompt you for both the specific event id and the procname for the FSS-driven device, respond:

```
r xx,trace=usrp,jobnamep
```

GTF responds with the following message, asking for the event ids and the name of the FSS:

```
xx AHL101A SPECIFY TRACE EVENT KEYWORDS--USR=,JOBNAME=
```

Each FSI function call is assigned a GTF event id as follows:

FSI Function Call	Event id (usr=)
ORDER	F54
POST	F55
GETDS	F56
GETREC	F57
FREEREC	F58
RELDS	F59
CHKPT	F5A
SEND	F5B
CONNECT	F5C
DISCONNECT	F5D

To get FSI tracing for a set of specific function calls, respond to message AHL101A with the user event ids for that set of FSI function calls.

To get FSI tracing for a specific FSS, respond to message AHL101A with the jobname parameter set equal to the procname specified on your JES FSSDEF initialization statement. Use the following table to determine the value to code for the jobname parameter to give you the trace you want.

Type of Tracing	Value of jobname Parameter	Result of Trace
Traces originating from the FSS	FSS procname (from the FSSDEF initialization statement)	All specified user events originating from this FSS.
Traces originating from JES2	JES procname	All specified user events originating from JES2. **Note:** This trace results in data for each active FSS.
Traces originating from JES3	FSS procname (from the FSSDEF initialization statement)	All specified user events originating from this FSS.

The following example is a response to message AHL101A with GTF tracing for all ORDER and POST function calls processed in the FSS whose procname is fss1:

```
r xx,usr=(f54,f55),jobname=(fss1)
```

GTF then issues the following message to ask if you want to specify more options, or are finished specifying the trace options:

```
xx AHL102A CONTINUE TRACE DEFINITION OR REPLY END
```

If you are finished specifying the GTF options, respond:

```
r xx,end
```

GTF then issues two messages, the first to confirm your trace options and the second to allow you to re-specify the options if they were entered incorrectly:

```
AHL103I TRACE OPTIONS SELECTED--USR=(f54,f55),JOBNAME=(fss1)

xx AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
```

If you specified the trace options correctly, respond:

```
r xx,u
```

Finally, GTF issues the following message to specify that GTF tracing is now in effect:

```
AHL031A GTF INITIALIZATION COMPLETE
```

Recreating the Problem

You can now issue JES commands to recreate the conditions under which the problem occurred. In this example, GTF captures the information exchanged during the ORDER and POST FSI function calls. You may need additional information to help diagnose the problem beyond that which is provided in the FSI trace records. See Table 5 on page 128 for a summary of the information that is provided in the FSI trace records. If you decide that you need additional information you can use the USERDATA area to record that information. Use the following procedure to establish a user data area to record additional trace information:

1. GETMAIN storage for the user data area.
2. Place the address of this user data area into the FSITEXT field in the IAZFSIP parameter list common header area.
3. Use the FSIUDATA mapping (in the IAZFSIP mapping macro) to fill in the following:

   FSIUDLEN
   Length of the user data area, which includes the FSIUDLEN and FSIUDNAM fields as well as the actual user data. This length cannot exceed 2,000 bytes.

   FSIUDNAM
   Eight-character routine name that generates the FSI function call.

   FSIUDTXT
   User specified data to be generated in the FSI trace record when GTF tracing is active. Include information that is not available in the IAZFSIP. For example, the name of the data set that was printing when the problem occurred.

Stopping GTF

After you have collected the necessary data, issue the following command to stop GTF:

```
P identifier
```

where:

identifier
specifies the user-specified name identifying this specific GTF session.

Note: For more information about the parameter values on the STOP GTF command, refer to *z/OS MVS Diagnosis: Tools and Service Aids*.

Viewing FSI Trace Data

Use the MVS IPCS facility to view the information gathered by GTF. These are the steps:

- Define the default data set name used by IPCS to the data set name that contains the trace data
- Enter this command from the IPCS command panel:

```
gtf usr(f54, f55, f56) terminal
```

The formatted trace records are displayed on the terminal. For more information on the GTF command, enter:

```
h gtf
```

Reading GTF Records

Use the following example to understand the contents of the different sections of the FSI trace record. Table 5 on page 128 follows the example. Use this table to determine the parts of the FSI trace record that are displayed for each FSI function call.

```
IAZFSIH: 00000000
TOD..... 01/03/89 16:19:08

         **********************************
         **       FSI TRACE RECORD       **
         **            CALL              **
         **          FREE REC            **
         **********************************
NAME..... PRINTR2   FSID..... 00030001  SEQ..... 00000001  FLAG... .. 01
1 TOD..... 01/03/89 16:19:08
D7D9C9D5  E3D9F240  00030001  00000001  01F0F161                          | PRINTR2         01/        |
IAZFSIP: 00000000
+0000  LEN..... 00000038  FUNC..... 00000005  FSSID.... 0003       FSAID.... 0001      RESN..... 00000000
2 +0010  TEXT..... 00000000
+0000  00000038  00000005  00030001  00000000  00000000  00000000             |                            |
FLRPARM: 00000000
+0000  INDX..... 0010D3B0  DSID..... 9FA9F27A  67C1D401  00000000   EXTN..... 000B5208
3 +0000  0010D3B0  9FA9F27A  67C1D401  00000000  000B5208   D1C5E2F3  00000000    |   L0Z2: AM         JES3     |
+001C  00000000                                                                  |                             |
FSIGPRS: 00000000
R00..... 01FD4728  R01..... 000415DE  R02..... 81A23000  R03..... 00000001  R04..... 000B51C8  R05..... 81A23000
4 R06..... 000B4A80  R07..... 0004380B  R08..... 00CACEE0  R09..... 0010D3B0  R10..... 000B51E0  R11..... 0004280C
R12..... 0004180D  R14..... 00000000  R15..... 000BEE08
          BLOCKID.. USERDATA  BLOCKLEN. 00000000
 5        BLOCKID.. RSV1      BLOCKLEN. 00000000
          BLOCKID.. IAZIDX    BLOCKLEN. 00000138
```

1 - The **common header** contains the following fields:

NAME
The FSA name, if the FSA had been started; otherwise, the FSS name. To determine if the FSA has been started, look at the FSID field. If the last four characters of the FSID field are zeros, the FSA has not been started.

FSID
Specifies a value that uniquely identifies the FSS or FSA. JES assigns the FSS an identifier of the form xxxx0000, where xxxx is a unique number. JES assigns the FSA an identifier of the form xxxxyyyy, where xxxx corresponds to the controlling FSS identifier, and yyyy is a unique number for each FSA.

SEQ
Specifies the order in which this event occurred for this specific FSID.

FLAG
Specifies whether the trace record is for the function call (01) or for the return from the function call (81).

TOD
Specifies the time of day the event was recorded.

2 - The **IAZFSIP common header** contains the following fields:

LEN
Specifies the total length of the parameter list for this function call.

FUNC
The function id number of this function call. Refer to Appendix B, "Numeric Values of FSI Services and Orders," on page 141 for the values of the function id number.

FSSID
The FSS identifier that JES assigned when it started the FSS.

FSAID
The FSA identifier that JES assigned when it started the FSS.

RESN
The reason code for the function failure. If this value is not zero, a problem has occurred.

TEXT
The address of user-supplied trace data. You can use this to help diagnose FSI problems.

3 - The **specific function call** section contains the value of the fields in the parameter lists for that function call. Table 5 on page 128 specifies which formatted sections are included for each specific type of FSI function call.

4 - The **general purpose register** section contains the contents of registers 0-15 at the time the event was traced.

5 - The **specific function call** section contains the BLOCKID and BLOCKLEN for those function-specific parameter lists that are not formatted. Table 5 on page 128 specifies which unformatted sections are included for each type of FSI function call.

Summary of FSI Trace Output

The following table is arranged by FSI function calls. It gives a summary of the sections that appear in the FSI trace output for each type of function call.

Table 5. FSI Trace Output Summary

Event id	Type of Function Call	Formatted Sections	Unformatted Sections
F54	Start FSA	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDSS (Start/Stop order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response area parameter values)	• USERDATA (User trace data) • RSV1
F54	Start device	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDSS (Start/Stop order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response area parameter values)	• USERDATA (User trace data) • RSV1
F54	Stop device	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDSS (Start/Stop order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response area parameter values)	• USERDATA (User trace data) • RSV1
F54	Stop FSA	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDSS (Start/Stop order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response area parameter values)	• USERDATA (User trace data) • RSV1

Table 5. FSI Trace Output Summary (continued)

Event id	Type of Function Call	Formatted Sections	Unformatted Sections
F54	Stop FSS	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDSS (Start/Stop order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response area parameter values)	• USERDATA (User trace data) • RSV1
F54	Intervention Order	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDIV (Intervention order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response parameter values)	• USERDATA (User trace data) • RSV1
F54	Set Order	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • ORDST (Set order parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response parameter values)	• USERDATA (User trace data) • RSV1
F54	Synch Order	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response parameter values)	• USERDATA (User trace data) • RSV1
F54	Query Order	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • ORDPARM (Common order section parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response parameter values)	• USERDATA (User trace data) • RSV1

Table 5. FSI Trace Output Summary (continued)

Event id	Type of Function Call	Formatted Sections	Unformatted Sections
F55	POST	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • POSTPARM (POST parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1
F56	GETDS	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • GDSPARM (GETDS parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZCHK • IAZJSPA	• USERDATA (User trace data) • RSV1
F57	GETREC	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • GLRPARM (GETREC parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1 • IAZIDX
F58	FREEREC	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • FLRPARM (FREEREC parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1 • IAZIDX
F59	RELDS	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • RDSPARM (RELDS parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1
F5A	CHKPT	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • CHKPARM (Checkpoint parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1
F5B	SEND	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • SNDPARM (Send parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15) • IAZRESPA (Order response area parameter values)	• USERDATA (User trace data) • RSV1

Table 5. FSI Trace Output Summary (continued)

Event id	Type of Function Call	Formatted Sections	Unformatted Sections
F5C	FSA Connect	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • CDFPARM (Connect/Disconnect parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1 • IEFJSSOB • IEFJSSIB • CDFPAIRS • IAZFSIP • GPRS
F5D	FSS Disconnect	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • CDFPARM (Connect/Disconnect parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1 • IEFJSSOB • IEFJSSIB • CDFPAIRS
F5D	FSA Disconnect	• Common Header • IAZFSIP (IAZFSIP common header parameter values) • CDFPARM (Connect/Disconnect parameter values) • FSIGPRS (Contents of general purpose registers R0 - R15)	• USERDATA (User trace data) • RSV1 • IEFJSSOB • IEFJSSIB • CDFPAIRS

Appendix A. FSIREQ Parameter List

The mapping macro for the parameter lists of all of the FSI functions is IAZFSIP. The FSS/FSA must provide storage for the parameter list when it issues FSIREQ requests. This macro adheres to the following guidelines:

- A general header precedes the function-dependent parameter lists.
- Each parameter area begins on a fullword boundary. Further status information for the specific request may be returned, depending on the service, in flag bytes of the parameter/response area. Both successful completion and failure can have more status to report.

  A non-zero return code from an FSI request always indicates an abnormal termination condition. Therefore, the FSS/FSA should abnormally terminate when it receives a non-zero return code. The specific non-zero return code values for all FSI functions depend on JES and FSS and are defined by the JES or FSS owning the FSI routine.

Note: The FSS/FSA should take a dump when it receives a non-zero return code from an FSI request.

For information about the proper location for the FSIREQ parameter list, see "Types of Orders" on page 26.

The following sections illustrate the storage maps for each section of the FSI parameter list. The section that specifically deals with the corresponding function contains the parameter descriptions.

CDFPAIRS

The following area consists of pairs of function IDs and their corresponding routine entry point addresses. The number of pairs is specified by the value of the CDFIDNO field.

0	CDFPAIRS
0	CDFID
4	CDFAD

Orders Parameter Section

The ORDER parameter list is made up of three separate sections:

- Common order header
- Variable order data section (dependent on the specified order)
- Order response area

Common Order Header

The common order header portion of the ORDER parameter list follows:

0	ORDFLGS1	RESERVED	
4	ORDFDATA		
8	ORDRSPAD		

12	ORDID	RESERVED
16	RESERVED	

START/STOP Order Data Section

0	ORDSSSP			
4	ORDSSF1	RESERVED	ORDSSMX	
8	ORDSSID			
8	ORDSSSI		ORDSSAI	
12	ORDSSAD4			
12	ORDSSAD			RESERVED
16	ORDSSNA (an 8-byte field)			
24	RESERVED			
28	ORDSSSP2			
32	RESERVED			
36	RESERVED			

Device Initialization Area for START FSA Order

0	ORDSSPF1	ORDSSPF2	ORDSSPF3	RESERVED
4	ORDSSKI			
8	ORDSSNI			

Message Routing Information Area for Start FSA Order

0	ORDSS2LN	ORDSS2FL	RESERVED
4	ORDSS2RC (a 16-byte field)		
20	ORDSS2CN		
24	RESERVED		
28	RESERVED		
32	RESERVED		

SET Order Data Section

0	ORDSTRI	RESERVED
4	ORDSTNI	
8	RESERVED	
12	RESERVED	
16	RESERVED	
20	RESERVED	

SYNCH Order Data Section

0	ORDSYR1	ORDSYR2	ORDSYR3	ORDSYR4
4	ORDSYR5	ORDSYR6	RESERVED	RESERVED
8	ORDSYNP			
12	ORDSYKI			
16	ORDSYCP		RESERVED	
20	ORDSYSMX			
24	RESERVED			
28	RESERVED			
32	RESERVED			
36	RESERVED			
40	ORDSYMSG (a 120-byte field)			

INTERVENTION Order Data Section

0	ORDIVF1	ORDIVF2	RESERVED
4	ORDIVBTT (an 8-byte field)		
12	ORDIVFLT (an 8-byte field)		
20	ORDIVFOT (an 8-byte field)		
28	ORDIVCFT (an 8-byte field)		
36	RESERVED		
40	RESERVED		
44	RESERVED		
48	RESERVED		

IAZRESPA - Order Response Data Area

0	RESPID		
4	RESPLEN		
8	RESPFL1	RESPFL2	RESERVED
12	RESPRETC		
16	RESERVED		
20	RESPCPYC		RESERVED
24	RESPPGEC		
28	RESPLREC		
32	RESPOOPI (a 12-byte field)		
44	RESERVED		
48	RESERVED		
52	RESERVED		
56	RESERVED		

GETDS Function Dependent Area

The GETDS parameter list contains the following information:

0	GDSFLGR1	GDSFLGR2	GDSFLGS1	RESERVED
4	GDSCKPL			
8	GDSCKPA			
12	GDSJSPA			
16	GDSDDTK (an 8-byte field)			
24	GDSOUTK (an 8-byte field)			
32	GDSJDVTN (an 8-byte field)			
40	GDSDSID (a 12-byte field)			
52	RESERVED			
56	GDSRECFM	GDSMRECL		RESERVED
60	RESERVED			
64	RESERVED			
	GDSSJMSG (an 80-byte field)			

GETDS Function Dependent Extension Area

0	FSIEGLEN	FSIEGVSN
4	FSIEGFID	
8	FSIEGUTK (80-byte field)	
88	FSIEGRTK (80-byte field)	
168	FSIEGOGT (20-byte field)	

IAZJSPA - JES Job Separator Page Data Area

0	JSPAID		
4	JSPALEN	JSPAFLG1	RESERVED
8	JSPAJBNM (an 8-byte field)		
16	JSPAJBID (an 8-byte field)		
24	JSPADEVN (an 8-byte field)		
32	JSPADEVA		
36	JSPAJMR		

IAZJSPA - JES Dependent Section

0	JSPAJES	
0	JSPJGRPN (an 8-byte field)	
8	JSPJGRP1	JSPJGRP2
12	JSPJGRPD (an 8-byte field)	
20	JSPJRMNO	
24	JSPJPNAM (a 20-byte field)	

44	JSPJDSNM		
	44	JSPJDSPN (an 8-byte field)	
	52	JSPJDSSN (an 8-byte field)	
	60	JSPJDSDD (an 8-byte field)	
68	JSPJSOCL	JSPJPRIO	not part of parameter list

IAZJSPA - User Dependent Section

0	JSPAUSER
0	JSPAUSR1
4	JSPAUSR2

GETREC Function Dependent Area

0	GLRFLGR1	RESERVED	GLRFLGS1	RESERVED
4	GLRINDX			
8	GLRECID (an 8-byte field)			
16	GLRDSID (a 12-byte field)			
28	RESERVED			
32	RESERVED			
36	RESERVED			
40	RESERVED			

IAZIDX - Index Returned by GETREC

Index Header Area

0	IDXID	
4	IDXNUM	IDXTOK
8	RESERVED	

Index Entry

0	IDXENTRY	
0	IDXENTRL	IDXRECL
4	IDXFLAG1	RESERVED
8	IDXRADR	
12	IDXRECID (an 8-byte field)	

FREEREC Function Dependent Area

The FREEREC parameter list contains the following information:

0	FLRINDX

4	FLRDSID (a 12-byte field)
16	RESERVED
20	RESERVED
24	RESERVED
28	RESERVED

RELDS Function Dependent Area

0	RDSFLGS1	RESERVED
4	RDSDSID (a 12-byte field)	
16	RESERVED	
20	RDSMIDSE (an 8-byte field)	
28	RESERVED	

CHKPT Function Dependent Area

0	CHKADR			
4	CHKFLGR1	RESERVED	CHKFLGS1	RESERVED
8	CHKDSID (a 12-byte field)			
20	RESERVED			
24	RESERVED			
28	RESERVED			
32	RESERVED			

IAZCHK - FSI Checkpoint Record

0	CHKID	
4	CHKLNGTH	RESERVED
8	CHKJESWK (a 64-byte field)	
72	CHKRBA (an 8-byte field)	
80	CHKDEV	
84	CHKMOD	
88	CHKCOPY	
92	CHKTRNC	
96	CHKREC	
100	CHKPAGE	
104	CHKPROD (an 8-byte field)	
112	CHKVER	
116	CHKRELS	
120	CHKMODF	
124	CHKSERV	

POST Dependent Section

0	POSTFLS1	RESERVED
4	POSFDATA	
8	RESERVED	

SEND Dependent Section

0	SNDTYPE	RESERVED
4	SNDRSPTR	
8	RESERVED	

FSIUDATA - User Trace Data Area

0	FSIUDLEN
4	FSIUDNAM (an 8-byte field)
12	FSIUDTXT (a maximum of 2000 bytes)

Appendix B. Numeric Values of FSI Services and Orders

The following chart provides the absolute values for the FSI services that the FSS/FSA specifies in the FSIFUNC field of the FSI parameter list (IAZFSIP).

Table 6. Numerical Values of FSIFUNC

FSI Service	Numerical Value
FSICON	254
FSIDCON	255
FSIGDS	3
FSIRDS	6
FSIGREC	4
FSIFREC	5
FSICKPT	7
FSISEND	8
FSIORDER	1
FSIPOST	2
SNDTYRSP	X'80'

The following chart provides the absolute values for the orders that the JES specifies in the ORDID field of the order function dependent area of the FSI parameter list (IAZFSIP).

Table 7. Numerical Values of ORDID

FSI Order	Numerical Value
ORDSPFSS	4
ORDSTFSA	8
ORDSPFSA	12
ORDSTDEV	16
ORDSPDEV	20
ORDQUERY	24
ORDSET	28
ORDSYNCH	32
ORDINTV	36

Appendix C. Accessibility

Accessible publications for this product are offered through the z/OS® Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrcfs@us.ibm.com or to the following mailing address:

IBM® Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size.

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually

exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1!

(KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Note:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This book is intended to help the customer write and install a function subsystem (FSS) and its functional subsystem application (FSA). This book primarily documents Product-sensitive Programming Interface and Associated Guidance Information provided by z/OS.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of z/OS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

However, this book also documents Diagnosis, Modification or Tuning Information, which is provided to help the customer to do diagnosis of z/OS.

Attention: Do not use this Diagnosis, Modification or Tuning Information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs by an introductory statement to a chapter or section or by the following marking:

Diagnosis, Modification or Tuning Information ....

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 (http://www.ibm.com/legal/copytrade.shtml).

Index

G

I

J

K

M

N

O

P

Q

R

S

T

U

Product Number: 5650-ZOS

Printed in USA

SA38-0678-00