z/OS
Version 2 Release 4

MVS Programming: JES Common Coupling Services

SA23-1387-40

Note

Before using this information and the product it supports, read the information in “Notices” on page 129.

This edition applies to Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2019-07-10

Contents

Figures vii

Tables ix

About This document xi
- Who Should Use This document xi
- How to Use This document xi
- Where to find more information xi

How to send your comments to IBM xiii
- If you have a technical problem xiii

Summary of changes xv
- Summary of changes for MVS and JES for Version 2 Release 3 (V2R3) xv
- Summary of changes for MVS and JES for Version 2 Release 3 (V2R3) xv
- Summary of changes for MVS and JES for Version 2 Release 2 (V2R2) xv

Chapter 1. Introduction to the JES Common Coupling Services 1
- Your Installation's Goals - Do You Need to Use JES XCF Component Services? 2
- JES benefits from the use of the common coupling services component 3
- How You Can Affect JES Message Processing 5
- Basic JES XCF Terms and Concepts 5
- Providing Security for JES XCF 6
- Macro Overview 6
- Exit Overview 7

Chapter 2. Using the JES XCF exits and macros 9
- JES and JES XCF Processing Overview 9
  - JES initialization processing 9
  - JES message processing 10
  - JES Termination Processing 11
- Using JES XCF Services 12
- The JES-Defined Attachment to JES XCF 13
  - Determining If You Should Use the JES-Defined Attachment 13
- Retrieving the JES XCF Group Token 13
- Creating an Installation-Defined Attachment to JES XCF (IXZXIXAT Macro) 14
- Building a Mailbox (IXZXIXMB Macro) 15
  - Mailbox Availability 15
  - Identifying the Installation-Written POST Exit Routine 16
  - Requesting XCF Event Monitoring 16
- Sending a Message (IXZXIXSM Macro) 16
  - Recovery levels 17
  - Sending an Asynchronous Message Without Acknowledgement or Recovery 20
  - Sending an Asynchronous Message With Acknowledgement 20
  - Sending an Asynchronous Message Without Acknowledgement 20
  - Sending a Synchronous Message 20
  - Identifying Which Mailbox is to Receive the Acknowledgement 21
  - Message Segments 21
  - Providing a Place for Data Returned from the Message Receiver 22

Identifying Which Member is to Receive the Message....22
Restrictions and IBM Recommendations....23
Receiving a Message (IXZXIXRM Macro)....23
Mapping Messages....24
Restrictions....25
Acknowledging Receipt of a Message (IXZXIXAC Macro)....25
Obtaining information about JES members in the XCF group (IXZXIXIF macro)....26
Collecting JES3 performance information data (IXZXIXPI macro)....27
Indicating that system cleanup is complete (IXZXIXCL macro)....30
Clearing a Mailbox of Messages (IXZXIXMC Macro)....30
Deleting a Mailbox (IXZXIXMD Macro)....30
System Return and Reason Codes When Messages are Undelivered....30
What Happens to Mailboxes When the JES Session Ends....31
Detaching from a JES XCF Group (IXZXIXDT Macro)....31
Component trace and diagnostic support....31
Controlling the Quantity of Traced Data....32
Obtaining Trace Data....32
Module Footprint Tracing (FLOW Sub-Level)....32
XCF and JES XCF Event Tracing (XCFEVT Sub-Level)....33
Detailed Message Tracing (MSGTRC Sub-Level)....33
Installation Exit Tracing (USRXIT Sub-Level)....33
Ending and Restarting the JESXCF Address Space....33

Chapter 3. Coding the JES XCF exits....37
Creating Your Own Attachment to JES XCF....37
Determining Your Need to Create Your Own JES XCF Attachment....37
Defining a JES XCF Group....38
Using the JES XCF exits....38
IXZXIT01 - Transport Exit....43
Installing the Exit Routine....43
Exit Routine Environment....43
Exit routine processing....43
Processing Multi-Segment Messages....46
Programming Considerations....47
Entry Specifications....47
Return Specifications....48
Coded Example of Exit Routine....49
IXZXIT02 - Receive Exit....49
Installing the Exit Routine....49
Exit Routine Environment....49
Exit routine processing....50
Programming Considerations....52
Entry Specifications....52
Return Specifications....53
Coded Example of Exit Routine....54
IXZXIT03 - Attach/Detach Exit....54
Installing the Exit Routine....54
Exit Routine Environment....54
Exit Routine Processing....55
Programming Considerations....59
Entry Specifications....59
Return Specifications....60
Coded Example of Exit Routine....61

Chapter 4. JES XCF macro reference....63
How to Read a Syntax Diagram....63
Controlling Macro Expansion Printing....65

IXZXIXAC - acknowledge receipt of a message....65
IXZXIXAT - Attach to a JES XCF Group....70
IXZXIXCL - System cleanup initiated indicator....75
IXZXIXDT - Detach from a JES XCF Group....78
IXZXIXIF - obtain information about members of an XCF group....82
IXZXIXMB - Build a mailbox....88
IXZXIXMC - Clear a Mailbox....94
IXZXIXMD - Delete a Mailbox....98
IXZXIXPI - Collect JES3 Performance Information Data....103
IXZXIXRM - Receive a message....106
IXZXIXSM - Send a Message....112

Appendix A. POST Exit Routine....121
Installing the Exit Routine....121
Exit Routine Environment....121
Exit Recovery....121
Exit Routine Processing....122
Programming Considerations....122
Entry Specifications....122
Registers at Entry....122
Parameter List Contents....122
Return Specifications....123
Registers at Exit....123
Return Code Meanings....123

Appendix B. Accessibility....125
Accessibility features....125
Consult assistive technologies....125
Keyboard navigation of the user interface....125
Dotted decimal syntax diagrams....125

Notices....129
Terms and conditions for product documentation....130
IBM Online Privacy Statement....131
Policy for unsupported hardware....131
Minimum supported hardware....132
Programming Interface Information....132
Trademarks....132

Index....133

Figures

1. A JES XCF group and its relationship to an MVS sysplex..... 2

2. JES communication mechanism..... 4

3. JES initialization processing and its relationship to the JES XCF processing..... 10

4. JES message processing and its relationship to JES XCF processing..... 11

5. JES termination processing and its relationship to JES XCF processing..... 12

6. Send message example showing levels of recovery..... 18

7. Example synchronous message request..... 21

8. JES XCF message mapping macro structure..... 25

9. Array format of data returned by IXZXIXIF..... 27

10. Example JES3 Performance Data Returned by IXZXIXPI..... 29

11. JES XCF trace structure..... 32

12. Example of using JES XCF macros and exits for an ASYNCACK message cycle..... 39

13. Access path from exit caller to message data..... 44

14. IXZ$XPL mapping of message data and extents..... 46

15. Access path from exit caller to message data..... 51

16. Example attach (IXZXIT03) code..... 56

17. Exit IXZXIT03 attach/detach processing..... 58

Tables

1. JES XCF Tasks Based on Your Need to Modify JES XCF Communication..... 3

2. JES Common Coupling Services Macros.....6

3. JES Common Coupling Services Exits.....7

4. JES Control Block from Which to Retrieve the JES XCF Group Token..... 14

5. Request Type Characteristics for Send Message Macro..... 17

6. Required Parameter Token Definitions for SEGTYPE and REQTYPE..... 22

7. System Return and Reason Codes.....31

8. Environment..... 65

9. Environment..... 70

10. Environment..... 75

11. Environment..... 78

12. Environment..... 82

13. Environment..... 89

14. Environment..... 94

15. Environment..... 99

16. Environment.....103

17. Environment.....107

18. Environment.....113

About This document

This document provides MVS base control program (BCP) and job entry subsystem (JES) system programmers with guidance and reference information about the macros and exits that allow an installation to customize its use of the JES common coupling services component of MVS. JES members in an MVS sysplex environment can use the macros to send and receive messages, and can use the exits to monitor or tailor messages and status.

JES2 uses the common coupling component for checkpoint reconfiguration processing, and JES3 uses the component as a replacement for its multi-system communication mechanism.

Who Should Use This document

This document is intended for MVS BCP, JES2, or JES3 system programmers or for anyone responsible for tailoring an installation's use of the JES common coupling services (JES XCF) component.

If you modify JES2 and JES3 processing through JES XCF, make those changes in JES standard exits and JES3 DSPs using assembler language programming. Assembler language programming is described in the following documents:

- *HLASM Language Reference*, SC26-4940
- *HLASM Programmer's Guide*, SC26-4941

How to Use This document

This document provides the information you need to:

- Use the macro interface to:
  - Attach and detach a JES common coupling services (JES XCF) group
  - Create, clear, and delete mailboxes
  - Send, receive, and acknowledge receipt of messages
  - Obtain JES2 multi-access spool configuration and JES3 complex member information.
- Use the exits to:
  - Attach/detach a JES XCF group
  - Allocate and initialize an installation data area
  - View, modify, or reroute a message prior to sending
  - View or modify a message prior to receiving.

Where to find more information

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

How to send your comments to IBM

We invite you to submit comments about the z/OS® product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xiii.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS MVS JES Common Coupling Services, SA23-1387-40
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority 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

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for MVS and JES for Version 2 Release 3 (V2R3)

This information contains no technical changes for this release.

Summary of changes for MVS and JES for Version 2 Release 3 (V2R3)

This information contains no technical changes for this release.

Summary of changes for MVS and JES for Version 2 Release 2 (V2R2)

This information contains no technical changes for this release.

Chapter 1. Introduction to the JES Common Coupling Services

The MVS **cross-system coupling facility** (XCF) allows up to 32 MVS systems to communicate in a single-image environment. That is, although there are from 1 to 32 MVS systems operating, the communications and control of those systems can be accomplished as if the individual systems were a single system. This single-image environment is an MVS **sysplex**. Within the MVS sysplex, you can define one or more JES XCF groups. A **JES XCF group** is a set of JES subsystems that communicate using the **JES common coupling services** component (JES XCF). **JES XCF** uses XCF as its communication vehicle. In a JES2 environment, the JES XCF group is a multi-access spool configuration (MAS), and in a JES3 environment, the JES XCF group is a JES3 complex.

The JES common coupling services component provides macros that enable communication among JES members of a sysplex. The macros are available to either JES2 or JES3. JES2 uses the service for communication of checkpoint data set information and during checkpoint reconfiguration processing. JES3 uses the JES common coupling service component as a replacement for its channel-to-channel (CTC) communication among JES3 mains. In addition to the macros, the JES common coupling services component also provides exits that can be used to monitor or tailor messages sent among the JESes; these exits are also available and used by JES2 and JES3.

Refer to *z/OS MVS Setting Up a Sysplex* for an in-depth discussion of the sysplex and its management.

The JES common coupling services component:

- Provides a communication mechanism to JES members (JES2 multi-access spool members or JES3 mains) to exchange data and communicate by exploiting the MVS XCF component.
- Enables a JES2 exit routine, running in three of the four JES execution environments (JES2 maintask, JES2 subtask, or user) to send messages to a JES2 exit in another processor control element (PCE) running on the same JES2 or on a different JES2.
- Enables a JES3 exit routine, running in any of the four JES execution environments (JES3 maintask, JES3 subtask, user, or FSS) to send messages to a dynamic support program (DSP) or function control table (FCT) running on the same JES3 or on a different JES3.
- Enables a JES3 DSP to send messages to a DSP or FCT running on the same JES3 or on a different JES3.
- Enables a **JES dispatchable unit** (JDU) to request that it be notified about XCF-monitored system events. **JDU** is a generic term for JES2 PCEs and JES3 DSPs and FCTs.

As each JES member is initialized, it automatically joins a JES XCF group. After all JES members have successfully joined an XCF group, JES exit routines or JES3 DSPs running on that JES member can use the JES common coupling services to communicate with the other JES members in the same JES XCF group.

IBM recommends that your JES subsystems map in a one-to-one relationship with the MVS sysplex members. That is, if you have 9 MVS systems comprising your sysplex, then each of the 9 primary JES subsystems should then form a JES XCF group as in Figure 1 on page 2. Configuring your JES subsystems and your MVS systems in this way will ease your overall sysplex management task.

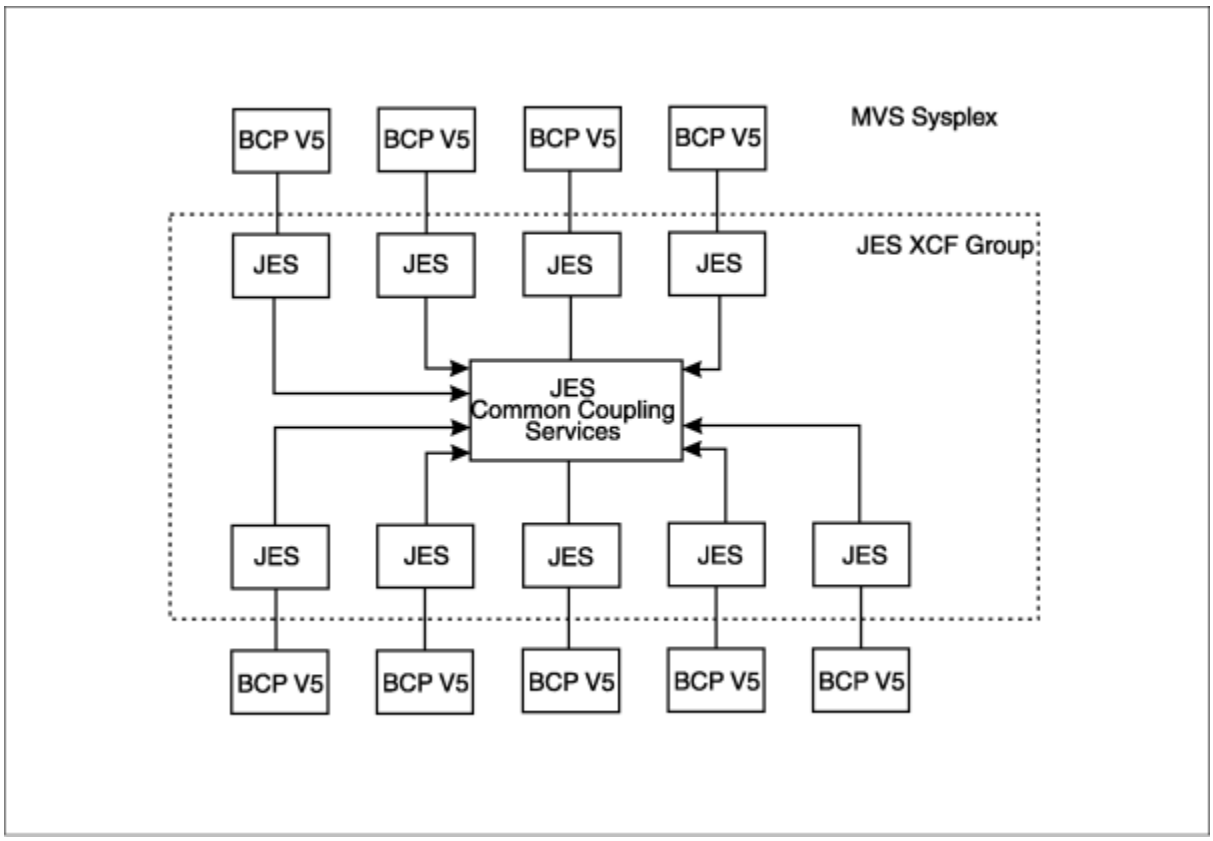


Figure 1. A JES XCF group and its relationship to an MVS sysplex

There are a number of restrictions that apply to JES XCF groups and their use. These include:

- The JES common coupling services component and its exits and macros are **not** available outside either a JES2 or JES3 environment.
- The JES systems in the JES XCF group must be either all JES2 or all JES3; do not mix JES2 and JES3 systems.
- All JES2 multi-access spool (MAS) members or JES3 complex mains must join JES XCF if they are to communicate.
- All JES2 MAS members or JES3 mains must be members of the same sysplex.
- All participating JES members must be in the same JES XCF group.

Your Installation's Goals - Do You Need to Use JES XCF Component Services?

As noted above, as JES2 MAS members or JES3 complex members initialize, each automatically attaches to the default JES XCF group. JES uses the JES XCF component as its member-to-member communication mechanism. Therefore, if you do not need to monitor or modify JES message processing or use the JES XCF services, you do not need to use the JES XCF macros and exits described in this document. If, however, you do need to monitor or modify JES message processing or define your own JES XCF groups, you should become familiar with the JES XCF macros, the services each provides, and the JES XCF exits. Table 1 on page 3 provides an overview of the tasks you will need to accomplish based on your installation's need to affect JES XCF processing.

Table 1. JES XCF Tasks Based on Your Need to Modify JES XCF Communication	
If your installation code:	**Do the following:**
Does **not need** to monitor JES messaging Does **not need** to modify JES messaging Will **not use** JES XCF to send data	Nothing. Your installation will realize the immediate benefits of JES XCF that are listed in "JES benefits from the use of the common coupling services component" on page 3.
Sends and receives messages within standard JES exits or JES3 DSPs	1. Obtain the JES XCF group token 2. Use JES XCF macros to send and/or receive messages in standard exits or JES3 DSPs
Monitor or modify JES messages	1. Code JES XCF exit IXZXIT03 to obtain storage for JES XCF transport exit IXZXIT01 and receive exit IXZXIT02 2. Use JES XCF exits IXZXIT01 and IXZXIT02 to monitor or modify JES messages

Note: If necessary, your installation can create its own JES XCF group and provide JES2 MAS or JES3 complex member attachments to it. Use this document to provide the information you require to create your own attachments.

JES benefits from the use of the common coupling services component

Both JES2 and JES3 use the JES common coupling services component and benefit from:

- Access to a common time reference available through the Sysplex Timer
- Use of a high-speed messaging service
- Monitoring system and address space events and status.

Further, JES2 uses the component's macros and exits for checkpoint reconfiguration processing to simplify what would otherwise require a complex, operator-intensive dialog. (The component provides a communication mechanism that informs each JES2 MAS member of system events such as system status, and checkpoint data set and structure status.) JES3 uses the component to allow communication across the JES3 complex that would otherwise require JES3-managed channel-to-channel (CTC) hardware. Additionally, your installation can use the component's macros and exits to satisfy your processing needs in JES installation exits and modifications. Figure 2 on page 4 shows a simplistic view of a JES member-to-member link as provided by the XCF and the JES XCF.

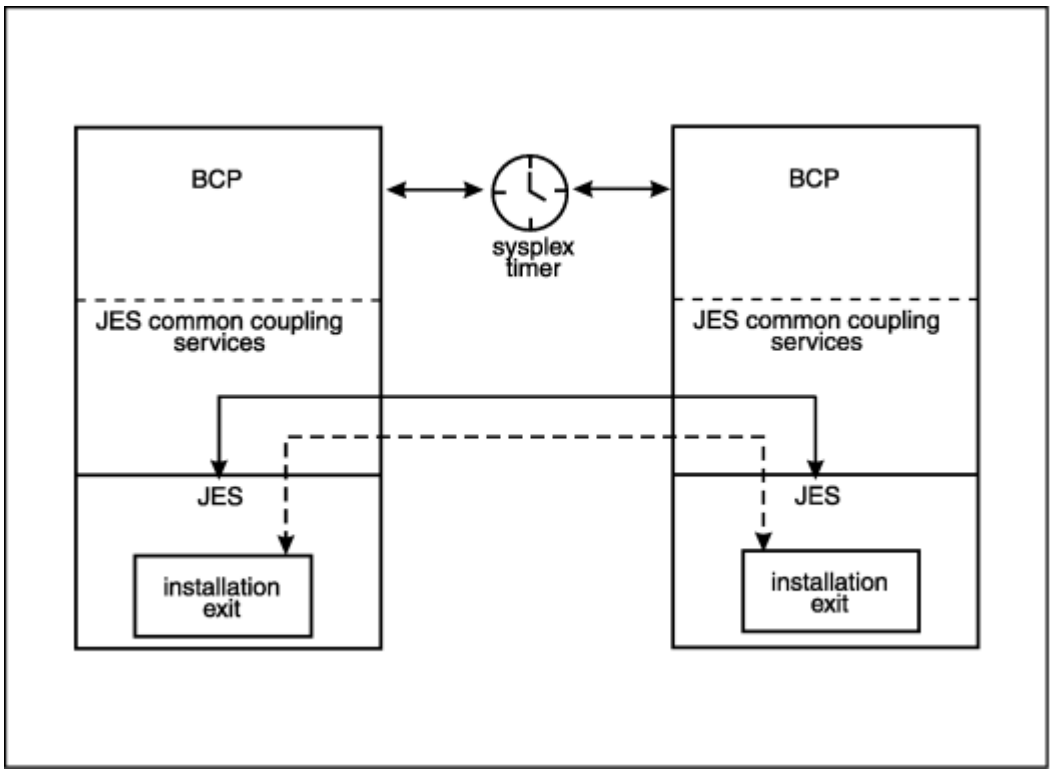


Figure 2. JES communication mechanism

There are several ways your installation might chose to use the JES XCF macros and exits. When each JES subsystem is a member of a JES XCF group, you can use the JES XCF macros to send and receive messages among the JES members. Thus, your installation is capable of creating multi-system modifications which increases the quality of your current or future installation modifications.

To ease diagnosis and future migration tasks, IBM recommends that you invoke the JES XCF macros within standard JES exits and JES3 DSPs. JES XCF allows you to:

- **Add to or modify JES component messages**

  You can use exits IXZXIT01, IXZXIT02, and IXZXIT03 to modify JES processing by changing the data that JES sends from one JDU to another. You can affect JES functions and thereby modify JES to meet more of your installation's processing requirements through IBM-supplied exits.
- **Write a JES3 DSP**

  You cannot write a JES3 DSP to provide function such as tape management unless your installation has the ability to modify or create function at the same level that IBM JES3 code requires. The JES common coupling services provide an interface to allow you to write such code.
- **Use ESCON architecture**

  In JES3, you can eliminate the use of JES3-managed 3088 channel-to-channel (CTC) hardware and replace it with MVS XCF-managed enterprise systems connection (ESCON) architecture hardware. Depending upon your specific application, you can use either CTCs or the coupling facility. Using the coupling facility provides increased ease of system management through more simplified system-to-system connectivity. The coupling facility eliminates the exponential growth of CTC connections associated with the addition of each new system to your sysplex. Refer to your ESCON hardware documentation for further information concerning the choices of available hardware and its use in a JES3 complex.

How You Can Affect JES Message Processing

Depending on your installation's needs, you can use the JES common coupling services component to affect JES message processing through several methods. Use JES XCF macros, JES XCF exits, or standard JES exits and JES3 DSPs to affect message processing to meet your installation's needs. The services provide the following options:

- You can send and receive installation messages using the JES common coupling services macros. These macros can be invoked from standard JES2 and JES3 installation exits and from JES3 DSPs. Refer to *z/OS JES2 Installation Exits* or *z/OS JES3 Customization* for JES2 and JES3 exit information and coding requirements, respectively.
- You can view and modify all messages handled by JES XCF using the transport exit and the receive exit. These exits are invoked by JES XCF to process both system messages and installation-defined messages.

The use of these services is optional. Depending on your installation requirements you can:

- Use neither the macros nor the installation exits provided by JES XCF.
- Use the macros provided by JES XCF to send and receive installation messages in standard JES exits and JES3 DSPs.
- View and monitor IBM messages using the transport and receive installation exits. This does not require your use of the macros. Refer to Chapter 3, “Coding the JES XCF exits,” on page 37 for a description of the exits.
- Use the JES XCF macros to send and receive installation messages from standard JES exits and JES3 DSPs, and use the JES XCF exits to view or modify all messages handled by JES XCF.

Basic JES XCF Terms and Concepts

Before discussing the specifics of how you can affect JES message processing, you need a basic understanding of the terminology used in this document and of the underlying concepts of JES XCF.

In general, the JES common coupling services allow JES members to communicate, but they also allow you to affect that communication. When each JES member of either your JES2 MAS or JES3 complex initialize, it is automatically **attached** to the **JES XCF group**. The members of the group will be the same set of members in your MAS or complex. When using the macros, you need to use the **group token** that JES XCF provides during attach processing to identify the member and group. The group token is the unique identifier for a particular member of a group.

The term **message** refers to data that is transported between members of the JES XCF group. Messages also include XCF events. An XCF event includes any XCF events on any JES member of the JES XCF group and the MVS system under which it runs. The messages are transported by XCF and JES XCF from one JDU to another. If the quantity of data being sent is larger than 60 kilobytes, you have the ability to break the entire set of data into parts and transport it as a **multi-segment** message.

Acknowledgements are required by JES XCF for every message that is received. This is required so that JES XCF knows when it can free whatever resources it held for that message. Also, the message sender can request that an acknowledgement be returned by the receiver. The sender can then base further processing on that acknowledgement. This can be valuable if you chose to send messages in an asynchronous manner such that the acknowledgement is used to confirm the future delivery or reception of the message at the destination member without causing the sending member to *wait* for the response.

Messages are packaged in an **envelope** that contains the source and destination address as well as other data concerning the message data. The entire package is then sent to and placed in the designated mailbox. A **mailbox** is a named, logical queue of ordered messages that is maintained by JES XCF to hold messages that have been sent to a JDU but have not yet been received by that JDU. Should the message not be delivered or received as expected, the original sender will receive return and reason codes from several sources, including JES XCF and the receiver.

Chapter 2, “Using the JES XCF exits and macros,” on page 9 details the use of the JES XCF exits and macros to performs specific tasks, and Chapter 3, “Coding the JES XCF exits,” on page 37 details the use of the JES XCF exits to monitor and modify JES XCF message processing.

Providing Security for JES XCF

The JESXCF address space accesses system resources as part of its normal operation. In order to satisfy security requirements when accessing these resources, an installation must define the JESXCF address space as trusted. The member name/procedure name and jobname associated with the JESXCF address space is *JESXCF*.

To properly define JES XCF using the RACF started procedures table ICHRIN03 (at RACF 1.9.2 or lower), add an entry such as the following to your started procedures table.

```
        DC    CL8'JESXCF'     PROCEDURE NAME
        DC    CL8'userid'     USERID
        DC    CL8'grpname'    GROUP NAME
        DC    XL1'40'         ATTRIBUTE BYTE
        DC    XL7'00'         RESERVED
```

Beginning with RACF 2.1, you can create a profile in the STARTED class with member name JESXCF and jobname JESXCF. Be certain to specify the TRUSTED(YES) keyword. Refer to *z/OS Security Server RACF Security Administrator's Guide* for further information on defining a component to the RACF started procedures table or using the STARTED class.

Macro Overview

Table 2 on page 6 lists the set of executable macros that JES XCF provides. Each is fully discussed in Chapter 4, “JES XCF macro reference,” on page 63.

Table 2. JES Common Coupling Services Macros

Macro	Used In JES XCF Exit	Used In Standard JES Exit
“IXZXIXAC - acknowledge receipt of a message” on page 65		X
“IXZXIXAT - Attach to a JES XCF Group” on page 70	X	
“IXZXIXMB - Build a mailbox” on page 88		X
“IXZXIXMC - Clear a Mailbox” on page 94		X
“IXZXIXPI - Collect JES3 Performance Information Data” on page 103		X
“IXZXIXMD - Delete a Mailbox” on page 98		X
“IXZXIXDT - Detach from a JES XCF Group” on page 78	X	
“IXZXIXIF - obtain information about members of an XCF group” on page 82		X
“IXZXIXRM - Receive a message” on page 106		X
“IXZXIXSM - Send a Message” on page 112		X

Table 2. JES Common Coupling Services Macros (continued)		
Macro	**Used In JES XCF Exit**	**Used In Standard JES Exit**
"IXZXIXCL - System cleanup initiated indicator" on page 75		X

Exit Overview

Table 3 on page 7 lists the IBM-defined exits that JES XCF provides. Each is fully discussed in Chapter 3, "Coding the JES XCF exits," on page 37. Also, a sample of each exit is provided in SYS1.SAMPLIB, members IXZEX01A, IXZEX02A, and IXZEX03A, respectively.

Table 3. JES Common Coupling Services Exits	
Exit	**Purpose**
"IXZXIT01 - Transport Exit" on page 43	View, modify, reroute a message or acknowledgement before the message arrives at the receiving member's mailbox.
"IXZXIT02 - Receive Exit" on page 49	View or modify a message before it is retrieved from a mailbox.
"IXZXIT03 - Attach/Detach Exit" on page 54	• Obtain a storage area during JES attach processing for use by exits IXZXIT01 and IXZXIT02, or free that storage during JES detach processing. • Attach to or detach from an installation-defined JES XCF group.

Chapter 2. Using the JES XCF exits and macros

This chapter provides a task-oriented approach to using the JES common coupling services to tailor the IBM-supplied services or create your own coupling processing. This chapter provides:

- An overview of JES and JES XCF processing
- An overview of available JES XCF exits and macros
- An explanation of which JES environments can use the macros
- The purpose of each exit and macro
- An overview of component trace and diagnostic support.

The macros and the services they provide are intended for use within JES. Throughout this chapter and those following are recommendations to use specific macros in specific exits. The macros, can be placed directly within JES source code; however, IBM does not intend such use.

Refer to Chapter 3, “Coding the JES XCF exits,” on page 37 and Chapter 4, “JES XCF macro reference,” on page 63 for detailed information about coding the exits and macros.

JES and JES XCF Processing Overview

Before learning how you can use the JES XCF exits and macros, it's important to understand how and when JES XCF relates to standard JES processing. Your use of the macros and exits will be based not only on what function they can provide your installation, but also whether they are available when you need them to affect JES communications.

As noted in “How You Can Affect JES Message Processing” on page 5, you can:

- Use the macros in standard JES exits and JES3 DSPs
- Use the attach/detach exit to attach or detach an installation member
- Use the transport and receive exits to change the messages JES routes among its members.

To explain the interrelationship of JES and JES XCF processing, refer to “JES initialization processing” on page 9, “JES message processing” on page 10, and “JES Termination Processing” on page 11. Much of JES2 and JES3 processing is similar but, as appropriate, differences are noted in the following discussion of initialization, message, and termination processing.

JES initialization processing

During JES initialization, the JES XCF attach/detach exit receives control under the JES main task. At this time, the JES member (JES2 MAS member or JES3 complex member) is attached automatically to the JES XCF group. Each member has a different name, but each is attached to the same group to form the JES XCF group, thereby allowing each to communicate (send messages, obtain member and JES XCF group status, and so on).

IBM recommends that if you attach your own JES XCF group, you do that attach using the JES XCF attach macro, IXZXIXAT, within the attach/detach exit, IXZXIT03. In so doing, you must be careful to not inadvertently cause JES XCF to enter into a recursive set of calls caused by the attach service calling the attach exit. “IXZXIT03 - Attach/Detach Exit” on page 54 offers a more complete description of JES XCF processing and Figure 16 on page 56 provides some sample exit IXZXIT03 code to circumvent potential problems when issuing IXZXIXAT to either acquire a data area or attach an installation-defined group.

Refer to Figure 3 on page 10 for a graphic view of how IXZXIXAT processing relates to overall JES processing.

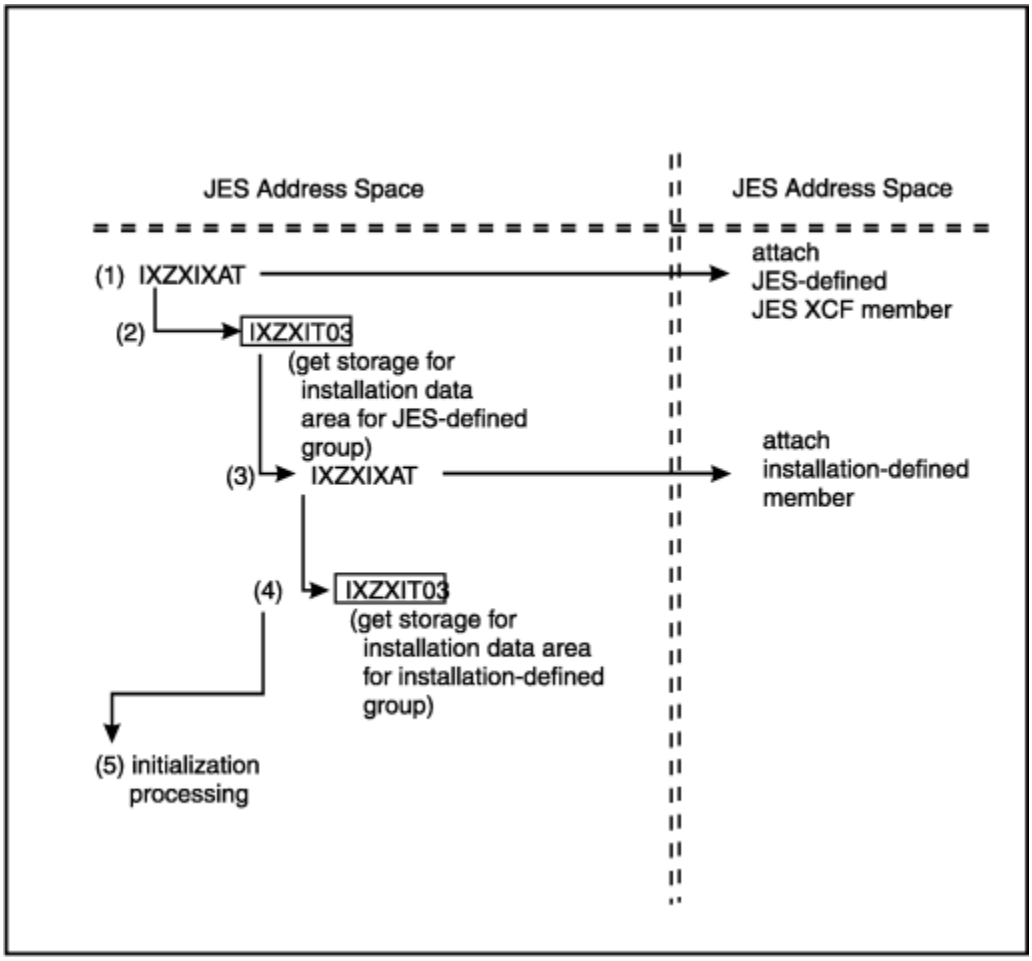


Figure 3. JES initialization processing and its relationship to the JES XCF processing

Notice that:

1. JES issues the attach macro, IXZXIXAT, during initialization processing
2. Invocation of the IXZXIXAT macro causes IXZXIT03 to be invoked to get the needed storage for IXZXIT01 and IXZXIT02 processing
3. If coded in exit IXZXIT03, the IXZXIXAT macro attaches your installation-defined group
4. If coded, exit IXZXIT03 gets the storage needed for use of exits IXZXIT01 and IXZXIT02 by members of the installation-defined group.
5. JES initialization continues.

If you require multiple attachments to the JES XCF group, or you determine the need to create an independent, installation-defined JES XCF group, you should add that code to exit IXZXIT03 prior to JES initialization so that those attachments or groups are built during initialization.

If you provide code in IXZXIT03 for attach processing, it may include macro IXZXIXAT. You cannot use any other of the JES common coupling service macros in this exit except IXZXIXDT (if you use IXZXIT03 for detach processing). (Mapping macro IXZ$XPL contains a field that indicates if IXZXIT03 is called for attach or detach processing.)

JES message processing

Each JES member is in constant communication with every other JES member of its JES2 MAS or JES3 complex. In JES2, the checkpoint data set is being continuously updated for job and output data. In JES3, there is a continuous flow of allocation requests, system status, job status, work requests, and so forth that is needed to properly coordinate the work JES3 is putting into and taking out of the MVS system.

These pieces of communicated data are considered messages, and each causes the invocation of two JES XCF exits: the transport exit and the receive exit.

Exit IXZXIT01, called the transport exit, gets control whenever JES or installation code sends a message. JES XCF packages the message data in an envelope that contains the address information and other related data, and sends it to a specified mailbox.

Refer to Figure 4 on page 11 for a graphic view of message processing.

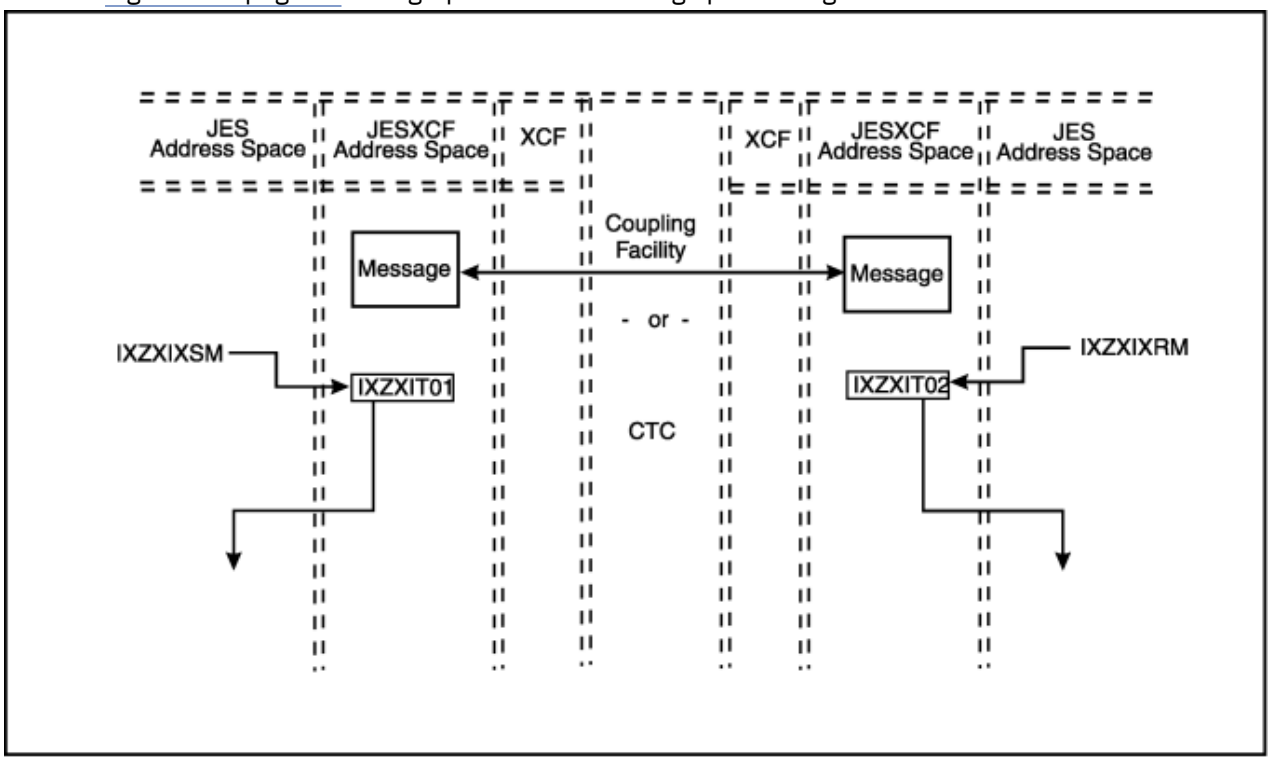


Figure 4. JES message processing and its relationship to JES XCF processing

You have the option of adding code to exit IXZXIT01 to alter the JES message processing. You can change the message text, add to the message text, or reroute the message to another member. This ability can be an extremely powerful tool that you can use to monitor member activity.

In addition, you can use standard JES exits and JES3 DSPs to invoke the JES XCF macros to build, clear, and delete mailboxes; send, receive, and acknowledge messages; and request member information during normal JES processing.

Once a message is sent to a mailbox (on the same member or another), the receiving member is informed that mail is waiting. After the receiving member issues the receive macro, IXZXIXRM, the receive message exit (IXZXIT02) gets control. This exit allows the receiving member to view or modify the message and to receive any extents that have been added to it by the transport exit. Once the message is received, the receiving member acknowledges the mail, informs JES XCF of its receipt and, if requested by the sending member, also returns an acknowledgement to the sender.

JES Termination Processing

During detach processing, exit IXZXIT03 receives control to perform clean-up processing for the terminating member of the JES XCF group. Therefore, this is the only opportunity you have to detach an installation member that you previously attached during initialization processing. **IBM recommends that you do not use the detach macro outside exit IXZXIT03.**

Detach processing

JES2 and JES3 detach processing are quite different, as shown in Figure 5 on page 12. During JES termination processing, only JES2 ($P JES2) and JES3 FSS provide clean up when terminating normally.

JES2 abnormal termination (abend) and JES3 termination (*RETURN) are equivalent;, neither invokes the detach macro.

Therefore, for JES2 and JES3 FSS normal termination only, JES issues the detach macro, IXZXIXDT. IXZXIT03 receives control as a result of the IXZXIXDT macro call and should first free storage obtained for exit IXZXIT01 and IXZXIT02 processing. If you previously used IXZXIT03 to call IXZXIXAT, you should issue the IXZXIXDT macro. (Mapping macro IXZ$XPL contains a field that indicates if IXZXIT03 is called for attach or detach processing.) JES2 then continues termination and clean-up processing. As discussed, the same requirement is not relevant to JES3 except for JES3 FSS address space calls.

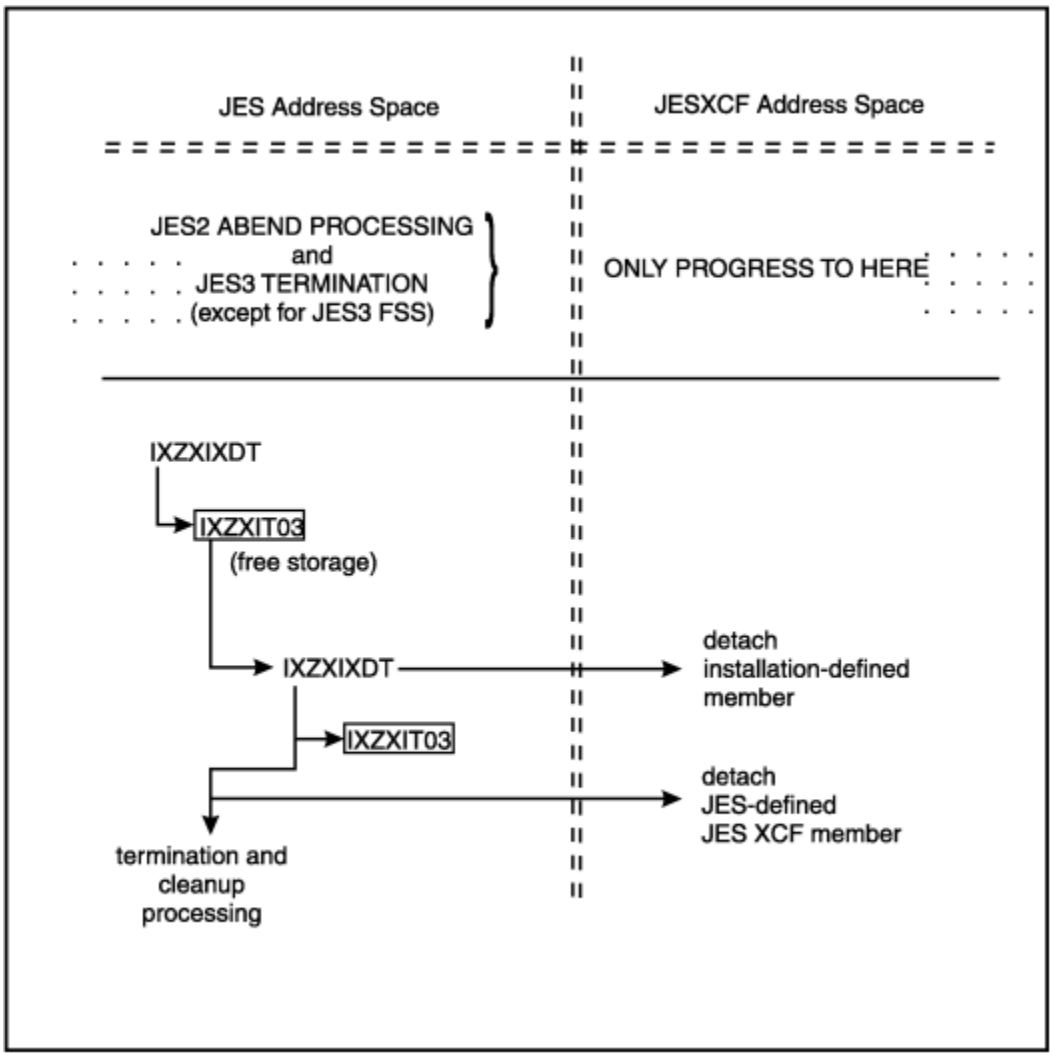


Figure 5. JES termination processing and its relationship to JES XCF processing

Using JES XCF Services

The following provides some recommendations and considerations when you invoke the JES XCF services through standard JES exits or JES3 DSPs. But first, a word of caution:

Attention:

If you choose to use the macros in JES exits or JES3 DSPs, or independent of any JES exit or JES3 DSP, you can do so, although their use outside standard exits is not recommended. Defining exits and writing installation exit routines is intended to be accomplished by experienced system programmers; the reader is assumed to have knowledge of JES2, JES3, or both.

If you want to customize your JES system, IBM recommends that you use the JES XCF installation exits and standard JES exits and JES3 DSPs to accomplish the task. **IBM does not recommend or support alteration to JES source code.**

You can use exit IXZXIT03 for two separate purposes:

- To maintain a separate attachment to the JES common coupling service.

  You don't necessarily need to use IXZXIT03 for this purpose unless you determine that your installation requires a separate attachment to JES XCF that is independent of the JES-defined attachment. Refer to "Determining If You Should Use the JES-Defined Attachment" on page 13 for a further discussion of your potential need. All JES common coupling services are available using the JES-defined attachment.
- To provide storage for use by, and shared by, both exit IXZXIT01 and IXZXIT02 processing. (You only need to obtain this storage if you intend to alter JES message processing.)

The JES-Defined Attachment to JES XCF

During initialization, JES automatically attaches to XCF to establish itself as a member of the JES XCF group. If you plan to use the JES XCF macros and exits, IBM recommends you first try to use the JES attachment instead of creating your own attachment. To do so, use exit IXZXIT01 to get control when JES or your installation sends message data; use IXZXIT02 to get control when JES or your installation receives message data, and code IXZXIT03 only to obtain a storage area to be shared by exits IXZXIT01 and IXZXIT02.

Determining If You Should Use the JES-Defined Attachment

You can use the JES-defined attachment providing there is not an adverse impact on JES performance. JES requires high-speed delivery of any messages that it sends. If you use the attachment to send large volumes of data (messages) from one member to another, you can expect an impact on JES performance.

Consider the following analogy: Years back, you decided to operate your small, single-agent real estate business from your home. Your spouse was away during the day at work. On a lean budget, you couldn't justify a second phone line although telephone calls were crucial to your business. Rarely did personal phone calls interrupt your use of the phone, in any case. As business increased, your spouse began working only part time, and the children grew. Inevitably, your family's increased use and your decreased access to the telephone began impacting your business; a second phone line became necessary. The second line, of course, accessed the same company you were using and provided all the same services. Two connections to the same set of services allowed all the required communication your household needed, without disrupting your business or the family's social environment.

Your need for a second *attachment* to JES XCF is very similar. The best way to start is to use the JES attachment. That is, **you need not code exit IXZXIT03 to attach** to the existing JES group to use the attachment. The JES XCF group is created automatically by JES in JES Version 5 and is available for your use. Once you are using the attachment, monitor JES performance. If JES performance is adversely impacted, code exit IXZXIT03 to create your own attachment to the JES common coupling service. To use the JES-defined attachment, you need to retrieve the JES XCF group token that identifies the JES XCF group and member name.

[Programming Interface Information] Retrieving the JES XCF Group Token

To use the JES-defined attachment, you must retrieve and use the JES XCF group token. It identifies both the JES member and the JES XCF group name.

Most of the JES XCF macros require either the JES-defined group's group token or the installation-defined group's group token as input. If you intend to use the JES-defined attachment, obtain the group token from the control block fields specified in Table 4 on page 14. If, however, you intend to establish your own attachment to XCF, use the group token returned by macro IXZXIXAT as described in "Creating an Installation-Defined Attachment to JES XCF (IXZXIXAT Macro)" on page 14. Depending upon the JES under which you are running, retrieve the JES XCF token from the appropriate field as follows:

Table 4. JES Control Block from Which to Retrieve the JES XCF Group Token	
JES	**Field Name in Data Area**
JES2	• $XCFIXVT in $HCT • CCTIXVT in $HCCT
JES3	• SVTJXGT in IATYSVT • TVTXJXGT in IATYTVTX
JES3 FSS	• FSCBJXGT in IATYFSCB

If you use the IXZXIXAT macro to attach your own JES XCF group, IXZXIXAT returns a group token in the field you specified with the GROUPTOKEN= keyword on the IXZXIXAT macro call. Simply use the group token as input to other JES XCF macro calls.

After retrieving the token, make it available to your other programs that will use the JES XCF macros. When your other programs subsequently use the JES XCF macros, they must provide the group token as input. However, if you detach an installation-defined member from JES XCF, be certain to clear the group token field in your program; it will no longer be valid.

[End Programming Interface Information]

Creating an Installation-Defined Attachment to JES XCF (IXZXIXAT Macro)

Whether you attach a new member to the JES-defined group or to an entirely different (installation-defined) group depends upon your XCF configuration. That is, consider the number of signalling paths and the additional message traffic you anticipate flowing across those paths. For JES-specific recommendations, refer to either *z/OS JES2 Initialization and Tuning Guide* or *z/OS JES3 Initialization and Tuning Guide*.

To create an installation-defined attachment to JES XCF, issue macro IXZXIXAT. IBM recommends that you issue IXZXIXAT from exit routine IXZXIT03. For a description of IXZXIT03, see "IXZXIT03 - Attach/Detach Exit" on page 54. Only programs executing in the JES2 or JES3 main task environment or in the JES2 or JES3 subtask environment can issue IXZXIXAT.

When you issue IXZXIXAT, you must:

- Identify the JES, JES2 or JES3, under which your program is running.
- Provide the FMID of the JES release. (Function modification identifier (FMID) is the release-specific product identifier used to identify an IBM product.)
- Specify the name of the JES XCF group that your member is to join.
- Specify the JES XCF member name by which your member will be known within the JES XCF group.
- Provide a location where the system can return the group token for the JES XCF group.

When the system returns control it provides:

1. A return code that indicates whether it completed its function successfully.
2. The name of a default mailbox to collect system event data.

   Initially, JES XCF supplies a default name for a mailbox that collects system event data. (System event data includes any XCF events on any JES member of the JES XCF group and events on the MVS system under which it runs.) You must build this mailbox, named SYSJES$DEFAULT, at the time a member attaches to a group. The default mailbox will not collect system event data until the IXZXIXMB macro has been issued to build the mailbox. IBM recommends that you continue to use this mailbox for the collection of system event data after attach processing completes and that you build a separate mailbox to collect only messages and acknowledgements sent from JDU to JDU. Then use the

appropriate mailbox name for future member communications. If you neither require the default mailbox nor intend to use it after attach processing completes, you must still build it (using the IXZXIXMB macro) and then delete it (using the IXZXIXMD macro). Be aware that you must process and acknowledge any messages sent to the default mailbox. You cannot ignore them. If you do so, JES XCF will not be notified to clean up resources held for each message sent to it. This will eventually cause an 'out of buffer' condition.

IBM recommends keeping the JES-defined attachment active for the life of the JES address space.

Each additional attachment requires its own resources (such as storage) within the JES XCF address space. Initially be cautious and be certain to closely monitor each additional attachment for potential system performance degradation.

Building a Mailbox (IXZXIXMB Macro)

Any JES dispatchable unit (JDU) that intends to receive messages must build a mailbox. A mailbox holds messages that have been sent to a JDU but have not yet been processed by the JDU. A mailbox is simply a logical queue of ordered messages that is maintained by JES XCF. When a JDU retrieves and acknowledges a message, JES XCF removes that message from the mailbox.

When building a mailbox, you must identify a POST exit routine, which you previously coded, to receive control each time JES XCF places a message in the mailbox.

To build a mailbox, issue the IXZXIXMB macro. When you issue this macro, you must:

- Provide the group token. This can either be the JES-defined group token that was obtained from the specific JES data area as noted in "Retrieving the JES XCF Group Token" on page 13, or your installation attach-defined group token that you obtained from JES XCF on return from the IXZXIXAT call.
- Provide the address of an installation exit routine that receives control when JES XCF places a message in the mailbox. (This routine is hereafter referred to as the **POST exit routine**.)
- Provide a pointer to data that the POST exit routine needs to access when it receives control.
- Provide the name that you want assigned to the mailbox.

You must ensure that programs that issue macros that reference a mailbox have access to the mailbox name. The following macros reference mailboxes:

- IXZXIXMC (mailbox clear)
- IXZXIXMD (mailbox delete)
- IXZXIXRM (receive message)
- IXZXIXSM (send message).

Mailbox Availability

Once a mailbox is built, that mailbox is available to receive mail until one of the following events occur:

- The system on which the JES member resides is re-IPLed.

  After re-IPLing, the member should rebuild its mailbox. JES XCF resends all messages sent by **other** members, except those of REQTYPE=COMM, that were in the mailbox at the time the system was shut down. Your JDU must access that mailbox to determine if these messages should be received and processed or deleted. JES XCF sets the RESENT_DUE_TO_IPL bit in IXZYIXEN (the message envelope) for each message that was in the mailbox prior to termination. Consider the age and interim system events when deciding the disposition of such "resent" message data.
- A member associated with the mailbox issues the IXZXIXDT macro to become an inactive member of the related XCF group.

  Messages sent to a member that has been detached cannot be received by that member until it becomes a member of the XCF group again. Any messages in the mailbox at the time of the detach are

acknowledged by JES XCF with a return and reason code indicating the message is acknowledged because the receiver has detached from the XCF group.

- The IXZXIXMD macro is issued to delete the mailbox.

If, after you build a mailbox, JES terminates with a termination that does not result in a detach (that is, in JES2, a $P JES2 ABEND; or in JES3, a *RETURN), the mailbox is still available to receive messages. JES code running in another address space or on another JES can continue to send messages to the mailbox. When JES restarts, the mailbox will contain messages that were sent to the mailbox while JES was down, as well as messages that were in the mailbox when JES terminated. JES XCF sets the MESSAGE_RESIDUAL bit in IXZYIXEN (the message envelope) for each message that was in the mailbox prior to termination. Use this indicator to determine if the message is "old" or "new".

The routine that retrieves messages from the mailbox should then decide whether to clear messages from the mailbox or receive and process them.

Identifying the Installation-Written POST Exit Routine

When you build a mailbox using IXZXIXMB, you must identify a POST exit routine that is to receive control when JES XCF places a message in that mailbox. You must:

- Provide the exit routine (POSTXIT= parameter on the IXZXIXMB macro) and specify its entry point.
- Provide a data area that contains data required by the exit routine. If the data area is in a data space or in another address space, you must also provide the access list entry token (ALET) used to access the data. The ALET must be on the primary address space access list (PASN-AL).

The only function that this exit routine performs is posting the routine that is to receive the message that was placed into the mailbox. This posting action informs the receiving routine that there is a message in the mailbox.

When you create your POST exit routine, it must conform to certain MVS requirements and restrictions. Refer to Appendix A, “POST Exit Routine,” on page 121 for instructions for coding a POST exit routine.

Requesting XCF Event Monitoring

When you build a mailbox, you can request that to be notified when XCF events occur that affect your JES XCF group. You might use this option, for example, if your JDU needs to be aware of any state changes that affect JES members of your JES XCF group.

When the system returns XCF event information, the information is returned to your mailbox in the form of a system event message. To map a system event message, use mapping macro IXZYIXSE. To request notification of XCF events, code SYSEVENT=YES on the IXZXIXMB macro.

A system event can be either a JES or XCF event. Each system event message contains an **envelope**. The envelope (IXZYIXEN mapping macro) consists of the header data associated with a message. It contains the delivery information that is required to correctly deliver the message to the specified receiver. Mapping macro IXZYIXSE specifies that the event is either a JES event or an XCF event. If the event is a JES event, then mapping macro IXZYIXJE contains the JES data; if the event is an XCF event, mapping macro IXCYGEPL contains the XCF data. See *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for the IXZYIXEN mapping and the mappings for the other mapping macros discussed here. Also, see Figure 8 on page 25 for a pictorial view of the message envelope (IXZYIXEN mapping macro), and a system event's message structure.

Sending a Message (IXZXIXSM Macro)

JDUs running in any JES environment can use the JES XCF macro IXZXIXSM to send messages to a JDU running in a JES2 or JES3 address space, or to a JES3 JDU running in the functional subsystem (FSS) address space. **Messaging** is the process of one software component (JES JDU) running on one member sending the **data packet** (up to 60 kilobytes of data) to the same or other member in the group.

All messages are received in the order in which they are sent, and multi-segment messages are not sent until the entire message (all segments) is available. This message-ordering feature allows you to eliminate any previously written installation code you might have provided for that purpose. (See “Message Segments” on page 21 for an explanation of message segments.)

Simply stated, IXZXIXSM (send message) allows a JDU to pass a message to JES XCF, which then delivers the message to the requested receiver determined by both the:

- XCF member name
- Mailbox name.

Then:

1. JES XCF places the message in the receiver's mailbox
2. JES XCF passes control to the POST exit routine, which notifies the receiver of a mail delivery
3. The receiver processes the message.

   The receiver **must** acknowledge the message to JES XCF and optionally can return data or an acknowledgement to the sender.

Delivery, and subsequent processing, is determined by three message characteristics that are specified by the value of the REQTYPE= parameter on the IXZXIXSM macro. Refer to Table 5 on page 17 for a summary of the REQTYPE values and the message processing characteristics of each.

Table 5. Request Type Characteristics for Send Message Macro

REQTYPE= Parameter	Synchronous Processing	Acknowledgement Requested from Receiver	Multisystem Recovery Provided
COMM	No	No	No
ASYNCACK	No	Yes	Yes
ASYNC	No	No	Yes
SYNC	Yes	Yes	Yes

Recovery levels

Before continuing the discussion of message request types, some discussion of the levels of recovery is needed as background to the understanding and selection of the proper REQTYPE. There are basically two levels of recovery: one can be considered multisystem level, the other as application level. Refer to Figure 6 on page 18 for an illustration of the levels.

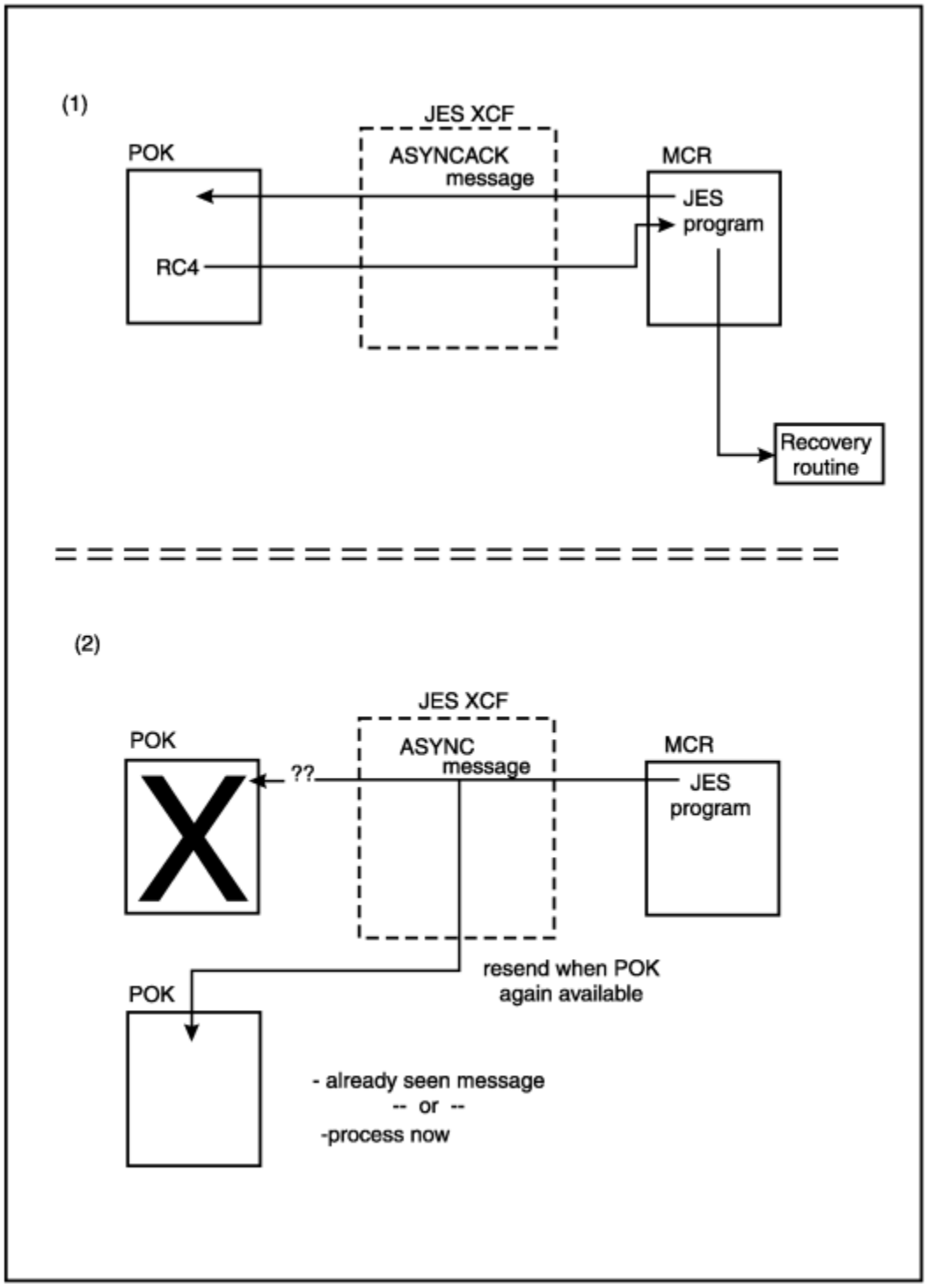


Figure 6. Send message example showing levels of recovery

In Part 1 of Figure 6 on page 18, assume *member MCR* sends an asynchronous with acknowledgement (ASYNCACK) message to *member POK*. (Assume for purposes of this example, the message consists of data and a command to perform some processing of that data.) Member MCR expects that all the data and the command will be received and processed successfully by member POK. However, an assumption is not good enough; member MCR must know if the command was processed, so it requests an acknowledgement upon which it can then base further processing. The acknowledgement returns a return code 4 (arbitrarily, for use in this example, RC=4 means that the data was not received completely). Member MCR then provides a recovery routine of its own to resend the data and command. This level of recovery is at the application level and your application program must provide it if it is needed.

The second level of recovery is considered multisystem level. Part 2 of Figure 6 on page 18 shows that member MCR sends an ASYNC message to member POK. MCR does not request an acknowledgement; it really doesn't need to know if the message is received or not. But before the message is properly acknowledged to JES XCF, member POK is re-IPLed. When member POK is returned to the XCF group, JES XCF automatically resends the message. At this point, member POK will receive it and do one of the following:

- Process it and acknowledge its receipt to JES XCF
- Ignore the message but acknowledge its receipt to JES XCF.

If member POK has already seen this message, JES XCF sets an indicator bit, RESENT_DUE_TO_IPL, in the message envelope IXZYIXEN.

Recovery at this level is provided by JES XCF and is referred to as **multisystem recovery**. That is, JES XCF saves the message and resends it when POK rejoins the group.

With that understanding, consider the various request types available and how critical an acknowledgement is to the sending member. Most JES system messages are sent as REQTYPE=COMM.

- **Synchronous or asynchronous processing**

  A request for synchronous communication causes the JDU to wait until the message is processed by the receiver. Asynchronous communication causes control to be returned to the caller immediately after the message is queued for sending. The type of processing you choose depends on:

  – Whether the JDU is operating in the JES main task environment

    Attention: Do not send REQTYPE=SYNC messages under the JES main task. SYNC implies a wait and potentially such communication can cause highly degraded system performance and a potential processing deadlock.

  – The frequency and size of messages being sent

  – Whether you need to base follow-on processing on the receipt of the acknowledgement and any associated data that is returned by the receiving member.

  Only REQTYPE=SYNC provides synchronous message processing.

- **The need for an acknowledgement**

  You can request an acknowledgement of message receipt and processing, and then use that data to determine follow-on processing. The receiver can provide up to 60 kilobytes of data that can be associated with the acknowledgement. Use REQTYPE=ASYNCACK or SYNC to request an acknowledgement from the receiving member.

- **The level of multisystem recovery**

  Recovery here indicates the action that JES XCF takes if the receiving system fails prior to receipt of the message. JES XCF maintains a copy of the message and its associated routing data on the sending system and automatically attempts to resend that message when the failed member reattaches to the JES XCF group. Recovery is based on the REQTYPE= parameter value on the IXZXIXSM macro as follows:

  COMM
  No recovery

  ASYNC
  JES XCF recovery

  ASYNCACK
  JES XCF recovery

  SYNC
  JES XCF recovery

Sending an Asynchronous Message Without Acknowledgement or Recovery

The default request type for the IXZXIXSM macro is REQTYPE=COMM. COMM indicates that the message being sent is communicated asynchronously, the sender does not request an acknowledgement, and JES XCF does not attempt multisystem recovery if either the sending or receiving member fails prior to message delivery.

This option is useful when the message data is informational only and there is no critical follow-on processing based on the acknowledgement.

Sending an Asynchronous Message With Acknowledgement

Specify REQTYPE=ASYNCACK on IXZXIXSM to indicate that the message being sent is sent asynchronously, but once the message is processed, the sender receives an acknowledgement from the receiver. If recovery is needed, your installation can provide that recovery based on the acknowledgement and return and reason codes returned to the sender.

This option is useful when:

- The amount of message data is large and the sender cannot afford to wait for an acknowledgement.
- The sender needs to know that the message was processed successfully so that the sender can base further processing or attempt recovery on the sets of the return and reason codes such as those returned:
  - In IXZYIXEN (the message envelope)
  - From a macro invocation
  - By specifying the USERRC= parameter of the IXZXIXAC macro.
- The message data is critical to your installation's continued processing, but your installation cannot afford to wait for an acknowledgement. The sender does not need to know immediately that the message data was received but must ensure that JES XCF will get the message to the appropriate member.

How to Associate an Acknowledgement with the Original ASYNCACK Message

For ASYNCACK messages only, you need to associate the acknowledgement with the original message. To do so, specify the REQTOKEN= parameter on the IXZXIXSM macro. This field provides the name or address of a token returned by JES XCF to identify this specific message. The same token is returned in the YIXAC_REQ_TOKEN field of the IXZYIXAC macro (the mapping macro that maps acknowledgements). Refer to “When a Sender Requires an Acknowledgement of a Multi-Segment Message” on page 22 for further discussion of REQTOKEN= for multi-segment messages.

Sending an Asynchronous Message Without Acknowledgement

Specify REQTYPE=ASYNC on IXZXIXSM to indicate that the message being sent is sent asynchronously. This option is similar to REQTYPE=COMM, except that ASYNC requests that JES XCF provide recovery by resending the message if the receiving member fails prior to acknowledging the message.

This option is useful when the message data is critical to your installation's continued processing, but your installation cannot afford to wait for an acknowledgement. The sender does not need to know that the message data was received but must ensure that JES XCF will get the message to the appropriate member.

Sending a Synchronous Message

Specify REQTYPE=SYNC on IXZXIXSM to indicate that the message is sent synchronously so the sender waits until receiving the acknowledgement before continuing processing. This option also indicates that JES XCF provides recovery even if the receiving member fails prior to receiving or acknowledging the message.

This option is useful when:

- A member needs data from another member and must wait until that data is received before processing can continue.

  For example, in a JES3 complex, the local main requires a spool buffer; the local cannot continue its current processing until the global main returns that data.
- The message data is critical to your installation's continued immediate processing. The sender needs to know that the message data was received and requires JES XCF to attempt to resend that data if it was not received.

  For example, in Figure 7 on page 21, the local processor (MCR) sends a synchronous message (a spool allocation request) to the global processor (POK). The member MCR cannot continue any further work until the request is satisfied by the member POK. Therefore, a wait is not relevant to MCR's performance; it is an absolute necessity before MCR continues any further work. Member MCR sends the message, waits for the acknowledgement, and eventually receives the spool space and continues work.

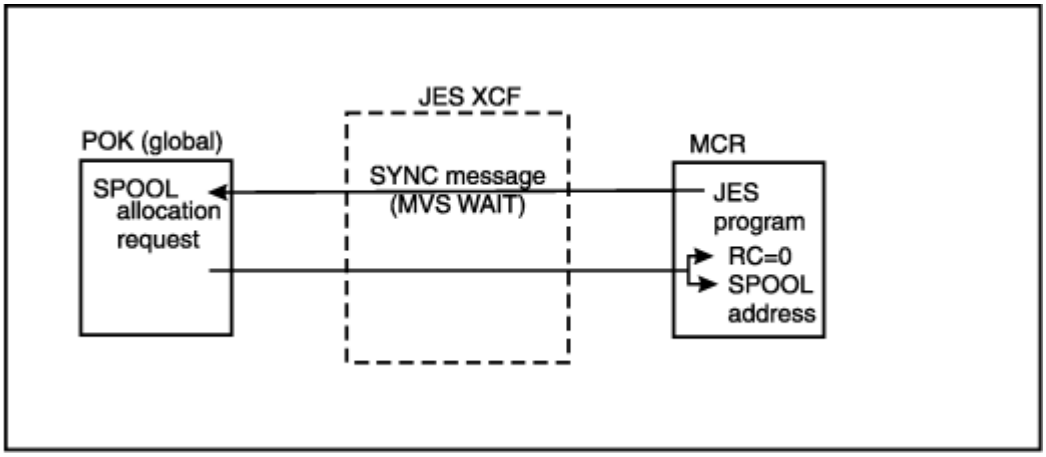


Figure 7. Example synchronous message request

Unless a synchronous message is absolutely necessary, do not use this form of sending messages. Be certain to issue a REQTYPE=SYNC message request only in environments other than the JES main task.

Identifying Which Mailbox is to Receive the Acknowledgement

To complete a message cycle, JES XCF needs to be aware that each message is acknowledged. Therefore, an acknowledgement is always returned to the sending member. Specify the mailbox name that is to receive the acknowledgement with the REQMBOX= parameter on IXZXIXSM.

Message Segments

Messages can be either **single-segment** or **multi-segment** messages. Each segment of the message is then a separate data packet of up to 60 kilobytes of data. JES XCF returns a token on return from sending the first segment. The sender uses this token as input on subsequent segment calls to associate them with each other.

Sending a Single-Segment Message

Specify or accept the default of SEGTYPE=SINGLE on the IXZXIXSM macro to send a single-segment message.

Sending a Multi-Segment Message

Because a multi-segment message is made of multiple parts, those parts need to be associated with each other or connected so that JES XCF can associate all pieces of the original message data and also identify the last segment of the multi-segment message group. To provide this multi-segment link, you must first indicate that the message data is other than a single segment. Use SEGTYPE=FIRST | MIDDLE | LAST on the IXZXIXSM macro to indicate that the segment is the first of a series of segments, a segment other than the first or last of the series of segments, or the last segment of the series, respectively.

A send message call on which you specify SEGTYPE=FIRST provides a CONNECT token on return; use this token as input on all subsequent send segment calls to associate the various segments of the message.

A multi-segment message is useful when the message contains large records (over 60 kilobytes of data altogether) that must arrive at the receiver's mailbox at the same time. JES XCF sends the multi-segment message only when all its segments are available. The individual segments of a multi-segment message are always ordered in the sequence in which they were sent. See "Receiving a Message (IXZXIXRM Macro)" on page 23 for specifics on receiving messages in a mailbox.

Discontinuing a Request to Send a Multi-Segment Message

One final type of message indicates that all previous segments of a multi-segment were sent in error. If, after you have sent one or more segments of a multi-segment message, and realize that you need to discontinue the send request for any reason, you can do so by specifying SEGTYPE=ABORT on the next message segment sent with the IXZXIXSM macro. This segment also can be a null segment if you specify SEGTYPE=ABORT. The message, or null statement, is sent and timestamped, and the receiver returns an acknowledgement. However, it is likely that the receiver will immediately recognize this as "junk mail" and discard the message, although that is not necessary. The message must be received and acknowledged as the receiver processes any other message prior to being discarded. An aborted message is indicated by the ABORT_SEGMENT flag set on in the message envelope (mapped by IXZYIXEN) for both the first and last segments.

When a Sender Requires an Acknowledgement of a Multi-Segment Message

If you request an acknowledgement by specifying REQTYPE=ASYNCACK, you also must specify the REQTOKEN= parameter on the SEGTYPE=FIRST call. REQTOKEN= is required to associate the message acknowledgement with the original message.

When sending a multi-segment message with REQTYPE=SYNC, you must specify the MSGTOKEN= parameter on the last segment, whether the last segment is specified as SEGTYPE=LAST or SEGTYPE=ABORT. MSGTOKEN= is required on synchronous calls to provide an output field to contain the token that represents the acknowledgement. That is, you need to supply a field to hold the token that will later be used to acknowledge the acknowledgement. Table 6 on page 22 shows the required parameters for each SEGTYPE= based on the REQTYPE= call.

Table 6. Required Parameter Token Definitions for SEGTYPE and REQTYPE

REQTYPE=	SEGTYPE= FIRST	SEGTYPE= MIDDLE	SEGTYPE= LAST	SEGTYPE= ABORT
SYNC	--	--	MSGTOKEN=	MSGTOKEN=
ASYNCACK	REQTOKEN=	--	--	--
ASYNC	--	--	--	--
COMM	--	--	--	--

Providing a Place for Data Returned from the Message Receiver

By using the IXZXIXAC macro, the message receiver optionally can return data to the message sender. Specify RESPDATA= on the IXZXIXSM macro to receive the address of a message buffer that can contain a response message, and specify USERRC= on IXZXIXSM to indicate processing results as specified by the message receiver.

Identifying Which Member is to Receive the Message

Use MEMBER= on IXZXIXSM to supply the member name to which the message is sent. This member can be any member of the JES XCF group, including the sending member. Use MBOXNAME= on the IXZXIXSM macro to identify the mailbox name for the receiving member. Once a message is placed in a mailbox, the receiving member's JDU can access and process the message.

Restrictions and IBM Recommendations

Throughout the discussion about sending a message, there are a number of IBM recommendations and cautions; they are summarized here:

- Do not send REQTYPE=SYNC messages under the JES main task.
- Be certain that the receiver of a message can process it. If, for example, two members send each other a REQTYPE=SYNC message, there is the possibility that the two will create a deadlock situation and each wait forever to receive the acknowledgement.
- Use the minimal level of REQTYPE= messaging as required to meet your multisystem recovery needs. The message types are listed here from least (no) recovery to most recovery; as the recovery capability increases, so does the potential for system performance impact.

  1. COMM
  2. ASYNC
  3. ASYNCACK and SYNC
- If a member detaches from JES XCF, be certain to clear the group token field in your application program; it will no longer be valid.

Receiving a Message (IXZXIXRM Macro)

JDUs running in the JES2 or JES3 address space, and JES3 JDUs running in a functional subsystem (FSS) address space, can receive messages. When the system places a message in a JDU's mailbox, the POST exit routine (that you specified on IXZXIXMB POSTXIT= parameter of the IXZXIXSM macro) associated with that mailbox posts the JDU. This posting action notifies the JDU that there is a message to receive.

To receive a message, issue the IXZXIXRM macro. When you issue IXZXIXRM, you must:

- Provide the group token that represents the JES XCF member and group.
- Identify the mailbox from which you are retrieving the message.
- Indicate the type of message you wish to receive. Each time you issue IXZXIXRM, you can receive one of the following:
  - Only system event messages that are in the mailbox (MSGFETCH=SYSEVENTS).
  - Only acknowledgement messages that are in the mailbox (MSGFETCH=ACKS).
  - Only JES- or installation-created messages that are in the mailbox (MSGFETCH=MESSAGES).

    Each segment of a multi-segment message is presented as a single message and all segments are ordered as originally sent. However, individual messages from multiple members might not be ordered in the mailbox in a timestamp order.
  - All messages that are in the mailbox (MSGFETCH=ALL) ordered by type as follows:

    1. SYSEVENTS
    2. ACKS
    3. MESSAGES.
- Provide a location where the system can return a pointer to the message. The system places this message within the address space from which IXZXIXRM is invoked and stores the buffer pointer in the location you provide by specifying the DATA= parameter.
- Provide a location where the system can return the length of the message by specifying the DATALEN= parameter.
- Provide a location where the system can return a message token that represents the message you receive. When you acknowledge receipt of a message, you must use the message token that the system returns in the area you specify through the MSGTOKEN= parameter.

Process an aborted message as you would any message. You must receive and acknowledge it. An aborted message is a multi-segment message that was either sent erroneously or incompletely. The error

was discovered by the sender prior to sending all segments. The last segment indicates the error with the ABORT_SEGMENT bit set on in the message envelope (mapped by IXZYIXEN). The SEGTYPE=ABORT indication is also propagated to the first segment when the last segment is sent. Typically you will recognize and immediately discard such messages. An acknowledgement, as always, is returned to JES XCF.

Messages can be undeliverable because the receiving mailbox or member does not exist or because a mailbox was cleared or deleted prior to a message being acknowledged. JES XCF provides return and reason codes for REQTYPE=SYNC and ASYNCACK messages that identify the exact cause of the failure. But because the SYNC message causes an MVS wait, JES and JES XCF processing for each is different.

- For SYNC messages only, if the mailbox was cleared (through IXZXIXMC) or deleted (through IXZXIXMD), JES XCF eventually provides a return and reason code, but the sender is unable to continue until the return code is returned. Once received, the acknowledgement from IXZXIXMC or IXZXIXMD ends the wait condition, and the sender receives the appropriate return and reason codes.
- For ASYNCACK messages only, JES XCF returns system return and reason codes in the message envelope to provide an explanation for why a message was not received. If the mailbox was cleared or deleted, JES XCF eventually provides a return and reason code, but the sender proceeds with other processing until posted to issue an IXZXIXRM call, at which time, the return and reason codes are available in the message envelope. Fields SYSTEM_RETURN_CODE and SYSTEM_REASON_CODE in the message envelope IXZYIXEN are used to indicate this information to the sender.

Refer to Table 7 on page 31 for the list of system return and reason codes JES XCF can return in the message envelope.

Mapping Messages

When you receive a message, it is preceded by a message envelope. The **message envelope** is the header for the message and contains information that includes:

- The addresses of the sender and receiver of the message
- An identifier of the message type
- An offset to the actual message.

Refer to Figure 8 on page 25 for a pictorial view of the JES XCF mapping structure. Use IXZYIXEN to map the message envelope, and use IXZYIXSE to map a system event message. IXZYIXSE points to either IXZYIXJE (if the message is a JES event) or IXCYGEPL (if the message is a XCF event). These macros, in turn, provide the actual event mapping. Also, refer to “Obtaining information about JES members in the XCF group (IXZXIXIF macro)” on page 26 for a discussion of the IXZXIXIF macro and the data it returns.

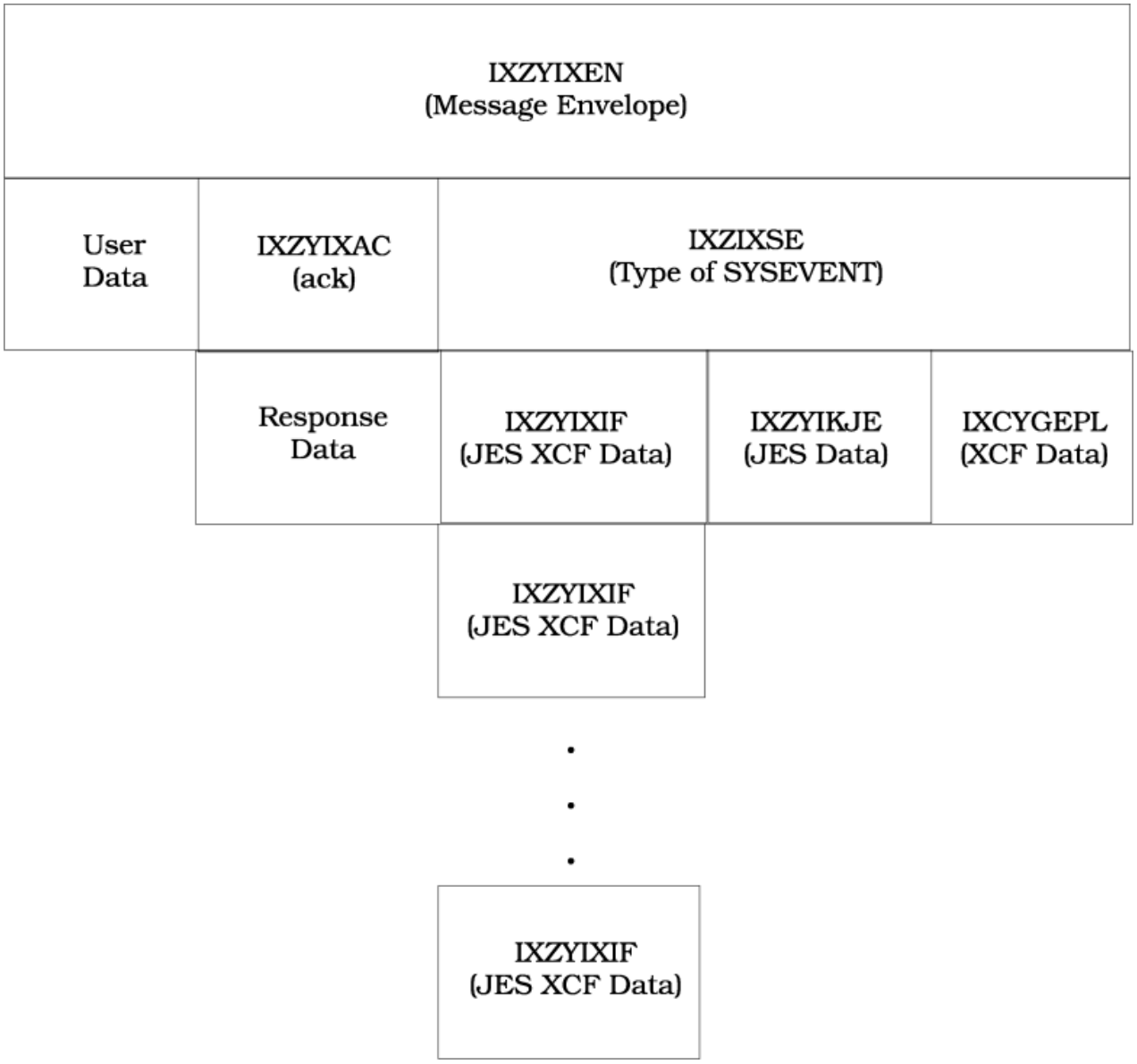


Figure 8. JES XCF message mapping macro structure

To map an acknowledgement message, use the IXZYIXAC mapping macro.

Your installation is responsible for providing mapping macros for installation-created messages.

Restrictions

Be certain that whenever you receive messages from a mailbox, you do it under the same task in which the mailbox was built.

Acknowledging Receipt of a Message (IXZXIXAC Macro)

You must acknowledge all messages that you receive to JES XCF whether or not you acknowledge receipt of a message to the sender.

Failure to acknowledge a message could have an adverse impact on the JDU that sent the message: the JDU, for example, might suspend processing while waiting for the acknowledgement. Also, resources held by JES XCF for a message that is not acknowledged continue to be held, thus making them unavailable for use.

After a JDU has received and processed a message, it must acknowledge receipt of the message. Every message received must be acknowledged; even if the sender of the message does not request an acknowledgement, JES XCF needs the acknowledgement. If the sender of the message requests an acknowledgement, the system returns the JDU's acknowledgement message as follows:

- For REQTYPE=ASYNCACK messages, to the mailbox specified by the REQMBOX= parameter of the IXZXIXSM macro
- For REQTYPE=SYNC messages, to the address as specified by the RESPDATA= parameter of the IXZXIXSM macro.

After you acknowledge a message, the system reclaims any resources held by the original message.

To acknowledge a message, issue the IXZXIXAC macro. When you issue IXZXIXAC, you:

- **Must** provide the group token that represents the JES XCF member. (GROUPTOKEN= parameter)
- **Must** provide the message token that the system returned when you received the message. (MSGTOKEN= parameter)
- Can provide a pointer to a data area that contains information to be sent to the sending member along with the acknowledgement. The data area is made available as part of the acknowledgement message. The format and content of the data area must be previously agreed to by the sending and receiving JDUs. (DATA= parameter)
- Can provide the length of the data area. (DATALEN= parameter) (DATALEN= must be specified if you specified a pointer to a data area with the DATA= parameter.)

Optionally, you can provide a return code as part of the acknowledgement message by using the USERRC= parameter on IXZXIXAC. The values and meaning of the return code must be previously agreed to by the sending and receiving JDUs.

Obtaining information about JES members in the XCF group (IXZXIXIF macro)

Your JDU might need to obtain information about other JES members in the JES XCF group. For example, you might need to know whether another JES has completed initialization. You can obtain this information by issuing the IXZXIXIF macro. When you issue IXZXIXIF, you must provide either the group token that represents the XCF member (GROUPTOKEN= parameter) or the group name of the JES XCF group (GROUPNAME= parameter).

IXZXIXIF can return much more data to the member requesting the information than that member requires. To limit the data JES XCF returns, you can use a set of *filtering* parameters on the IXZXIXIF macro call to provide only data for certain members or members in certain states. To use the filtering parameters, you **must** specify GROUPNAME=. The following lists these filtering parameters with an explanation of the type of data they can suppress:

IXZXIXIF Parameter
Limit data based on whether member is:

STATE=
Active (a member on an *active* MVS system only)

SYSTEM=
Current (a member on the MVS system from which this IXZXIXIF macro was issued)

POLYJES=
One of the JES2 members running on an MVS system, and you require information returned from only a single JES2 member. (This parameter has meaning in a JES2 environment only.)

FUNCTION=
OS/390 MVS Version 2 Release 6 or higher and supports the automatic restart manager

You can specify any or none of these parameters; they are all optional and each defaults to no data filtering. Therefore, if you do not set them, JES XCF returns data from all members of the JES XCF group in any state.

To map the member information records that the system returns, use mapping macro IXZYIXIF. The data returned by IXZXIXIF can be either put in a mailbox or put into a named buffer. If you choose to have the data put in a mailbox, use the REQMBOX= parameter on IXZXIXIF to identify the mailbox name. The data is returned to the mailbox and considered by JES XCF as system event data. Therefore, you need to specify MSGFETCH=ALL or MSGFETCH=SYSEVENT when issuing an IXZXIXRM (receive message) macro. If you choose to have the data returned in a buffer, specify the ANSAREA= and ANSLEN= parameters on IXZXIXIF to specify the name and length of the buffer. To use a buffer, you need to obtain storage for it. Calculate the size of this buffer as follows:

```
buffer size = (size of IXZYIXIF mapping macro) x (number of JES XCF group members)
```

If you have miscalculated this value and provide a buffer that is too small, JES XCF returns the necessary size in the field you specified on the ANSLEN= parameter. You must then free the incorrectly sized buffer, obtain one of the required size, and invoke IXZXIXIF again. The data IXZXIXIF returns in the buffer is available in an array format as shown in Figure 9 on page 27. Note that the last element in the array contains a value of 0 (zero).

Attention:

IBM recommends that you use REQMBOX= to receive the data JES XCF returns. Particularly when running under the JES main task, the JDU using this method minimizes potential performance degradation when the JDU issues an IXZXIXIF call. Performance can suffer because IXZXIXIF requires I/O, therefore, issue IXZXIXIF sparingly. When you do issue IXZXIXIF, you can minimize the performance impact for the JDU by using a mailbox rather than a buffer. Using a mailbox causes asynchronous processing of the request, whereas using a buffer causes synchronous processing.

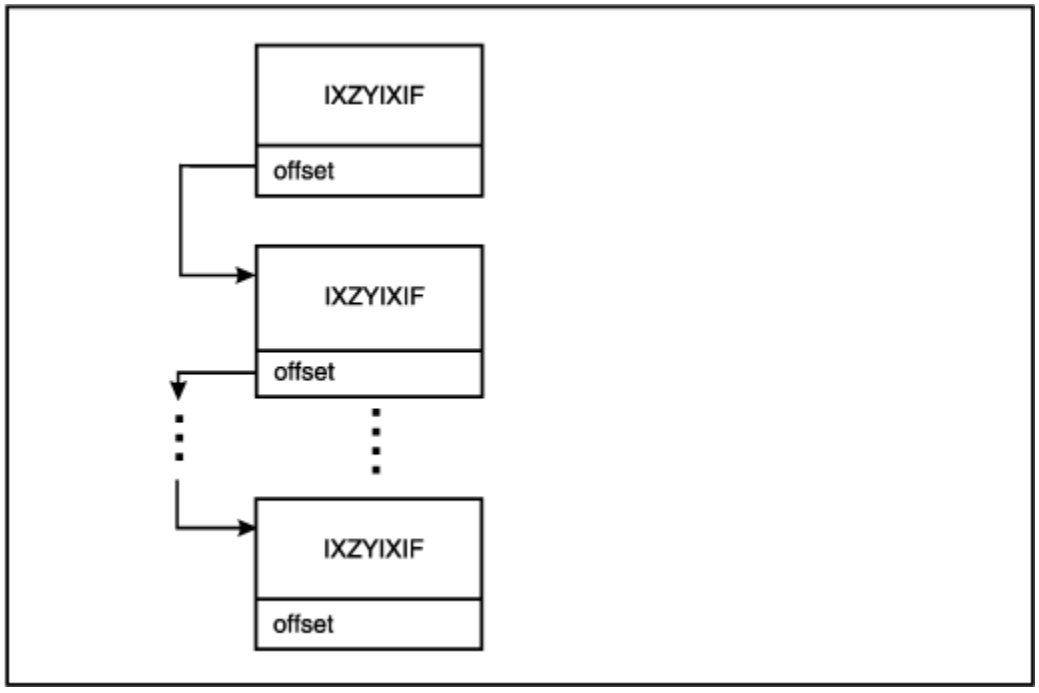


Figure 9. Array format of data returned by IXZXIXIF

[Programming Interface Information] Collecting JES3 performance information data (IXZXIXPI macro)

Resource monitoring products such as Resource Measurements Facility (RMF) need to be aware of any JES3 delays that impact performance of job processing or the JES3 subsystem. This data is not maintained by JES3, but rather collected by JES XCF on its message queues. Therefore, when running JES XCF in a JES3 environment, you might find it necessary to access and extract data from the JES XCF message queues.

JES XCF provides macro IXZXIXPI as its interface to the JES XCF message queues. When you issue IXZXIXPI, you must:

- Obtain storage for the performance information data stream area that JES XCF uses to contain the information it extracts from its message queues.
- Identify the name of the field or a register that contains the address of the performance information data stream. (PIDS= parameter)
- Identify the name of the field or a register that contains the size of the performance information data stream area specified on the PIDS= parameter (PIDSSIZE= parameter). If the size of the area is too small, JES XCF returns the correct size requirement in this field.
- Provide the group token that identifies your XCF member. (GROUPTOKEN= parameter)

JES XCF returns the JES3 processing data in key-length data (KLD) format. Figure 10 on page 29 shows an example of this data stream.

```
1 2  3     6  7 8  9     12 13   15   17    20 21   24  25   27    30
0001 00000034 0002 00000002 0138 0003 00000004 00000001 0004 0000001C
31   33    36 37   39   41    44 45 46   48    51 52
0005 00000002 0298 0006 00000001 17 0007 00000001 00
```

Byte
Meaning

1 - 2
Key indicating the beginning of the data stream (1)

3 - 6
Length of the data stream (34)

7 - 8
JES ASID key (2)

9 - 12
Length of ASID (2)

13 - 14
ASID value (138)

15 - 16
'Number of delays' key (3)

17 - 20
Length of 'number of delays' (4)

21 - 24
Number of delays (1)

25 - 26
Delay entry key (4)

27 - 30
Delay entry length (1C)

31 - 32
Requestor's ASID key (5)

33 - 36
ASID length (2)

37 - 38
ASID value (298)

39 - 40
Request type key (6)

41 - 44
Request type length (1)

45
Request type (17)

46 - 47
Request subtype key (7)

48 - 51
Request subtype length (1)

52
Request subtype (0)

Figure 10. Example JES3 Performance Data Returned by IXZXIXPI

The performance information data stream mapping macro, IXZYPIDS, maps the returned keys. See *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for a description of that mapping.

[End Programming Interface Information]

Indicating that system cleanup is complete (IXZXIXCL macro)

If a system in the sysplex fails, JES XCF needs to be aware that JES is initiating and handling necessary resource and system cleanup. When a system fails, mapping macro IXCYGEPL returns a token that identifies the system. Using IXZXIXCL, you can then pass this information to JES XCF to initiate follow-on processing. See *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for a description of mapping macro IXCYGEPL.

When you issue IXZXIXCL, you must:

- Provide the group token that identifies your XCF member. (GROUPTOKEN= parameter)
- Specify the name of a field that contains the XCF system token of the failed system. (FAILEDSYS= parameter). Mapping macro IXCYGEPL provides this token.

Clearing a Mailbox of Messages (IXZXIXMC Macro)

Your JDU might need to clear messages from its mailbox. When you clear a mailbox, JES XCF acknowledges those messages it clears. Further, for message type ASYNCACK, JES XCF places a return and reason code in the message envelope in fields SYSTEM_RETURN_CODE and SYSTEM_REASON_CODE indicating the messages were cleared but not processed by the receiver.

When you issue IXZXIXMC, you must:

- Identify the mailbox to be cleared by providing the mailbox name. (MBOXNAME= parameter)
- Provide the group token that identifies your XCF member. (GROUPTOKEN= parameter)

Be certain that whenever you clear a mailbox, you do it under the same task that built the mailbox.

Deleting a Mailbox (IXZXIXMD Macro)

When a JDU terminates or no longer needs to communicate with other JDUs, you can delete the mailbox. When you delete a mailbox, JES XCF acknowledges those messages it deletes. Further, for message type ASYNCACK, JES XCF places a return and reason code in the message envelope, IXZYIXEN, in fields SYSTEM_RETURN_CODE and SYSTEM_REASON_CODE, indicating the messages were deleted, but not processed by the receiver.

To delete a mailbox, issue the IXZXIXMD macro. When you issue IXZXIXMD, you must:

- Identify the mailbox to be deleted by providing the mailbox name. (MBOXNAME= parameter)
- Provide the group token that identifies your JES XCF member. (GROUPTOKEN= parameter)

Be certain that whenever you delete a mailbox, you do it under the same task that built the mailbox.

System Return and Reason Codes When Messages are Undelivered

JES XCF returns system return and reason codes if a message is undelivered. In mapping macro IXZYIXEN, SYSTEM_RETURN_CODE contains one of the decimal return codes and SYSTEM_REASON_CODE contains one of the decimal reason codes listed in Table 7 on page 31.

Table 7. System Return and Reason Codes		
Return Code	**Reason Code**	**Meaning**
4	4	Message was acknowledged by JES XCF because it was in a mailbox that was cleared.
4	8	System error prevented the message from being delivered.
4	16	Message was acknowledged by JES XCF because it was in a mailbox that was deleted.
4	20	Message was acknowledged by JES XCF because it could not be delivered. Either the destination member or mailbox does not exist.

What Happens to Mailboxes When the JES Session Ends

JES2 and JES3 provide somewhat different processing based on how the current JES session ends; therefore the mailboxes being used are treated differently.

If JES2 abnormally terminates (abends) or JES3 terminates (except JES3 FSS), mailboxes are not deleted. However, when the JES member is restarted, you need to rebuild the mailbox. Perform the rebuild (using the IXZXIXMB macro) after attach processing so that the group token is available. Also be certain to name the mailbox with the same name it had prior to the abnormal end.

For JES2 and JES3 FSS, if the attachment is broken with a detach (through use of the IXZXIXDT macro), the member is no longer active in the JES XCF address space, the mailbox is emptied, and mailbox resources are reclaimed

Detaching from a JES XCF Group (IXZXIXDT Macro)

Once you have created an installation-defined attachment to JES XCF, IBM recommends that you keep that attachment active until the JES address space terminates normally (that is, when JES detaches itself from JES XCF). When the JES address space detaches from JES XCF, the system breaks the attachment and reclaims any resources held by that attachment.

If you decide to break an attachment to JES XCF, do so only when IXZXIT03 gets control during JES termination processing. Be aware that if a program attempts to use the group token following the detach, the token will not be valid and its continued use may cause unpredictable results.

[Programming Interface Information] Component trace and diagnostic support

The trace records discussed in this section are useful only if you need to contact your IBM support center and have to supply requested diagnosis data. Your IBM representative may request that you request an SVC dump (using the DUMP operator command) of the JES XCF address space or collect other relevant data.

JES XCF provides multiple component trace support in the form of the following sub-level types:

- Module footprint trace (FLOW sub-level)
- System event trace (XCFEVT sub-level)
- Detailed message trace (MSGTRC sub-level)
- Installation exit trace (USRXIT sub-level).

This section provides a general description of the trace types. Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for a detailed description of each trace and how to invoke it. Figure 11 on page 32 shows the

basic JES XCF tracing structure. Each JES XCF trace is invoked as a sub-level trace (SUB= keyword on the TRACE command) under the system component ID of SYSJES.

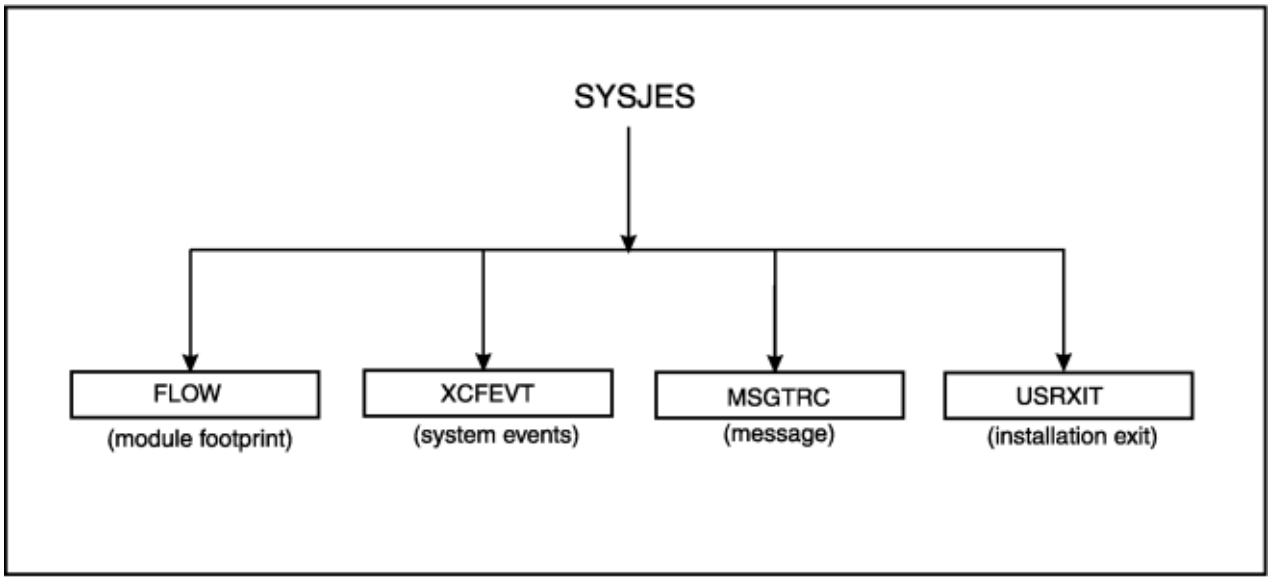


Figure 11. JES XCF trace structure

For example, to trace messages, enter:

```
TRACE CT,ON,COMP=SYSJES,SUB=(MSGTRC)…
```

For a description of the MVS TRACE command, refer to *z/OS MVS System Commands*.

[End Programming Interface Information]

Controlling the Quantity of Traced Data

Each trace type must be started and stopped individually, but be aware that trace types MSGTRC, USRXIT, and XCFXIT are always active at a minimal level of tracing; FLOW defaults to the detailed level of tracing.

At the minimal trace level, MSGTRC, USRXIT, and XCFEVT trace only error events. By turning the detailed trace on, you can optionally request detailed MSGTRC, USRXIT, and XCFEVT trace records of events such as those associated with messages or exits.

FLOW trace is at a detailed level and by default is always on. FLOW provides trace data for all error and non-error processing events.

The data obtained from these traces is useful when you need to contact your IBM support center.

Obtaining Trace Data

Use the CTRACE external writer to obtain trace data, and use the IPCS merge function to merge trace data chronologically into a single report. You can either collect trace records in a single external writer, or use individual writers to collect individual trace types. Also, you can view trace records as part of a stand-alone dump or SVC dump. Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for a discussion of the CTRACE external writer, and *z/OS MVS IPCS Commands* for IPCS merge information.

Module Footprint Tracing (FLOW Sub-Level)

The module footprint trace records messages and events as they flow through the JES XCF component. FLOW always traces all available records; that is, all noteworthy events such as errors, system state changes, and all processing events associated with messages and systems events. Using the IPCS CTRACE EXCEPTION subcommand, you can filter FLOW trace data to trace only non-standard (unexpected) events.

XCF and JES XCF Event Tracing (XCFEVT Sub-Level)

The system event trace records XCF and JES XCF event data that is processed by JES XCF. XCF and JES XCF always trace all available records; that is, all error events and all processing events associated with systems and group members.

Detailed Message Tracing (MSGTRC Sub-Level)

The message trace records all messages that are handled by JES XCF. The data is recorded at specific points in JES XCF processing from the time a JDU issues the IXZXIXSM macro through the time the receiving JDU issues the IXZXIXRM macro. Message tracing also traces acknowledgements that are returned to message senders.

This trace produces a high volume of data; therefore, you can optionally start and stop detailed message tracing.

Installation Exit Tracing (USRXIT Sub-Level)

The installation exit trace records the parameter list (IXZ$XPL) and message data passed to and returned from installation exits IXZXIT01, IXZXIT02, and IXZXIT03.

This trace produces a high volume of data; therefore, you can optionally start and stop detailed exit tracing.

Ending and Restarting the JESXCF Address Space

For a variety of reasons you might need to end the JESXCF address space and then restart it. Indications that you might need to bring the JESXCF address space down include:

- JESXCF abended with system code DC5 or EC5
- One or more members of your system have become unresponsive
- A review of SYSLOG indicates a JESXCF problem
- On a JES2 system, you receive the $HASP501 and MVS IXZ0108E messages

If you determine, based on system messages and diagnostic codes, that you must end JESXCF, you must first dump the JES and JESXCF address spaces prior to restarting your JES and JESXCF. Follow the procedures listed here:

1. Dump the JES and JESXCF Address Spaces

   If the problem appears to be between the JES (JES2 or JES3) address space and the JESXCF address space on a SINGLE MVS image gather the following documentation:

   - Dump of the JES address space and its data spaces
   - Dump of the JESXCF address space and its data spaces
   - SYSLOG from MVS.

   If the problem appears to be between the JES (JES2 or JES3) address space and the JESXCF address spaces on MULTIPLE MVS images gather the following documentation:

   - Dump of the JES address space and its data spaces on each MVS image involved.
   - Dump of the JESXCF address space and its data spaces on each MVS image involved.
   - Dump of the XCFAS address space and its data spaces on each MVS image involved.
   - SYSLOG from each MVS image

   The following scenario helps to illustrate the steps you need to take. Assume the an environment of two MVS images:

```
-   SY1 - member name JESA
-   SY2 - member name JESB
-   JESA and JESB are running in the same MAS
```

```
-   JES2 is the primary subsystem on each
-   Both JES2s are members of a JESXCF group called WSC
```

a. Member JESA has tried to start an NJE session with another node but the connection does not complete.

b. SY1 received the following message:

```
$HASP501 NJE SIGNONS DELAYED, RESPONSES NOT RECEIVED FROM MEMBER JESB
```

c. A review of previous SYSLOG messages finds SY1 has previously received the following message:

```
IXZ0108E COMMUNICATION FROM WSC$JESB TO WSC$JESA HAS BEEN LOST, GROUP WSC
```

d. To determine the status of the WSC group as detected by MVS XCF, issue the following MVS command on both SY1 and SY2:

```
           D XCF,GROUP,WSC,ALL
```

e. The response on both SY1 and SY2 is:

```
IXC333I  hh.mm.ss  DISPLAY XCF INFORMATION FOR GROUP WSC
           MEMBER NAME:  SYSTEM:  JOB ID:  STATUS:
           WSC$JESA      SY1      JESXCF   ACTIVE
           WSC$JESB      SY2      JESXCF   ACTIVE
```

f. The response shows the two JESXCF members to be in a normal (active) state. Therefore, you do not need to take a dump of MVS XCF on either system.

g. Since the time between the IXZ0108E message and the $HASP501 message has been several minutes it appears there is a problem somewhere in the JESA/JESXCF(SY1) or JESB/JESXCF(SY2) flow. To gather the necessary documentation for problem determination, issue the following MVS dump command on SY1: (Because you can't determine which MVS image is the source of the problem, use this command to dump the JESXCF and JES2 address spaces and associated data spaces on both MVS images.)

```
DUMP COMM=(Start of NJE on SY1 hanging)
r ww,JOBNAME=(JES?,JESXCF),CONT
r xx,DSPNAME=('JES?'.*,'JESXCF'.*),CONT
r yy,REMOTE=(SYSLIST=SY2('JES?','JESXCF'),DSPNAME,SDATA),CONT
r zz,SDATA=(COUPLE,RGN,ALLNUC,CSA,PSA,SQA,SUM,TRT),END
```

Refer to OS/390 System Commands for additional information on the DUMP command and the use of the wildcard characters '?' and '*' within the command.

2. Dump JESXCF and Other JESXCF Group Member Address Spaces. Your system might include other members of the JESXCF group:

- In JES3 -- other than the JES3 address spaces on the JES3 global and JES3 locals, these members might include:
  - - JES3 writer FSS address spaces on any processor,
  - - JES3 C/I FSS address spaces on any processor,
  - - the JES3DLOG address space on the JES3 global (in release HJS5521 and higher)
- In JES2 -- the members of the JES2 MAS

a. To obtain a list of all currently defined XCF groups issue the MVS command:

```
           D XCF,GROUP
```

b. Then, to obtain the names of all the members of the specified group, issue the MVS command:

```
           D XCF,GROUP,groupname,ALL
```

c. In a manner similar to the previous example, where the problem appears to be between the JES and JESXCF, if the problem appears to be between JESXCF and any other JESXCF members,

dump the JESXCF address space, the other specified JESXCF members, and any associated data spaces.

3. Ending and Restarting JESXCF

   If you determine the need to end the JESXCF address space due to a failure in the JESXCF address space or you issued an operator-initiated request to terminate JESXCF, follow these steps to restart JESXCF:

   - In a JES2 Environment

     a. Issue the following JES2 command to end JES2:

     ```
     $P JES2,ABEND
     ```

     If JES2 does not end, then issue the following JES2 command:

     ```
     $P JES2,ABEND,FORCE
     ```

     b. After the JES2 address space ends, issue the following MVS commands to end JESXCF:

     ```
     FORCE JESXCF
     FORCE JESXCF,ARM
     ```

     c. After the JESXCF address ends, hot start JES2. Both JES2 and JESXCF will start.

   - In a JES3 Environment

     To restart the JESXCF address space in a JES3 environment a MVS IPL is required to recover critical resources.

Chapter 3. Coding the JES XCF exits

This chapter provides reference information necessary to code each of the JES XCF exits:

- IXZXIT01 - Transport Exit
- IXZXIT02 - Receive Exit
- IXZXIT03 - Attach/Detach Exit.

Each exit description provides:

- The purpose of the exit and how your installation can use the exit to modify processing of JES XCF.
- The parameter list that is passed to the exit and how to map the parameter list.
- The environment that exists when the exit receives control:
  - Address space in which the exit runs
  - Addressing mode (AMODE)
  - Address space control mode (ASC mode)
  - Authorization
  - Cross-memory mode
  - Dispatchable unit mode
  - Interrupt status
  - JES environments
  - Locks held at entry
  - Residency mode (RMODE)
  - Storage attributes (pageable and key)
- Register usage:
  - Content when the exit receives control
  - Content when the exit returns to its caller.
- How the exit communicates with its caller.
- Restrictions or requirements that affect the design of your exit routine.
  - JES or MVS services that the exit cannot use
  - Subpool usage requirements and restrictions
  - Maximum message lengths
  - Recovery responsibilities.

Creating Your Own Attachment to JES XCF

When JES initializes, it uses the IXZXIXAT macro to add the JES member as an active member of the JES XCF group. You also can use this same macro to create another attachment to the JES XCF component, a connection that is completely independent of the JES member attachment.

Determining Your Need to Create Your Own JES XCF Attachment

Your installation's need to create its own JES XCF attachment should be based on either 1) whether JES members will be sending large amounts of data or 2) whether your modifications will adversely affect system performance.

If you expect to use the JES-defined attachment to send large amounts of data, that added load can affect JES' ability to communicate between its members. This is likely to degrade performance and impact members' ability to communicate critical information across the JES XCF group. To maintain JES' high-speed delivery of such information, you cannot allow your installation to over-use the JES-defined attachment for other less critical communication needs.

If however, you do not expect to send large amounts of data or do not expect to use the attachment often, the JES-defined attachment might very well meet your installation's needs. Therefore, before creating another attachment, use the JES-defined attachment and closely monitor system performance. If you determine that you do not need to create a separate attachment to JES XCF, you can still use JES XCF to, for example, modify and reroute messages. To do so, you will need to retrieve the default JES XCF group token using the procedure in "Retrieving the JES XCF Group Token" on page 13.

If you determine that you will be sending large amounts of data, or have experimented with the JES XCF attachment and found performance degradation, then you will need to create an independent attachment to the JES XCF component.

Note: IBM recommends that you issue the IXZXIXAT service to attach a JES XCF member only once. Issue the IXZXIXAT macro in exit IXZXIT03 when the exit gets control during JES member initialization.

Defining a JES XCF Group

JES2 and JES3 use JES XCF for the same purpose - to communicate data among the JES XCF group members. To allow both MVS and all the JES XCF members to know about each other, you need to define the members to JES2 or JES3 in their respective initialization data sets.

In a JES2 environment, refer to: *z/OS JES2 Initialization and Tuning Guide* and *z/OS JES2 Initialization and Tuning Reference* for defining a JES2 MAS name and relating that MAS name to the group name defined on a JES XCF attach (IXZXIXAT).

In a JES3 environment, refer to: *z/OS JES3 Initialization and Tuning Guide* and *z/OS JES3 Initialization and Tuning Reference* for defining the JES3 complex name and for relating that complex name to the group name defined on a JES XCF attach (IXZXIXAT).

Using the JES XCF exits

As noted previously, there are a number of restrictions placed on the use of the JES XCF macros and their use in specific exits or JES3 DSPs. Remember, only two of the JES XCF macros (IXZXIXAT and IXZXIXDT) can be used in a JES XCF exit (IXZXIT03). All other JES XCF macros are for your use in standard JES exits and JES2 DSPs only.

To illustrate an attach, send, receive, detach scenario, refer to Figure 12 on page 39, which presents the overall flow between two JES members of the same JES XCF group. The example presented here shows the steps both JES members *MCR* and *POK* take to communicate. The figure shows the macros invoked, the exits used, and the basic flow for an entire asynchronous acknowledgement message cycle. Following Figure 12 on page 39 is sample code for each step.

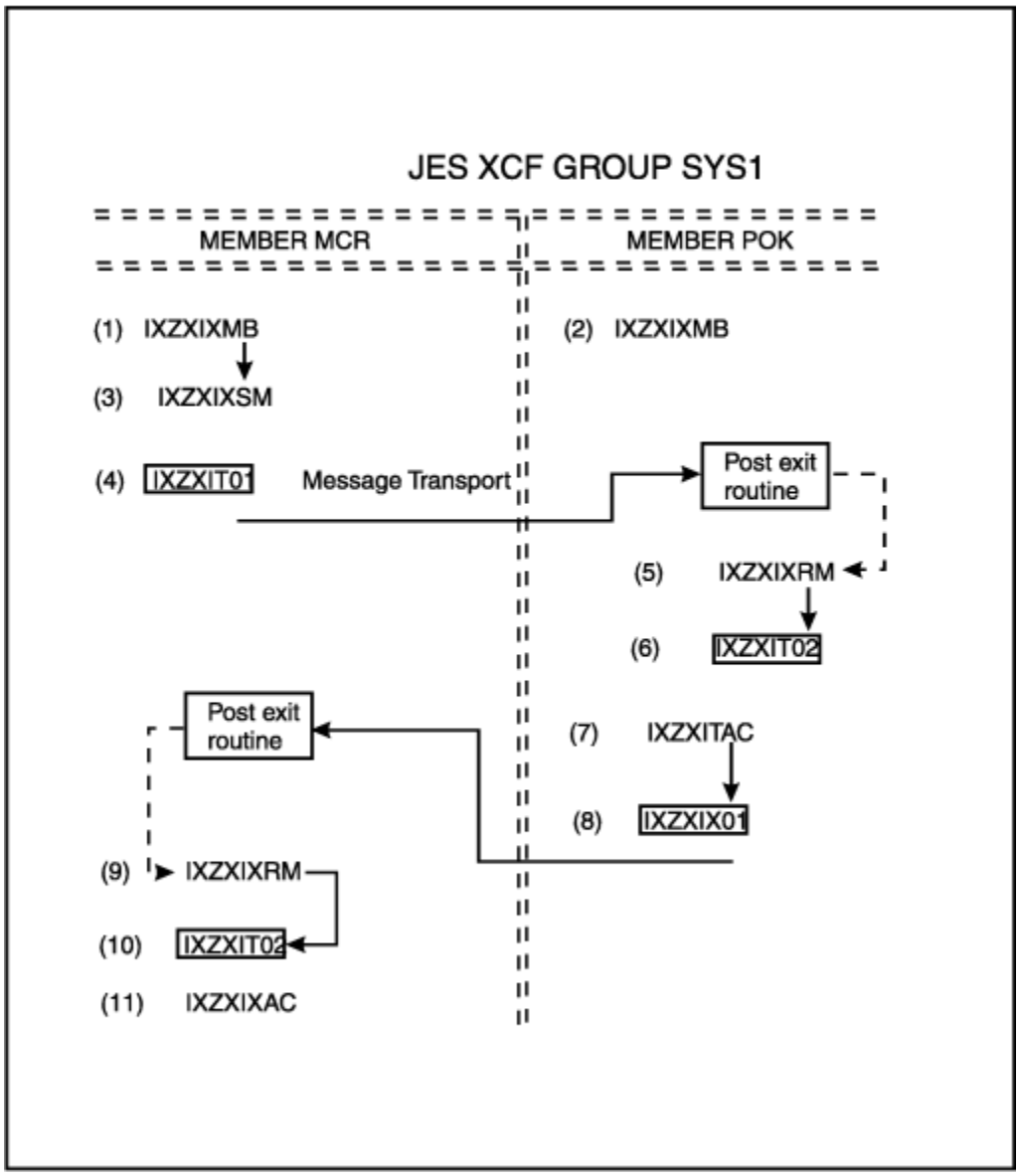


Figure 12. Example of using JES XCF macros and exits for an ASYNCACK message cycle

The following numbered steps describe the events in Figure 12 on page 39.

1 MCR builds mailbox *mcr mailbox* (using the IXZXIXMB macro in a standard JES exit or JES3 DSP)
2 POK builds mailbox *pok mailbox* (using the IXZXIXMB macro in a standard JES exit or JES3 DSP)
3 *MCR* sends a message to member *POK* (using the IXZXIXSM macro in a standard JES exit or JES3 DSP)
4 Exit IXZXIT01 is invoked to modify the ASYNCACK message before delivery to the *POK* mailbox
5 *POK* receives the message from *MCR* (using the IXZXIXRM macro in a standard JES exit or JES3 DSP)
6 Exit IXZXIT02 is invoked to modify the ASYNCACK message before *POK* receives the message
7 *POK* acknowledges the message from *MCR* (using the IXZXIXAC macro in a standard JES exit or JES3 DSP)
8 Exit IXZXIT01 is invoked to modify the acknowledgement to member *MCR* before delivery to the *MCR* mailbox
9 *MCR* receives the acknowledgement from *POK* (using the IXZXIXRM macro in a standard JES exit or JES3 DSP)
10 Exit IXZXIT02 is invoked to modify the ASYNCACK message acknowledgement from *POK* before *MCR* receives the acknowledgement.

11 *MCR* acknowledges the acknowledgement from *POK* (using the IXZXIXAC macro in a standard JES exit or JES3 DSP)

Steps A and B (member attach) and steps C and D (member detach) are provided in the following series of macro coding examples for completeness. It is **not** the intention to suggest that you would ever attach members, send a single message, and then immediately detach members.

(A) Exit IXZXIT03 (Attach/Detach) for MCR

```
⋮
attach   IXZXIXAT GROUP=GROUP,MEMBER=MCR,WHICHJES=JES2,    X
         RELEASE=FMID,GROUPTOKEN=MCRGTOKE,                 X
         RTNCODE=RTNCODE,RSNCODE=RSNCODE
⋮
GROUP     DC  CL8'IBMVSLAB'
MCR       DC  CL16'MCR'
FMID      DC  CL8'HJE5510'
MCRGTOKE  DS  F
RTNCODE   DC  F'0'
RSNCODE   DC  F'0'
⋮
```

Member *MCR* attaches to JES XCF group *IBMVSLAB*. The group token is returned in field *MCRGTOKE*.

(B) Exit IXZXIT03 (Attach/Detach) for POK

```
⋮
attach   IXZXIXAT GROUP=GROUP,MEMBER=POK,WHICHJES=JES2,    X
         RELEASE=FMID,GROUPTOKEN=POKGTOKE,                 X
         RTNCODE=RTNCODE,RSNCODE=RSNCODE
⋮
GROUP     DC  CL8'IBMVSLAB'
POK       DC  CL16'POK'
FMID      DC  CL8'HJE5510'
POKGTOKE  DS  F
RTNCODE   DC  F'0'
RSNCODE   DC  F'0'
⋮
```

Member *POK* attaches to JES XCF group *IBMVSLAB*. The group token is returned in field *POKGTOKE*.

(1) Mailbox Build (IXZXIXMB) for MCR

```
⋮
mbbuild  IXZXIXMB MBOXNAME=MBOXNAME,POSTXIT=POSTXIT,      X
         POSTDATA=POSTDATA,GROUPTOKEN=MCRGTOKE,SYSEVENT=NO,...
⋮
MBOXNAME  DC  CL16'MCRbMAILBOX'
POSTXIT   DS  A
POSTDATA  DS  A
⋮
```

Member *MCR* builds mailbox *MCRbMAILBOX*. Field *POSTXIT* contains the address of the POST exit routine. Field *POSTDATA* contains the address of a data area passed to the POST exit routine.

(2) Mailbox Build (IXZXIXMB) for POK

```
⋮
mbbuild  IXZXIXMB MBOXNAME=MBOXNAME,POSTXIT=POSTXIT,      X
         POSTDATA=POSTDATA,GROUPTOKEN=POKGTOKE,SYSEVENT=NO,...
⋮
MBOXNAME  DC  CL16'POKbMAILBOX'
POSTXIT   DS  A
POSTDATA  DS  A
⋮
```

Member *POK* builds mailbox *POK*␣*MAILBOX*. Field *POSTXIT* contains the address of the POST exit routine. Field *POSTDATA* names the field that contains the address of a data area passed to the POST exit routine.

(3) Send Message (IXZXIXSM) from MCR to POK

```
⋮
send     IXZXIXSM MBOXNAME=POK_MBOX,MEMBER=POK,           X
         GROUPTOKEN=MCRGTOKE,DATA=TOMESSAGE,DATALEN=DLENGTH,    X
         REQTYPE=ASYNCACK,REQTOKEN=TOTOKEN,REQMBOX=MCR_MBOX,    X
         RTNCODE=RTNCODE,RSNCODE=RSNCODE
⋮

TOMESSAGE DC  A(MESSAGE)
DLENGTH   DC  A(L'MESSAGE)
MESSAGE   DC  CL80'Message to process'
MCR_MBOX  DC  CL16'MCR␣MAILBOX'
POK_MBOX  DC  CL16'POK␣MAILBOX'
TOTOKEN   DS  XL8
RTNCODE   DS  F
RSNCODE   DS  F
⋮
```

Member *MCR* sends an ASYNCACK message to member *POK*'s mailbox (*POK*␣*MAILBOX*). Field *TOMESSAGE* contains the address of the message being sent. REQTOKEN defines a field to contain a token returned by send message processing to associate the message acknowledgment with this message. The acknowledgement is to be returned to MCR␣MAILBOX.

4 Exit IXZXIT01 is invoked on member *MCR* to modify the message before delivery to *POK*␣*MAILBOX*.

(5) Receive Message (IXZXIXRM) from MCR to POK

```
⋮
receive  IXZXIXRM MBOXNAME=POK_MBOX,GROUPTOKEN=POKGTOKE, X
         MSGTOKEN=MTOKEN,DATA=DATA,DATALEN=DLEN,...
⋮
POK_MBOX  DC  CL16'POK␣MAILBOX'
DATA      DC  A(0)
DLEN      DC  F'0'
MTOKEN    DS  XL8
⋮
```

The message is sent to member *POK*. The POST exit routine posts member *POK*, which then receives the ASYNCACK message from *POK*␣*MAILBOX*. Field *DATA* contains the address of the message to be received. MSGTOKEN defines a field to contain the token member *POK* is to use to associate its acknowledgement with this message.

6 Exit IXZXIT02 is invoked on member *POK* to modify the message before delivery to *POK*.

(7) Acknowledge Message (IXZXIXAC) from POK to MCR

```
⋮
acknow   IXZXIXAC GROUPTOKEN=POKGTOKE,MSGTOKEN=MTOKEN,     X
         USERRC=USERRC,...
⋮
USERRC    DC  F'0'
⋮
```

Member *POK* acknowledges the ASYNCACK message from member *MCR*. *POK* uses the message token returned in *MTOKEN* to identify the message being acknowledged. The user return code (*USERRC*) provides a return code to the sender of the original message (member *MCR*).

8 Exit IXZXIT01 is invoked on member *POK* to modify the acknowledgement.

(9) Receive Acknowledgement (IXZXIXRM) from POK to MCR

```
⋮
receive  IXZXIXRM MBOXNAME=MCR_MBOX,GROUPTOKEN=MCRGTOKE,   X
         MSGTOKEN=MTOKEN,DATA=ACKMSG,DATALEN=DLEN,...
⋮
MTOKEN    DS  XL8
ACKMSG    DC  A(0)
DLEN      DC  F'0'
⋮
```

The message is sent to member *MCR*. The POST exit routine posts member *MCR*, which then receives the acknowledgement in *MCR*'s *MAILBOX*. JES XCF places the address of the acknowledgement in field *ACKMSG*. The message token, which member *POK* used to associate the acknowledgement with the message, is returned in field *MTOKEN*.

10 Exit IXZXIT02 is invoked on member *MCR* to modify the acknowledgement before delivery to *MCR*.

(11) Acknowledge Message (IXZXIXAC) from POK to MCR

```
⋮
acknow   IXZXIXAC GROUPTOKEN=MCRGTOKE,MSGTOKEN=MTOKEN,...
⋮
MTOKEN    DS  XL8
⋮
```

Member *MCR* acknowledges the acknowledgement from *POK*. *MCR* uses the message token returned in field *MTOKEN* to identify the acknowledgement that is being acknowledged.

In this example, the two JES members have completed their communication and each could detach from the JES XCF group. Remember, a detach is accomplished only through exit IXZXIT03, and exit IXZXIT03 gets control only during JES initialization and termination. This detach code is not called while the JES address space remains active.

(C) Exit IXZXIT03 (Attach/Detach) for MCR

```
⋮
detach    IXZXIXDT GROUPTOKEN=MCRGTOKE,RTNCODE=RTNCODE,   X
          RSNCODE=RSNCODE
⋮
RTNCODE   DC  F'0'
RSNCODE   DC  F'0'
⋮
```

Member *MCR* detaches from JES XCF group *IBMVSLAB*.

(D) Exit IXZXIT03 (Attach/Detach) for POK

```
⋮
detach    IXZXIXDT GROUPTOKEN=POKGTOKE,RTNCODE=RTNCODE,   X
          RSNCODE=RSNCODE
⋮
RTNCODE   DC  F'0'
RSNCODE   DC  F'0'
⋮
```

Member *POK* detaches from JES XCF group *IBMVSLAB*.

IXZXIT01 - Transport Exit

Exit IXZXIT01 enables an installation to view, modify, add to, or reroute a message prior to the message being delivered to the receiving member. This exit gets control for IXZXIXSM (send message) and IXZXIXAC (acknowledge message) macro invocations.

Installing the Exit Routine

To install IXZXIT01, be certain it is named IXZXIT01 and is linkedited into an authorized library in the LNKLST concatenation. Define the exit as follows:

- AMODE 31
- Reentrant.

For general instructions on installing MVS exits, refer to *z/OS MVS Installation Exits*. For information about the LNKLST concatenation and the LNKLSTxx parmlib member refer to *z/OS MVS Initialization and Tuning Reference*.

Exit Routine Environment

The environment at the time the exit routine receives control is as follows:

Address space
JES XCF address space

AMODE
31

ASC mode
Access register (AR)

Authorization
Supervisor state and key 1

Cross memory mode
Any PASN, any HASN, any SASN

Dispatchable unit mode
Task

Interrupt status
Enabled for I/O and external interrupts

JES environments
JES XCF

Locks held
None

RMODE
ANY

Storage
Pageable, key 1

Exit Recovery

JES XCF provides recovery for the exit routine.

If exit IXZXIT01 returns a non-zero return code or ends abnormally, control is returned to the caller, and IXZXIT01 is **not** called again for the life of the currently active attachment between this member and JES XCF.

Exit routine processing

Exit IXZXIT01 receives control under the same task that issues the IXZXIXSM or IXZXIXAC macro. This exit receives control after a message packet has been created but before that packet is sent to its target

member (specified by the IXZXIXSM macro) and when the acknowledging member replies to the originating system for all acknowledgements (REQTYPE= SYNC | ASYNC | ASYNCACK). Although REQTYPE=COMM messages are also acknowledged to JES XCF, exit IXZXIT01 is not called for that purpose.

The data passed to this exit is in a data space; therefore, the data can be accessed through an AR/GPR pair. The exit receives control in access register address space control (AR ASC) mode. AR1 contains the ALET of the data space containing the exit parameter list (mapped by IXZ$XPL) passed to the exit; GPR1 contains the address of IXZ$XPL within the data space.

IXZXIT01 communicates with its caller through the parameter list mapped by IXZ$XPL. By setting flag bits and changing data fields in the parameter list, exit IXZXIT01 can:

- Change the destination of a message
- Reroute the related message acknowledgment
- Change message content
- Add additional data to the message by appending one or more message extents.

The path from the exit caller to the message data includes the following links as shown in Figure 13 on page 44:

1. An installation exit or JES sends (through macro IXZXIXSM) or acknowledges (through macro IXZXIXAC) a message
2. When JES XCF invokes IXZXIT01 in the JES XCF address space, AR1 contains the ALET of the data space containing the parameter list, and GPR1 contains the address of the parameter list within the address space
3. Mapping IXZ$XPL contains data common to all of the exits, associated exit-specific data, and a pointer to the message data.

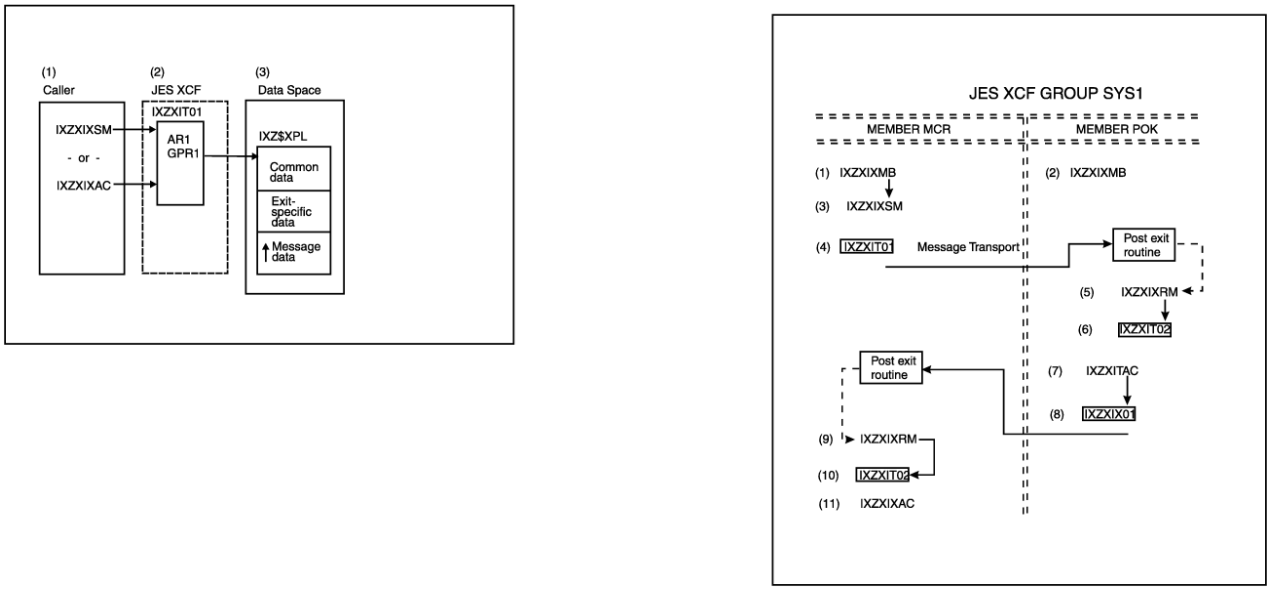


Figure 13. Access path from exit caller to message data

Changing a Message Destination

The message destination is determined by the following parameters on the IXZXIXSM macro:

- XCF member name (MEMBER= parameter)
- Mailbox name (MBOXNAME= parameter).

By changing either of these names, you can redirect the message to its appropriate destination. That is, to change the MEMBER= specification, change XIT01_DXCFMEMBER in IXZ$XPL. To change the

MBOXNAME= specification, change XIT01_DXCFMAILBOX in IXZ$XPL. Note that the message cannot be routed outside this JES XCF group; communication is limited to intra-group communication only.

Once exit IXZXIT01 makes the address change, it also must set the message destination change flag, *XIT01_DEST_UP*, in IXZ$XPL to inform JES XCF that the address is changed, and thus direct JES XCF to use the new address.

Rerouting an Acknowledgement

The message acknowledgement destination is determined by the REQMBOX= parameter on the IXZXIXSM macro. By changing this name (that is, by changing the value of XIT01_SXCFMAILBOX in IXZ $XPL), you can redirect the destination of an acknowledgement message. Once exit IXZXIT01 makes the address change, it also must set the acknowledgement destination change flag, *XIT01_SOURCE_UP*, in IXZ$XPL to inform JES XCF that the address is changed, and thus direct JES XCF to use the new acknowledgement address.

Changing the Message Content

Message data is presented to exit IXZXIT01, through mapping macro IXZ$XPL, by a pair of fields:

- XIT01_MESSAGE contains the address of the data
- XIT01_MESSAGE_LEN contains the length of that data.

The data is transported as presented unless your exit routine changes that data. To change the data:

1. Obtain storage to contain the new copy of the data

   This storage must be obtained from subpool 0. That is, **do not** specify a subpool. (IBM recommends using the STORAGE macro rather than $GETMAIN (for JES2) or AGETMAIN (for JES3) when obtaining this storage.)
2. Save the address of that storage in XIT01_MESSAGE_UPADDR
3. Save the length of that storage in XIT01_MESSAGE_UPLEN
4. Copy the data into the storage area and change the data as required
5. Set the indicator bit, XIT01_MESSAGE_UP, to inform JES XCF of the changes.

JES XCF transports the revised message rather than the original message data. Following message processing, JES XCF frees the storage obtained by this exit.

Appending message extents

You may need to add additional data to a message. If so, you can use exit IXZXIT01 to append data to a message. To do so:

1. Your exit must obtain, individually for each message extent, a contiguous block of storage from subpool 0 that is to contain header information and the message data to be added. (IBM recommends using the STORAGE macro rather than $GETMAIN (for JES2) or AGETMAIN (for JES3) when obtaining this storage).
2. Use the MSG_EXTENTS DSECT in mapping macro IXZ$XPL to map the following header data:
   - Eyecatcher - MSG_EXTENT_NAME (specify any meaningful, 8-character name)
   - Message length - MSG_EXTENT_LEN
   - Address of the data - MSG_EXTENT
   - The address of the next extent or 0 to indicate the end of the queue - NEXT_EXTENT
3. Queue this block of storage to the end of the single-threaded, null-terminated message extent queue. Use XIT01_NAME_EXTENTS in IXZ$XPL to point to the first element in the message extent queue.
4. Set indicator bit XIT01_EXTENTS on to indicate that an extent has been added.

As message extents are added to the data packet, the entire packet grows. If the data exceeds the maximum size as specified in XIT01_MAX_ADD, the original data is sent without the associated extents.

IXZXIT01 updates the XIT01_MESSAGE_UPLEN field in the exit-specific section of IXZ$XPL to provide the length of the message after exit IXZXIT01 modifies it.

Following message processing, JES XCF frees the storage obtained by this exit.

Figure 14 on page 46 shows the IXZ$XPL counters, XIT01_MESSAGE_UPLEN and XIT01_MAX_ADD.

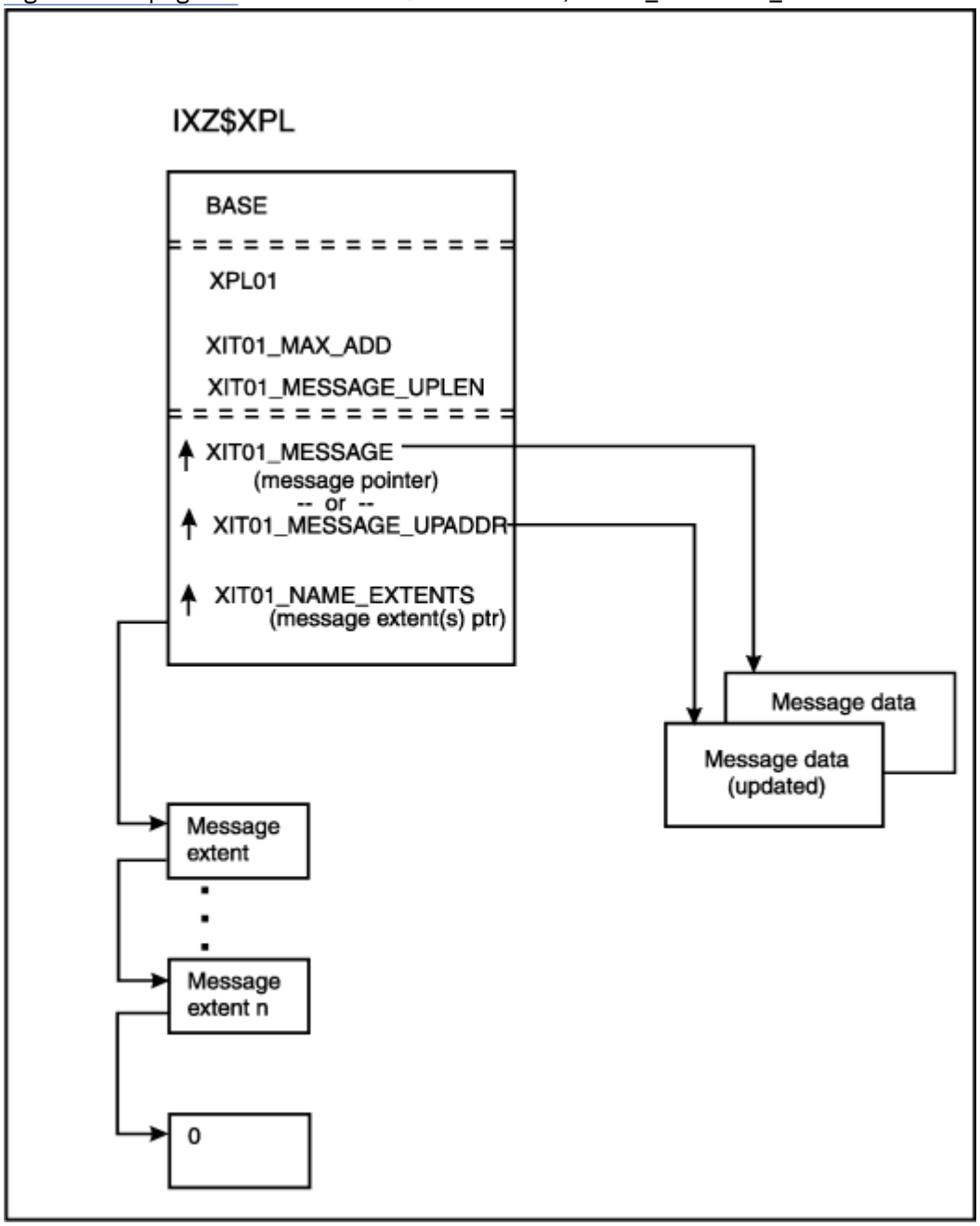


Figure 14. IXZ$XPL mapping of message data and extents

Processing Multi-Segment Messages

A multi-segment message is composed of two or more message segments. Each segment is passed individually to JES XCF through an IXZXIXSM (send message) macro invocation. Although JES XCF doesn't deliver the message to the receiver's mailbox until all message segments have been sent, the receiver retrieves the message from the mailbox one segment at a time. Because the segments are sent and retrieved individually, you can use this exit to change or add to segments individually. If you inadvertently make inconsistent changes to two or more segments, JES XCF uses the changes made to the first segment only.

Programming Considerations

- Avoid using services that can cause an MVS wait.
- If you add an extent to the message packet or alter the length of the message update, use XIT01_MAX_ADD in IXZ$XPL to reflect the message size change. XIT01_MAX_ADD contains the maximum number of bytes that can still be added to a message packet. Be certain to decrease its size as the message packet grows.

Entry Specifications

When IXZXIT01 receives control, GPR1 and AR1 point to the parameter list which is mapped by macro IXZ$XPL.

Registers at Entry

When exit IXZXIT01 receives control, the general purpose registers (GPRs) contain the following information:

Register
Content

0
Does not contain any information for use by this routine

1
Address of the IXZ$XPL parameter list

2-13
Do not contain any information for use by this routine

14
Return address

15
Entry point address

When exit IXZXIT01 receives control, the access registers (ARs) contain the following information:

Register
Content

0
Does not contain any information for use by this routine

1
ALET to access the IXZ$XPL parameter list

2-13
Do not contain any information for use by this routine

14
Primary ALET

15
Primary ALET

Parameter list contents

For a description of IXZ$XPL, see *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Sections in the parameter list are:

- A base section
- An exit-specific section
- The message content.

Base Section

The base section is the same for all exits. It specifies:

- Under which JES the exit was called
- The version of the parameter list presented to the exit
- The size of the base section
- The total size of the parameter list
- Optionally, a pointer to a shared data area created by exit IXZXIT03.

  When exit routine IXZXIT03 receives control, it has the option to create a data area. If exit routine IXZXIT03 creates the data area, exits IXZXIT01 and IXZXIT02 can use the data area to share data such as constants, pointers, and so forth. If exits IXZXIT01 and IXZXIT02 need to share a data area, your installation routine is responsible for the creation of this data area and its content.

Exit-Specific Section

The content of the exit-specific section varies depending upon the exit that has been invoked. The exit-specific section for exit IXZXIT01 provides:

- Destination address
- Source address
- Message length
- Pointer to the message content.
- Flags to indicate the following changes made in the installation exit:
  - Destination change
  - Source change (affecting which mailbox receives the acknowledgement)
  - Message change
  - Message extent additions
  - The message content.

Message Extent Section

You can optionally add more data to the message in a message extent, which is the last section of the message packet. The message extent section includes:

- An 8-character name (used to identify the message extent to the exit)
- The message extent length
- A pointer to the message extent text
- A pointer to the next message extent

Return Specifications

When exit IXZXIT01 returns control to JES XCF, it must return control to the address that was specified in register 14 at the time the exit was entered.

Registers at Exit

On return from exit processing, the contents of the general purpose registers (GPRs) must be:

Register
Content

0-14
The routine must restore the contents to what they were when the routine received control

15
Return code

On return from exit processing, the contents of the access registers (ARs) must be:

Register
Content

0-15
The routine must restore the contents to what they were when the routine received control

Return Code Meanings

When IXZXIT01 returns control to JES XCF, IXZXIT01 must provide one of the following return codes.

Return Code
Meaning

0
Processing was successful

4
Processing failed. This exit will **not** be invoked again for the life of this JES XCF attachment.

Coded Example of Exit Routine

A sample exit IXZXIT01 is provided in SYS1.SAMPLIB member IXZEX01A. The sample code illustrates coding techniques only; it **does not** perform actual function so that you can use it as written.

IXZXIT02 - Receive Exit

Exit IXZXIT02 enables an installation to view or modify a message before it is retrieved from a mailbox. The exit can also retrieve any message extents that IXZXIT01 might have added. This exit gets control for IXZXIXRM (receive message) macro invocations.

Installing the Exit Routine

To install IXZXIT02, be certain it is named IXZXIT02 and is linkedited into an authorized library in the LNKLST concatenation. Define the exit as follows:

- AMODE 31
- Reentrant.

For general instructions on installing MVS exits, refer to *z/OS MVS Installation Exits*. For information about the LNKLST concatenation and the LNKLSTxx parmlib member, refer to *z/OS MVS Initialization and Tuning Reference*.

Exit Routine Environment

The environment at the time the exit routine receives control is as follows:

Address space
JES XCF address space

AMODE
31

ASC mode
Access register (AR)

Authorization
Supervisor state and key 1

Cross memory mode
PASN not = HASN not = SASN PASN≠HASN≠SASN

Dispatchable unit mode
Task

Interrupt status
Enabled for I/O and external interrupts

JES environments
JES XCF

Locks held
None

RMODE
ANY

Storage
Pageable, key 1

Exit Recovery

When JES XCF invokes the exit routine, it also provides recovery for the exit routine. If the exit routine abnormally terminates, the message is lost and the issuer of the IXZXIXRM macro receives a return code of 8. IXZXIT02 is not invoked again for the life of the currently active attachment between this member and JES XCF.

Exit routine processing

Exit IXZXIT02 receives control under the same task that issues the IXZXIXRM macro to receive a message.

The data passed to this exit is in a data space; therefore, the data can be accessed only through an AR/GPR pair. The exit receives control in AR ASC mode. AR1 contains the ALET of the data space containing the parameter list IXZ$XPL; GPR1 contains the address of IXZ$XPL within the address space.

Exit IXZXIT02 communicates with its caller through the parameter list mapped by IXZ$XPL. By setting flag bits and changing data fields in the parameter list, exit IXZXIT02 can:

- Reroute an acknowledgement
- Change message content
- Retrieve any message extents that exit IXZXIT01 appended.

The path from the exit caller to the message data includes the following links as shown in Figure 15 on page 51:

1. An installation exit of JES issues a receive request (through the IXZXIXRM macro) to JES XCF
2. When JES XCF invokes IXZXIT02 in the JES XCF address space, AR1 contains the ALET of the data space containing the parameter list, and GPR1 contains the address of the parameter list within the data space
3. Mapping IXZ$XPL contains data common to all the exits, associated exit-specific data, and pointers to the message data and message extents.

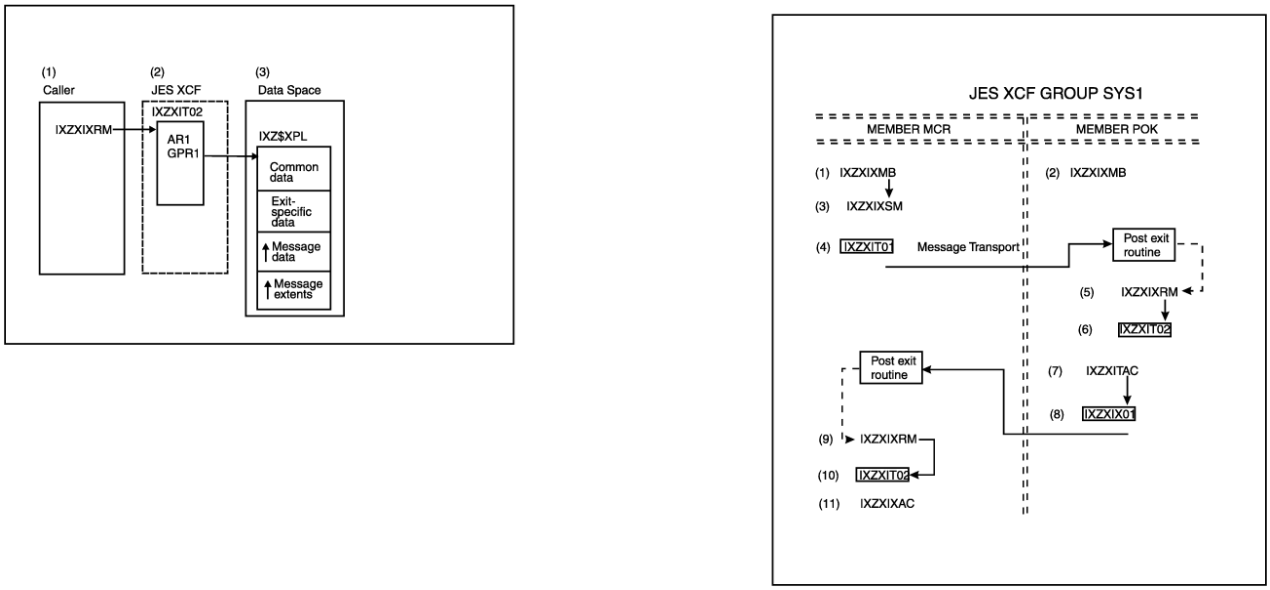


Figure 15. Access path from exit caller to message data

A message is composed of one or more message segments. Each segment is passed individually to JES XCF through an IXZXIXSM (send message) macro invocation. Although JES XCF does not deliver the message until all message segments are sent, the receiver retrieves the message from the mailbox one segment at a time. Because the segments are sent and retrieved individually, you can use this exit to change individual message segments. Exit IXZXIT02 must extract all message extents associated with each message segment or the extents will be lost.

Rerouting an Acknowledgement for ASYNCACK Messages

The message acknowledgement destination can be changed for ASYNCACK messages only. The sender codes the REQMBOX= parameter on the IXZXIXSM macro to specify the initial mailbox. By changing XIT02_SXCFMAILBOX in IXZ$XPL, you can redirect an acknowledgement message. Note that the message cannot be routed outside the JES XCF group or to a different member within the group. Once exit IXZXIT02 makes the address change, it also must set the acknowledgement destination change flag, *XIT02_SOURCE_UP*, in IXZ$XPL to inform JES XCF that the address is changed, and thus direct JES XCF to use the new acknowledgement address.

Changing the Message Content

Message data is presented to exit IXZXIT02, through mapping macro IXZ$XPL, by a pair of fields:

- XIT02_MESSAGE provides a pointer to the message data
- XIT02_MESSAGE_LEN provides the length of that data.

To change the data, your exit routine must:

1. Obtain storage to contain the new copy of the data

   This storage must be obtained from subpool 0. That is, **do not** specify a subpool. (IBM recommends using the STORAGE macro rather than $GETMAIN (for JES2) or AGETMAIN (for JES3) when obtaining this storage.)
2. Save the address of that storage in XIT02_MESSAGE_UPADDR
3. Save the length of that storage in XIT02_MESSAGE_UPLEN
4. Update XIT02_MAX_ADD in IXZ$XPL to reflect the message size change. XIT02_MAX_ADD contains the maximum number of bytes that can still be added to a message packet. Be certain to decrease its size as the message packet grows.
5. Copy the data into the storage and change the data as required
6. Set the indicator bit, XIT02_MESSAGE_UP, in IXZ$XPL to inform JES XCF of the changes.

JES XCF delivers the revised message rather than the original message data. Following message processing, JES XCF frees the storage obtained by this exit.

Retrieving an Appended Message Extent

Messages that were originally sent to the receiving member may have been intercepted by exit IXZXIT01 and appended with additional data. Unless exit IXZXIT02 retrieves the extents from the extent queue, any extents added to the original message are lost. Such message extents are not automatically propagated beyond exit IXZXIT02. Therefore, exit IXZXIT02 must search for and match the message extent name on the queue of extents. XIT02_NAME_EXTENTS in the exit-specific segment of IXZ$XPL points to this queue.

Programming Considerations

Do not use services that can cause an MVS wait.

Entry Specifications

When IXZXIT02 receives control, GPR1 and AR1 point to the parameter list, which is mapped by macro IXZ$XPL.

Registers at Entry

When exit IXZXIT02 receives control, the general purpose registers (GPRs) contain the following information:

Register
Content

0
Does not contain any information for use by this routine

1
Address of the IXZ$XPL parameter list

2-13
Do not contain any information for use by this routine

14
Return address

15
Entry point address

When exit IXZXIT02 receives control, the access registers (ARs) contain the following information:

Register
Content

0
Does not contain any information for use by this routine

1
ALET to access the IXZ$XPL parameter list

2-13
Do not contain any information for use by this routine

14
Primary ALET

15
Primary ALET

Parameter list contents

For a description of IXZ$XPL, see *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Sections in the parameter list are::

- A base section
- An exit-specific section
- The message content
- A message extent section.

Base Section

The base section is the same for all exits; it specifies:

- Under which JES the exit was called
- The version of the parameter list presented to the exit
- The size of the base section
- The total size of the parameter list
- Optionally, a pointer to a shared data area created by exit IXZXIT03.

When exit routine IXZXIT03 receives control, it has the option to create a data area. If exit routine IXZXIT03 creates the data area, exits IXZXIT01 and IXZXIT02 can use the data area to share data such as constants, pointers, and so forth. If exits IXZXIT01 and IXZXIT02 need to share a data area, your installation routine is responsible for the creation of this data area and its content.

Exit-Specific Section

The content of the exit-specific section varies depending upon the exit that has been invoked. The exit-specific section for exit IXZXIT02 provides:

- Destination address
- Source address
- Message length
- Pointer to the message content
- Flags to indicate the following changes made in the installation exit:
  - Source change (affecting which mailbox receives the acknowledgement)
  - Message change

Message Extent Section

If exit IXZXIT01 appended additional data to the message in the form of a message extent, the message extent queue becomes the last section of the message packet. Note that each segment of a multi-segment message can have its own message extent queue. The message extent section includes:

- An 8-character name (used to identify the message extent to the exit)
- The message extent length
- A pointer to the message extent text
- A pointer to the next message extent.

Return Specifications

When exit IXZXIT02 returns control to JES XCF, it must return control to the address that was specified in register 14 at the time the exit was entered.

Registers at Exit

On return from exit processing, the contents of the general purpose registers (GPRs) must be:

Register
Content

0-14
The routine must restore the contents to what they were when the routine received control

15
Return code

On return from exit processing, the contents of the access registers (ARs) must be:

Register
Content

0-15
The routine must restore the contents to what they were when the routine received control

Return Code Meanings

When IXZXIT02 returns control to JES XCF, IXZXIT02 must provide one of the following return codes.

Return Code
Meaning

0
Processing was successful

4
Processing failed. This exit will **not** be invoked again for the life of this JES XCF attachment.

Coded Example of Exit Routine

A sample exit IXZXIT02 is provided in SYS1.SAMPLIB member IXZEX02A. The sample code illustrates coding techniques only; it **does not** perform actual function so that you can use it as written.

IXZXIT03 - Attach/Detach Exit

Exit IXZXIT03 receives control during JES initialization and termination.

During JES initialization, exit IXZXIT03 enables an installation to:

- Obtain a storage area during attach processing and make that area available to exits IXZXIT01 and IXZXIT02.
- Invoke the attach macro IXZXIXAT. (This is necessary only if your installation experiences performance degradation when using the IBM-defined attachment.)

During JES termination, exit IXZXIT03 enables an installation to:

- Free the storage obtained during attach processing
- Invoke the detach macro service IXZXIXDT. (This is necessary only is you code IXZXIT03 to issue the IXZXIXAT macro during attach processing.)

Installing the Exit Routine

To install exit IXZXIT03, be certain it is named IXZXIT03 and is linkedited into an authorized library in the LNKLST concatenation. Define the exit as follows:

- AMODE 31
- Reentrant.

For general instructions on installing MVS exits, refer to *z/OS MVS Installation Exits*. For information about the LNKLST concatenation and the LNKLSTxx parmlib member, refer to *z/OS MVS Initialization and Tuning Reference*.

Exit Routine Environment

The environment at the time the exit routine receives control is as follows:

Address space
JES address space

AMODE
31

ASC mode
Primary

Authorization
Supervisor state

Cross memory mode
PASN=HASN=SASN

Dispatchable unit mode
Task

Interrupt status
Enabled for I/O and external interrupts

JES environments

- JES2 or JES3 Main Task
- JES2 or JES3 Subtask
- JES3 FSS

Locks held
None

RMODE
ANY

Storage
Pageable, key 1

Exit Recovery

JES XCF provides recovery for the exit routine. If recovery occurs, any resources held by the exit routine are not freed.

If the exit routine fails during attach processing, attach processing continues. Any storage area obtained by the exit, however, is unavailable to exits IXZXIT01 and IXZXIT02.

If the exit routine fails during detach processing, detach processing continues. Any storage area obtained by the exit during attach processing will not be freed.

Exit Routine Processing

IXZXIT03 receives control under the task that issued the IXZXIXAT or IXZXIXDT macro - usually the JES main task.

During attach processing, you can use exit IXZXIT03 to obtain a data area for IXZXIT01 and IXZXIT02 processing or to provide an additional attachment to JES XCF. IXZXIT03 receives control after the JES member has joined an XCF group but before either IXZXIT01 or IXZXIT02 receives control.

During detach processing, you can use exit IXZXIT03 receives control to free the data area obtained during attach processing or to drop the attachment to JES XCF. After IXZXIT03 receives control, neither IXZXIT01 nor IXZXIT02 receive control for the XCF group from which JES is detaching.

Exit IXZXIT03 communicates with JES XCF by setting flag bits and changing data fields in the parameter list mapped by the IXZ$XPL mapping macro. When exit IXZXIT03 receives control, it checks a bit in the parameter list to determine whether it is invoked for:

- An attach (the XIT03_CONNECT bit in IXZ$XPL is set)
- A detach (the XIT03_DISCONNECT bit in IXZ$XPL is set).

Attach processing

JES2 and JES3 attach processing are basically equivalent as shown in section A of Figure 17 on page 58. During JES initialization processing:

1. JES issues macro IXZXIXAT which causes exit IXZXIT03 to get control
2. IXZXIT03 first obtains storage from subpool 241 if storage is needed for the subsequent calls to IXZXIT01 and IXZXIT02.
3. Exit IXZXIT03 then optionally issues macro IXZXIXAT to establish a second attachment to the JES-defined group or to establish an installation-defined JES XCF group.
4. JES then continues initialization processing.

Attention:

Whenever you issue the IXZXIXAT macro, each invocation causes exit IXZXIT03 to get control. If you do not code IXZXIT03 to prevent subsequent recursive calls, JES XCF attempts to again attach the installation-defined member, and JES XCF produces a dump. Figure 16 on page 56 provides a graphic view of the IXZXIXAT call from JES initialization to attach JES XCF group N1 and then an installation-supplied call to attach installation group MINE. The figure also illustrates one coding technique that uses the set of CLC and BNE instructions to prevent an attempted reattach of installation group MINE. Be certain to add similar code if you attach an installation-defined group.

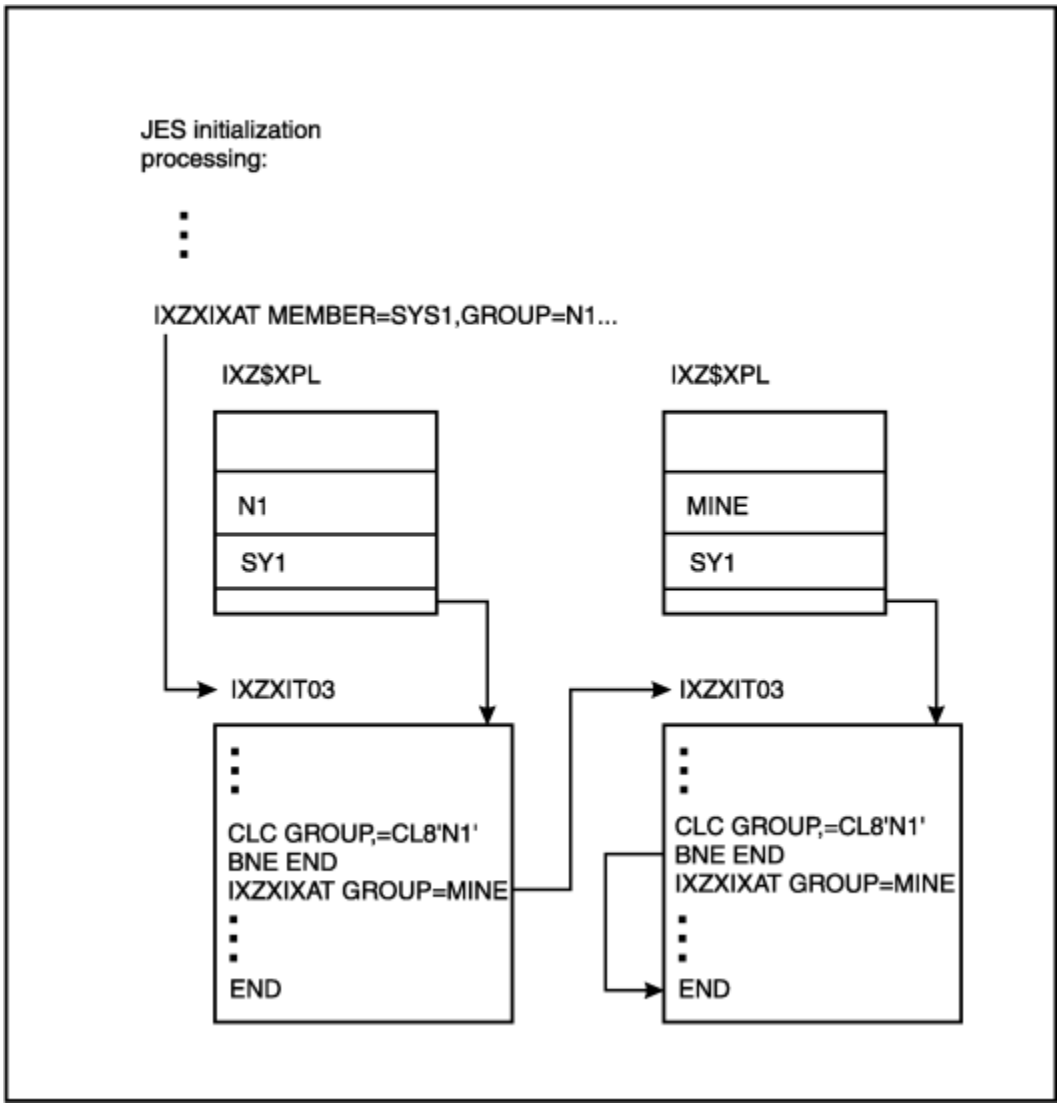


Figure 16. Example attach (IXZXIT03) code

Detach processing

JES2 and JES3 detach processing are quite different as shown in section B of Figure 17 on page 58. During JES termination processing, only JES2 provides clean up when terminating normally. Neither JES2 abnormal termination nor JES3 termination (except JES3 FSS, as stated in Figure 17 on page 58) frees either an installation-defined XCF group member or any members that were previously attached.

Therefore, for JES2 normal termination only, JES2 calls exit IXZXIT03 to issue the detach macro IXZXIXDT. IXZXIT03 first frees the storage that was obtained for exits IXZXIT01 and IXZXIT02. IXZXIT03 then detaches any installation-defined members. That is, if you use exit IXZXIT03 to call the IXZXIXAT macro during JES initialization, you must invoke IXZXIXDT to detach that member in IXZXIT03. JES2 then continues termination and clean-up processing. As discussed, the same requirement is not relevant to JES3.

Attention:

Whenever you issue the IXZXIXDT macro, each invocation causes exit IXZXIT03 to get control. If you do not code IXZXIT03 to prevent subsequent recursive calls, JES XCF attempts to again detach the installation-defined member, and JES XCF produces a dump. Be certain to add similar code to that shown in Figure 16 on page 56 if you detach an installation-defined group.

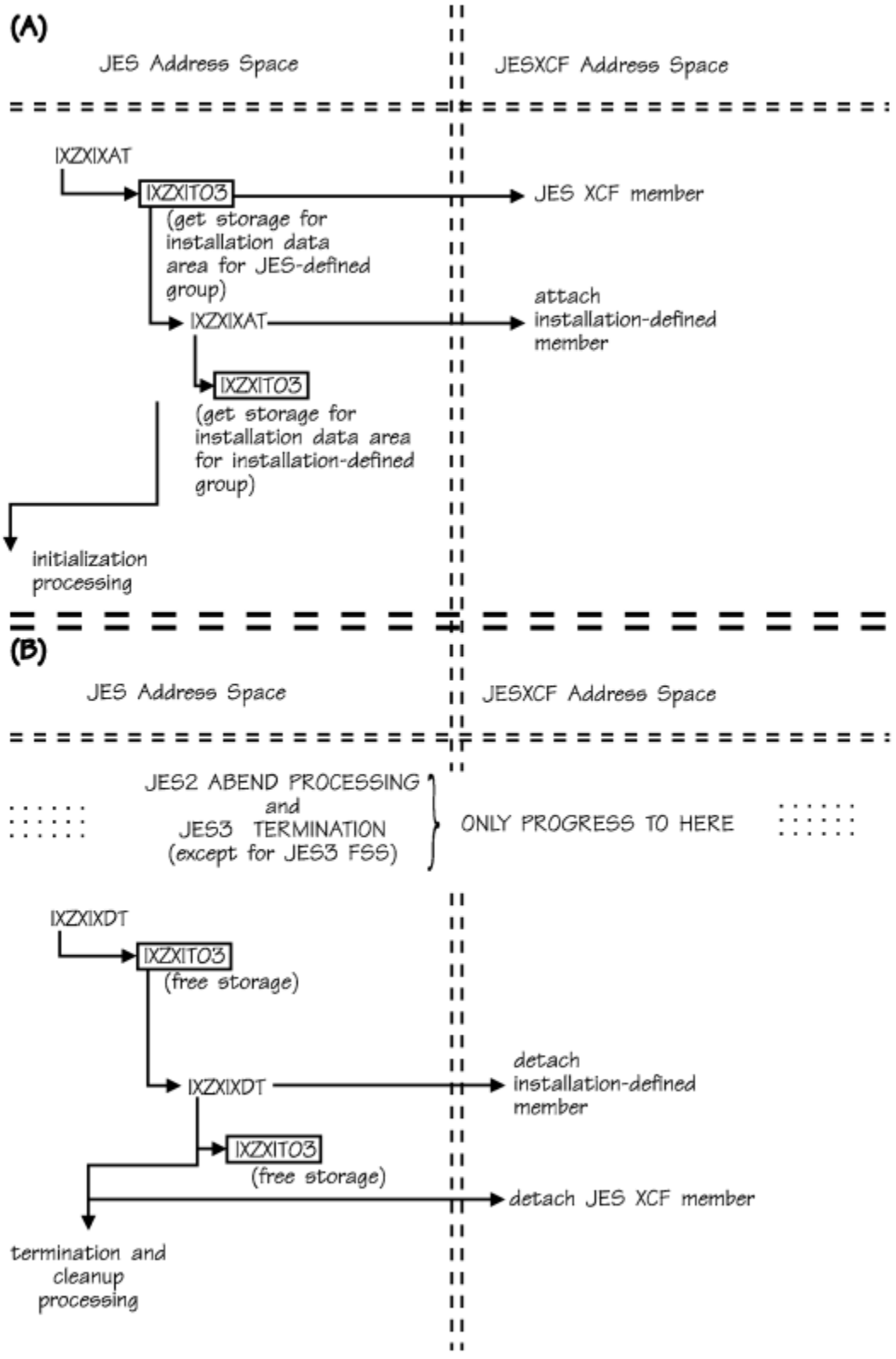


Figure 17. Exit IXZXIT03 attach/detach processing

Obtaining Storage During Attach Processing

The data area you obtain during attach processing with this exit must be accessible to both IXZXIT01 and IXZXIT02. Therefore, the data area must reside in common storage. Obtain storage for the data area from subpool 241 in key 1.

Use the data area only for the storage of pointers or constants that are to be shared by IXZXIT01 and IXZXIT02. Use the message extent capability of IXZXIT01 and IXZXIT02 to associate additional data with the messages rather than using this data area for that purpose.

Exit IXZXIT03 obtains a data area for every attach it processes. Therefore, as each JES member joins the XCF group, a separately addressable data area is created. If your installation requires that all JES XCF members on a single MVS system share a single installation data area, you must save the address of that data area the first time JES XCF invokes IXZXIT03. This address must then be accessed by IXZXIT03 on all subsequent invocations. You can do this through several techniques including use of name/token callable services (for example, IEANTCR). Refer to *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG* for a description of the name/token callable services and *z/OS MVS Programming: Authorized Assembler Services Guide* for information on using those services.

Freeing Storage During Detach Processing

During detach processing, IXZXIT03 gets control. At this call, be certain to release whatever storage the exit obtained during attach processing.

Example of Managing a Piece of Storage

Use the fields in the exit parameter list, IXZ$XPL, to manage the storage you obtain for exits IXZXIT01 and IXZXIT02 to use. For example:

1. Bits XIT03_CONNECT and XIT03_DISCONNECT, in the IXZXIT03-specific section of IXZ$XPL, determine whether this processing is part of attach or detach processing. (In this example, assume that XIT03_CONNECT is set to indicate attach processing.)
2. Obtain storage for exits IXZXIT01 and IXZXIT02.
3. Turn on the XIT03_INSTALL bit in the base section of IXZ$XPL to inform JES XCF that exit IXZXIT03 has obtained storage for IXZXIT01 and IXZXIT02.
4. Set the XIT03_INSTALLATION field in the IXZXIT03-specific section to the address of the storage.

When IXZXIT01 or IXZXIT02 receives control, the address of the data area is in the XPL_INSTALL_DATA field in the base section of IXZ$XPL.

Because IXZXIT03 obtained storage for the installation data area from common storage, the installation data area is addressable by all the JES XCF exits.

Programming Considerations

Do not use services that can cause an MVS wait.

Entry Specifications

When exit IXZXIT03 receives control, GPR1 points to the parameter list which is mapped by macro IXZ $XPL.

Registers on Entry

When exit IXZXIT03 receives control, the general purpose registers (GPRs) contain the following information:

Register
Content

0
Does not contain any information for use by this routine

1
Address of the IXZ$XPL parameter list

2-13
Do not contain any information for use by this routine

14
Return address

15
Entry point address

When exit IXZXIT03 receives control, the access registers (ARs) contain the following information:

Register
Content

0-15
Do not contain any information for use by this routine

Parameter list contents

For a description of IXZ$XPL, see *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Sections in the parameter list are:

- A base section
- An exit-specific section.

Base Section

The base section is the same for all exits. It specifies:

- Under which JES the exit was called
- The version of the parameter list presented to the exit
- The size of the base section
- The total size of the parameter list
- Optionally, a pointer to a data area created by IXZXIT03 during attach processing.

Exit-Specific Section

The content of the exit-specific section varies depending upon the exit that has been invoked. The exit-specific section for IXZXIT03 provides:

- A type-of-call bit (XIT03_CONNECT or XIT03_DISCONNECT) to inform IXZXIT03 of the type of call, that is, whether this processing is part of attach or detach processing. (IXZXIT03 should treat this bit as "read only".)
- Group name
- Member name.

Return Specifications

When exit IXZXIT03 returns control to JES XCF, it must return control to the address that was specified in register 14 at the time the exit was entered.

Registers at Exit

On return from exit processing, the contents of the general purpose registers (GPRs) must be:

Register
Content

0-15
The routine must restore the contents to what they were when the routine received control

On return from exit processing, the contents of the access registers (ARs) must be:

Register
Content

0-15
The routine must restore the contents to what they were when the routine received control

Return Code Meanings

If processing is successful, IXZXIT03 returns control to its caller without a return code.

Coded Example of Exit Routine

A sample exit IXZXIT03 is provided in SYS1.SAMPLIB member IXZEX03A. The sample code illustrates coding techniques only; it **does not** perform actual function so that you can use it as written.

Chapter 4. JES XCF macro reference

Use the IXZXIXAT (attach) and IXZXIXDT (detach) macros only in exit IXZXIT03. You can use the remaining JES XCF macros in standard JES exits and JES3 DSPs.

All macros (except IXZXIXAT and IXZXIXDT) should be packaged within standard JES exit routines, JES3 DSPs, or your own installation routines.

How to Read a Syntax Diagram

Syntax is described using the structure defined below.

- Read the syntax diagrams from left o right, from top to bottom, following the path of the line.

  The ►►— symbol indicates the beginning of a macro syntax.

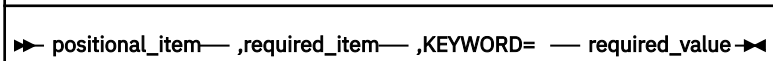
  The —► symbol indicates that the macro syntax is continued on the next line.

  The ►— symbol indicates that the macro syntax is continued from the previous line.

  The —►◄ indicates the end of the macro syntax.

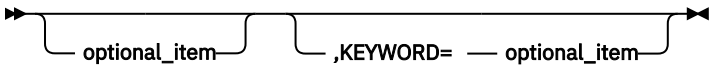
  A lower-case word indicates a variable.
- Required keywords (and their value) and required positioinal items appear on the horizontal line (the main path).

Required Items



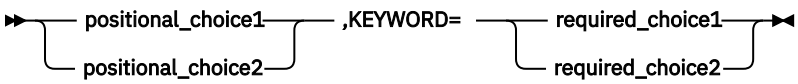
- Optional keywords (and their value) and posiitonal items appear below the main path.

Optional Items



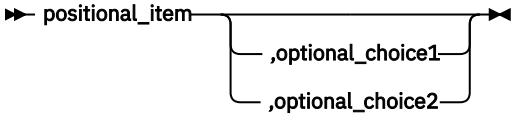
- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.

Required Choice



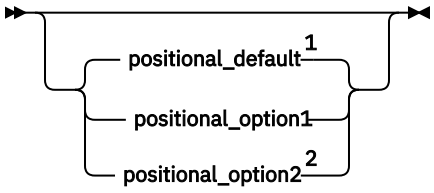
- If choosing one of the items is optional, the entire stack appears below the main path

Optional Choice



- If one of the items has a default, it appears above the main line (for that item) and the overriding choices will be shown on and/or below the main line. For POSITIONAL items the syntax is as follows:

Default Positional Item



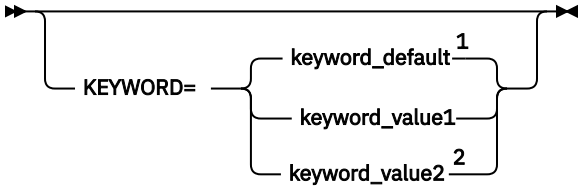
Notes:

[1] If a positional parameter is not specified, the positional default is used

[2] If a positional parameter is coded, then any one of the 3 positional parameters can be specified.

For KEYWORD items the syntax is as follows:

Default Keyword Item



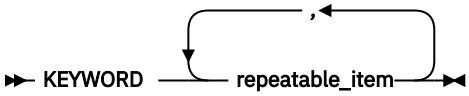
Notes:

[1] If KEYWORD is not coded, the keyword_default is used.

[2] If the KEYWORD is coded, then the keyword_default or the keyword value override MUST be coded.

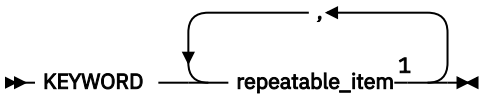
- An arrow returning to the left above the main line indicates an item that can be repeated indefinitely.

Indefinitely Repeated Item



- A repeat arrow with a syntax note indicates how many times this can be repeated.

Fixed Repeated Item

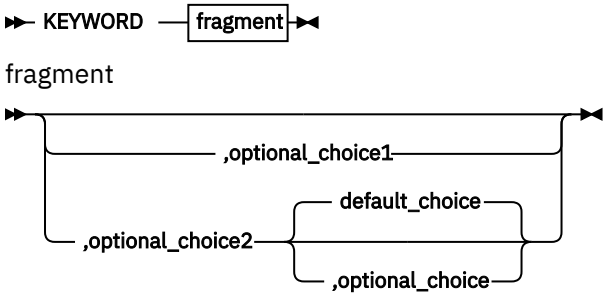


Notes:

[1] (1) Specify the <repeatable_item> 1 to n times.

- Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.

Fragment



Controlling Macro Expansion Printing

The ZPRINT global macro variable controls macro expansion printing. If you want to suppress macro expansion printing, set ZPRINT to 'NO' because the default is 'YES'. The following examples illustrate how to set the ZPRINT variable.

```
GBLC &ZPRINT;
&ZPRINT; SETC 'YES'
    -- or --
&ZPRINT; SETC 'NO'
```

IXZXIXAC - acknowledge receipt of a message

Use the IXZXIXAC macro to acknowledge that a message has been received and processed. You must issue IXZXIXAC to acknowledge the message after you issue the IXZXIXRM macro to receive it. Optionally, you can return to the sender a return code and one data segment along with the acknowledgement.

Refer to "Acknowledging Receipt of a Message (IXZXIXAC Macro)" on page 25 for a description of using IXZXIXAC to acknowledge messages.

Environment

The requirements for the caller are:

Table 8. Environment

Variable	Value
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Table 8. Environment (continued)	
Variable	**Value**
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXAC is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXAC is invoked.

Restrictions

Do not issue IXZXIXAC until JES initialization processing has established an attachment to the JES XCF group; that is, after IXZXIT03 processing has completed.

Input Register Information

Before issuing the IXZXIXAC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXAC returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as work registers by the system

2-13
Unchanged

14
Used as work registers by the system

15
Return code

When IXZXIXAC returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as a work register by the system

2-13
Unchanged

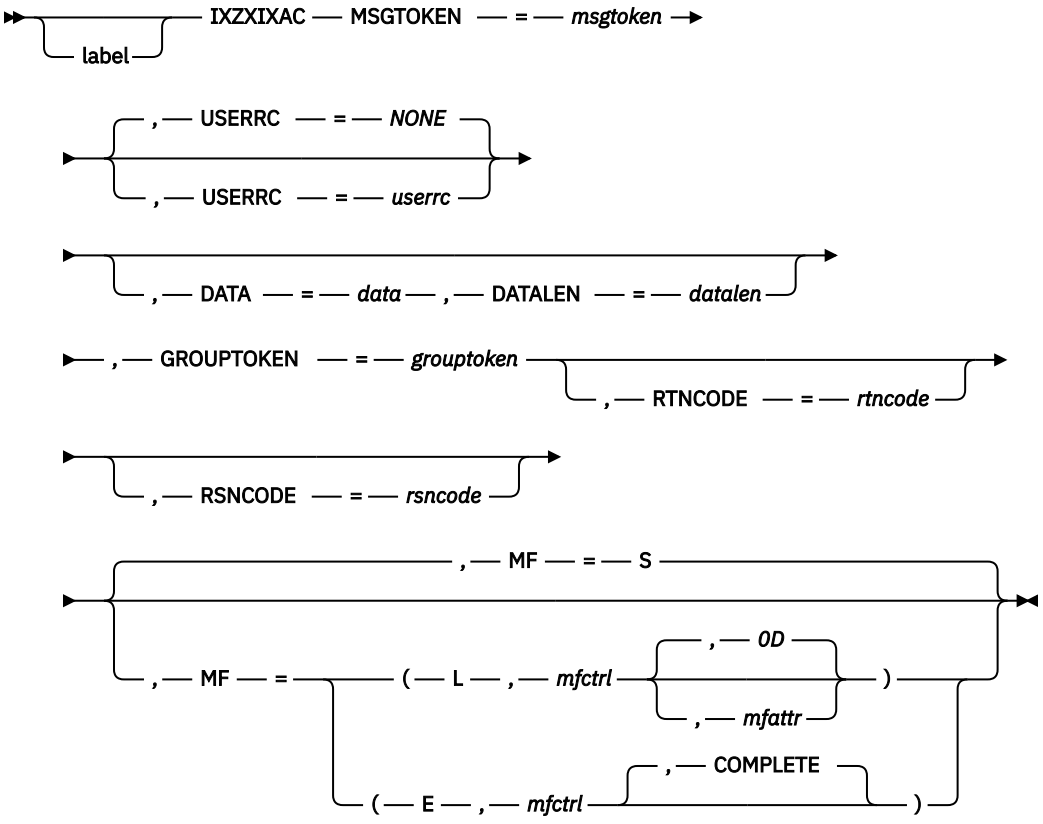
14,15
Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None

Syntax



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXAC macro invocation.

Default: no name

MSGTOKEN=msgtoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 64-bit input field that contains the message token returned from the message receive macro (IXZXIXRM).

Default: none

USERRC=userrc | NONE
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword input field that contains a user-specified return code that is passed back to the sender of the original message.

Default: NONE. If you do not specify a value for USERRC=, JES XCF returns an indication (in field RC_PROVIDED of the IXZYIXAC mapping macro) that a return code is not set.

DATA=data
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword input field that contains the address of the message data that is to be returned with the acknowledgement. If you specify DATA=, you must also specify DATALEN=.

Default: none

DATALEN=datalen
Specifies the name (RS-type) or address (in GPR2-GPR12) of a fullword input field that contains the length of the message data to be sent. The message length must be a positive decimal integer with a value of 0 to 60 (kilobytes). If you specify DATALEN=, you must also specify DATA=.

Default: none

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token. If you are using the JES XCF group token to communicate, refer to "Retrieving the JES XCF Group Token" on page 13 for the procedure for retrieving the JES group token. If you are using an installation-defined group token to communicate, use the group token returned in the field specified by GROUPTOKEN= on the IXZXIXAT macro following attach processing.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the in-line parameter list and invoke the desired service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr [3] 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the desired degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXAC service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and reason codes

When the IXZXIXAC macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. The message was acknowledged. **Action:** None
4	4	**Meaning:** Processing failed because the parameter list passed to the IXZXIXAC service contained an eyecatcher that was not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.
4	8	**Meaning:** Processing failed because the parameter list passed to the IXZXIXAC service had a version indicator that was not valid. **Action:** The module that called IXZXIXAC might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.
4	C	**Meaning:** Processing failed because the group token passed to the IXZXIXAC service was not valid. **Action:** Verify that the correct group token is being passed to IXZXIXAC.
4	10	**Meaning:** Processing failed because the IXZXIXAC macro was issued on a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	14	**Meaning:** Processing failed because the message token passed to the IXZXIXAC service is not valid. **Action:** Verify that the correct message token is being passed to IXZXIXAC.
4	18	**Meaning:** Processing failed because the parameter list passed to the IXZXIXAC service did not contain a DATALEN= specification. **Action:** Be certain that DATALEN= is specified when the DATA= parameter is used.
4	1C	**Meaning:** Processing failed because the parameter list passed to the IXZXIXAC service contains a DATALEN= value that is not valid. **Action:** Be certain that the DATALEN= value was not negative or too large. DATALEN= must be a positive decimal integer up to 60 kilobytes.

Return Code (hex)	Reason Code (hex)	Meaning and Action
8	0	**Meaning:** Exit IXZXIT01 failed. The acknowledgement being processed at the time IXZXIT01 failed is lost. **Action:** Review your IXZXIT01 code for coding errors, correct the error, and rerun the program that issues IXZXIXAC.
C	0	**Meaning:** Processing failed because the IXZXIXAC service ended abnormally. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

IXZXIXAT - Attach to a JES XCF Group

Use the IXZXIXAT macro to define and activate a JES as a member of an XCF group. Use this macro only when you are establishing a new XCF group for the JESes in your sysplex. IBM recommends that you establish your own group only when you send large amounts of data among members, and that data traffic is great enough to affect performance.

If you choose to use the JES-supplied attachment (that is, the JES XCF group), you do not need to attach to it using this macro. You need only retrieve the JES XCF group token to specify on any JES XCF macros you plan to use. For more information about deciding which XCF group to use, see "The JES-Defined Attachment to JES XCF" on page 13.

Refer to "Creating an Installation-Defined Attachment to JES XCF (IXZXIXAT Macro)" on page 14 for a description of using IXZXIXAT to attach to a JES XCF group.

Environment

The requirements for the caller are:

Table 9. Environment

Variable	Value
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXAT is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXAT is invoked.

Restrictions

If you do issue IXZXIXAT, IBM recommends the following **for each JES member** that is joining the XCF group:

- Issue IXZXIXAT during JES initialization.
- Call IXZXIXAT within exit IXZXIT03 only.
- Issue IXZXIXAT only once.

Input Register Information

Before issuing the IXZXIXAT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXAT returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXAT returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as work registers by the system

2-13
Unchanged

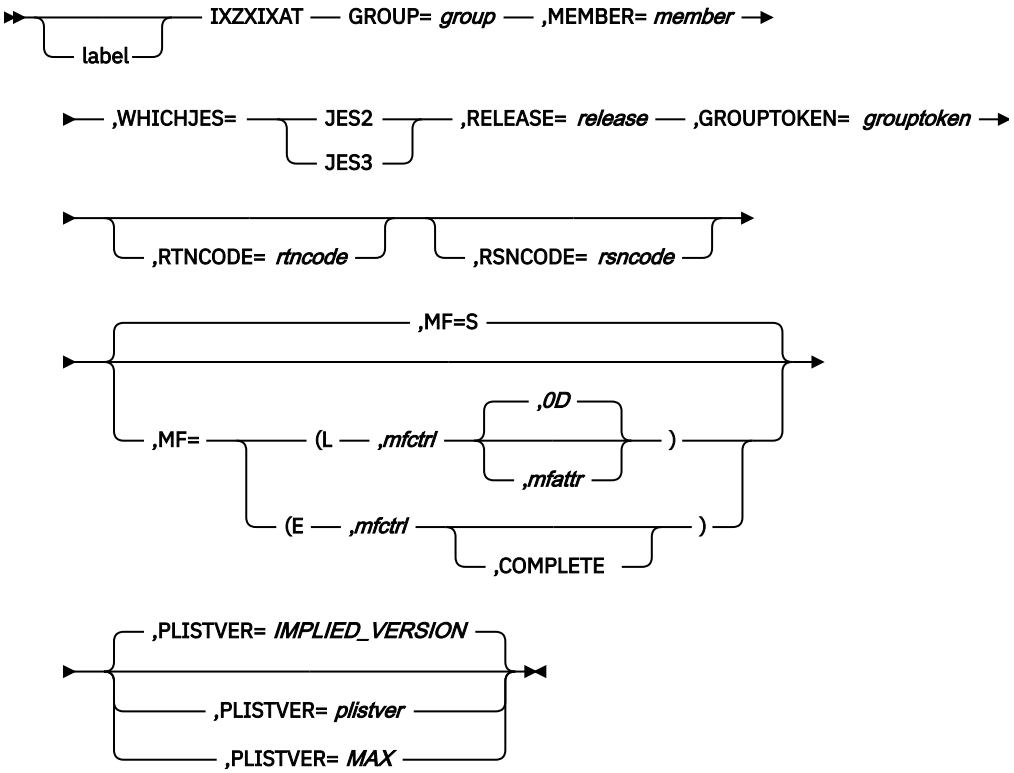
14,15
Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

IBM recommends the use of the JES-supplied attachment to JES XCF. If, however, you use JES XCF to send large amounts of data, you should assess your application's impact on JES performance. If you detect an unsatisfactory performance impact, use the IXZXIXAT macro to create your own attachment.

Syntax



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXAT macro invocation.

Default: no name

GROUP=group
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 1- through 8-character input field that contains the name of the XCF group the JES member is to join. This name must conform to the following naming standards for XCF groups: the name may consist of characters A-Z, 0-9, $, #, @, and must be left justified and padded on the right with blanks. Avoid names beginning with A-I, the character string *SYS*, or the reserved name *UNDESIG*. These are reserved for IBM use only.

Default: none

MEMBER=member
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the name of the XCF member that the JES member is to use when joining the XCF group. This name must conform to the following naming standards for XCF groups: the name may consist of characters A-Z, 0-9, $, #, @ and must be left justified and padded on the right with blanks. Avoid names beginning with A-I, the character string *SYS*, or the reserved name *UNDESIG*. These are reserved for IBM use only.

Default: none

WHICHJES=JES2 | JES3 | J3FSS | COMMON
Specifies under which JES the XCF group will run:

JES2
Indicates that the subtask runs under the JES2 subsystem.

JES3
Indicates that the subtask runs under the JES3 subsystem.

J3FSS
Indicates that the JES3 writer functional subsystem (FSS) is attaching.

COMMON
To allow the JESXCF group to communicate, the JESXCF component creates an XCF group during MVS initilization.

Default: none

RELEASE=release
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 8-character input field that contains the FMID for the JES release with which the subtask will run.

Default: none

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword output field that will contain the token that identifies the XCF group. If you use other JES XCF macros, use this token as input for those macros.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the in-line parameter list and invoke the desired service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the desired degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

PLISTVER=IMPLIED_VERSION | MAX | plistver
Use this input parameter to specify the version of the macro. See *z/OS MVS Programming: Sysplex Services Reference* for a description of the options available with PLISTVER.

This is the only parameter allowed on the list (L) form of this macro and determines which parameter list is generated.

IMPLIED_VERSION
Implies the lowest version which allows all the parameters specified on the macro call to be processed.

MAX
Specifies the parameter list will be created at the largest size currently supported. The size is subject to change. Therefore, use this specification only if your program can tolerate such change. Its use ensures that the list form of the parameter list is always sufficiently large to contain the parameters specified on the execute (E) form.

plistver
Specifies an optional input decimal value (2 through 5) that indicates the macro version.

Default: IMPLIED_VERSION

ABEND Codes

When control returns to your program, if you receive a return code of X'8', it indicates that processing failed because the IXZXIXAT service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and Reason Codes

When the IXZXIXAT macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. The subtask has been attached and the JES XCF group has been activated. **Action:** None
8	0	**Meaning:** Processing failed because the IXZXIXAT service ended abnormally. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.
8	4	**Meaning:** The system that made the request to attach to JESXCF is being partitioned out of the sysplex. All requests from this system to join any XCF group are permanently suspended. **Action:** See message IXZ0110E.

[Programming Interface Information] IXZXIXCL - System cleanup initiated indicator

Use the IXZXIXCL macro to allow JES to inform JES XCF that it initiated any cleanup actions that are required after a system in the sysplex failed. Through access to IXCYGEPL, JES monitors system event status. IXCYGEPL provides an indication that a system failed and provides the XCF system token of the failed system. This information and the group token allows JES XCF to identify the specific failed system within the XCF group.

See *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for a description of the IXCYGEPL mapping macro.

Environment

The requirements for the caller are:

Table 10. Environment

Variable	Value
JES environments:	• JES2 or JES3 main task • JES3 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXCL is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXCL is invoked.

Restrictions

Do not issue IXZXIXCL until JES3 initialization processing established an attachment to the JES XCF group, which is after IXZXIT03 processing completed.

Input register information

Before issuing the IXZXIXCL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output register information

When IXZXIXCL returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXCL returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as a work register by the system

2-13
Unchanged

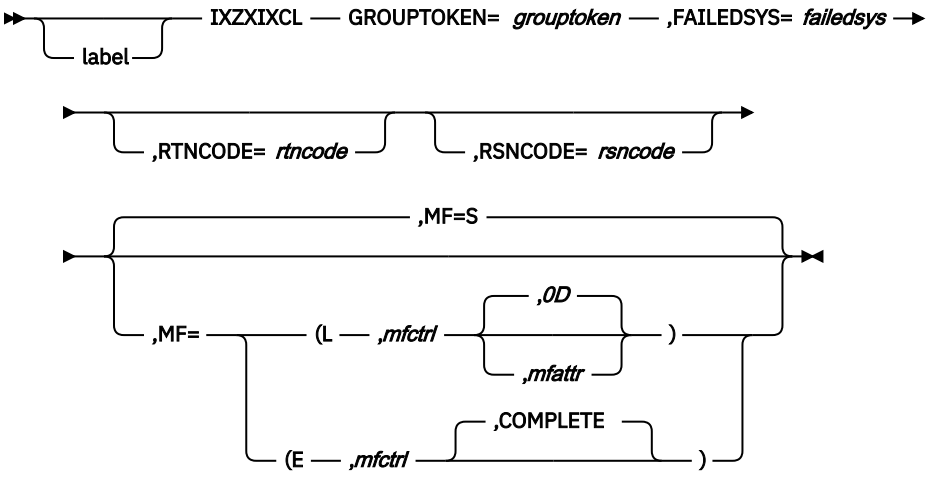
14,15
Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None

Syntax



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXCL macro invocation.

Default: no name

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token returned by the IXZXIXAT macro.

Default: none

FAILEDSYS=failedsys
Specifies the name (RS-type) of a required fullword input field or register (GPR2-GPR12) that contains the XCF system token of the failed system. Macro IXCYGEPL provides the token of the failed system.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro, which is to build the inline parameter list and invoke the wanted service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr |0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXCL service ended abnormally, and also received an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and reason codes

When the IXZXIXCL macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return code (hex)	Reason code (hex)	Meaning and action
0	0	**Meaning:** Processing was successful. JES XCF has queued the cleanup request for processing. **Action:** none
C	0	**Meaning:** Processing failed because the IXZXIXCL service ended abnormally. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

[End Programming Interface Information]

IXZXIXDT - Detach from a JES XCF Group

Use the IXZXIXDT macro to deactivate and remove (detach) a JES from an XCF group. Use this macro only when you have established your own XCF group for the JESes in the sysplex.

IBM recommends that once you have created an installation-defined attachment to JES XCF, keep that attachment active until the JES address space terminates. When the JES address space terminates, the system breaks the attachment and reclaims any resources that are held by that attachment.

Refer to “Detaching from a JES XCF Group (IXZXIXDT Macro)” on page 31 for a description of using IXZXIXDT to detach from a JES XCF group.

Environment

The requirements for the caller are:

Table 11. Environment

Variable	Value
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN

Table 11. Environment (continued)	
Variable	**Value**
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXDT is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXDT is invoked.

Restrictions

If you do issue IXZXIXDT, IBM recommends the following **for each JES member** that is leaving the XCF group.

- Issue IXZXIXDT during JES termination processing.
- Call IXZXIXDT within exit IXZXIT03 only.
- Issue IXZXIXDT only once.

Input Register Information

Before issuing the IXZXIXDT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXDT returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXDT returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as a work register by the system

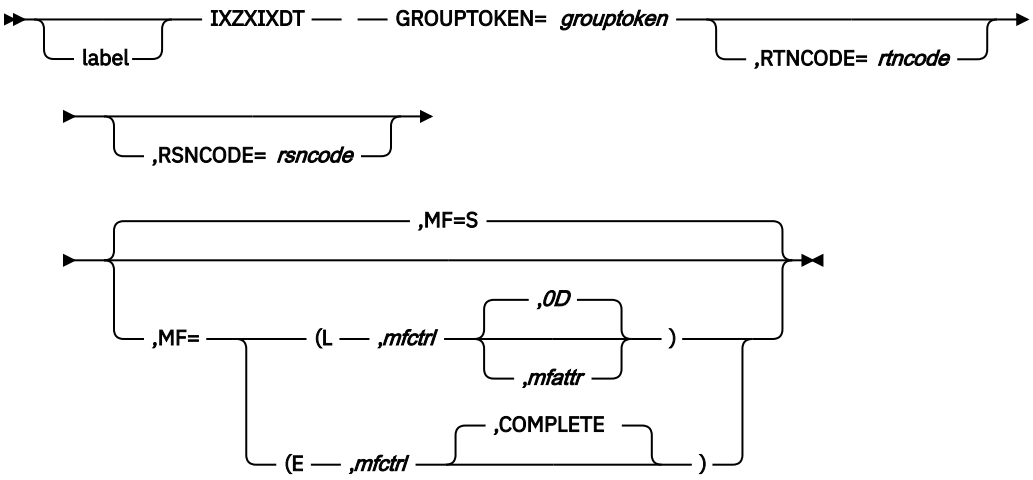
2-13
Unchanged

14,15
Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXDT macro invocation.

Default: no name

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token returned by the IXZXIXAT macro.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is to build the inline parameter list and invoke the wanted service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns to your program, if you receive a return code of X'8', it indicates that processing failed because the IXZXIXDT service ended abnormally, and an associated ABEND code is also received. Refer to *z/OS MVS System Codes* for a description of following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and Reason Codes

When the IXZXIXDT macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. JES has been detached from JES XCF and is no longer an active member of the corresponding XCF group. **Action:** None
8	0	**Meaning:** Processing failed because the IXZXIXDT service ended abnormally. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

IXZXIXIF - obtain information about members of an XCF group

Use the IXZXIXIF macro to obtain information about JES members of an XCF group. IXZXIXIF returns a record of information about the specific group member, and also information about the MVS system on which the member is running. Figure 8 on page 25 provides the JES XCF message mapping structure that shows the format in which JES XCF returns the requested information.

See "Obtaining information about JES members in the XCF group (IXZXIXIF macro)" on page 26 for a description of using IXZXIXIF to obtain JES XCF information.

Environment

The requirements for the caller are:

Table 12. Environment

Variable	Value
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	• If you specify GROUPTOKEN= – Supervisor state with PSW key 0 or 1 • If you specify GROUPNAME= – Supervisor state with PSW key 0 - 7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXIF is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXIF is invoked.

To map the member information record that IXZXIXIF returns, use mapping macro IXZYIXIF. IXZYIXIF is described in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary), and Figure 9 on page 27 provides a graphic view of its structure.

Restrictions

Do not issue IXZXIXIF until JES initialization processing established an attachment to the JES XCF group, which is after IXZXIT03 processing completed.

Input register information

Before issuing the IXZXIXIF macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output register information

When IXZXIXIF returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXIF returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as a work register by the system

2-13
Unchanged

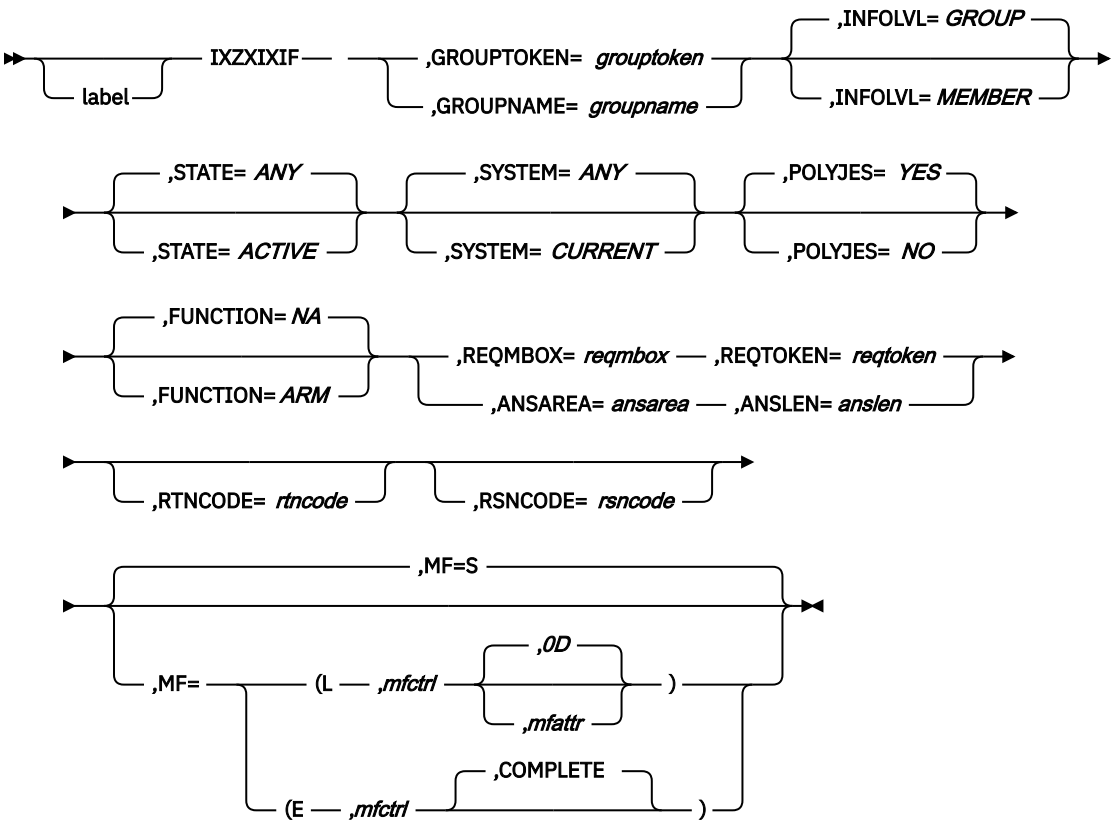
14,15
Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

Limit your use of IXZXIXIF because each IXZXIXIF invocation requires I/O to the couple data set. Requesting information using REQMBOX causes an asynchronous call to JES XCF, whereas using ANSAREA= and ANSLEN= causes a synchronous call. Therefore, use REQMBOX= to reduce performance impact.

Syntax



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXIF macro invocation.

Default: no name

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword input field that contains the group token. See “Retrieving the JES XCF Group Token” on page 13 for the procedure for retrieving the JES XCF group token. Use GROUPNAME= rather than GROUPTOKEN= if you intend to also specify INFOLVL=MEMBER or use any of the filtering parameters (FUNCTION=, POLYJES=, STATE=, and SYSTEM=). This parameter is mutually exclusive with the GROUPNAME= parameter; however, you must specify either GROUPTOKEN= or GROUPNAME=.

Default: none

GROUPNAME=groupname
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 8-character input field that contains the name of the JES XCF group. Use GROUPNAME= if you intend to also specify INFOLVL=MEMBER or use any of the filtering parameters (FUNCTION=, POLYJES=, STATE=, and SYSTEM=). This parameter is mutually exclusive with the GROUPTOKEN= parameter; however, you must specify either GROUPNAME= or GROUPTOKEN=.

Default: none

INFOLVL=GROUP | MEMBER
Specifies the type of information requested.

GROUP
Specifies a request for information about the entire group.

MEMBER
Specifies a request for information about a member of the XCF group.

If you specify INFOLVL=MEMBER, you must also specify GROUPNAME=.

Default: GROUP

REQMBOX=reqmbox
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 16-character input field that specifies the name of a mailbox into which JES XCF can place the requested information. The name can contain the characters: A-Z, 0-9, $, @, #, and embedded and trailing blanks only. The mailbox is associated with the member whose group token is specified by the GROUPTOKEN= keyword.

Default: none

REQTOKEN=reqtoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of an 8-character output field into which a token can be placed. You can use the token to match the information request with the response returned to the mailbox specified by the REQMBOX= parameter. If you specify REQMBOX=, you must also specify REQTOKEN=.

Default: none

ANSAREA=ansarea
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 4-byte input field that contains the address of a buffer used to contain the data that JES XCF returns. If the area passed is too small, return and reason codes of 4 are returned, and the system updates the ANSLEN= value with the required length.

Default: none

ANSLEN=anslen
Specifies the name (RS-type) or address (in GPR2-GPR12) of a fullword input/output field that specifies the length of the ANSAREA buffer passed to the system. If the buffer is not large enough, the system updates the ANSLEN value with the length required, and return and reason codes of 4 are returned to the caller. The caller can then obtain storage of the specified size and again try to obtain the requested information.

Default: none

STATE=ANY | ACTIVE
Optionally specifies whether the information request is to be filtered based on member state.

If you specify STATE=, you must also specify GROUPNAME=.

ANY
Indicates that JES XCF provides member information for members in *any* state.

ACTIVE
Indicates that JES XCF provides member information for active members only.

Default: ANY

SYSTEM=ANY | CURRENT
Optionally specifies whether the information request is filtered based on the JES member and MVS system from which this request was issued.

If you specify SYSTEM=, you must also specify GROUPNAME=.

ANY
Indicates that JES XCF provides member information for members on *any* MVS system in the sysplex.

CURRENT
Indicates that JES XCF provides member information for members on the *current* MVS system only; that is, the JES XCF members currently running on the same MVS system as the member that issued the IXZXIXIF macro request.

Default: ANY

POLYJES=YES | NO
Optionally specifies whether the information request is filtered to limit information from a single JES2 member from MVS systems running multiple instances of JES2 system.

If you specify POLYJES=, you must also specify GROUPNAME=.

YES
Indicates that JES XCF provides member information for all members (including secondary JES2 systems running in a poly-JES environment) running on each MVS system in the sysplex.

NO
Indicates that JES XCF selects and provides member information for only a single member on each MVS system in the sysplex. JES XCF selects the primary member unless the primary member is not active or the primary does not match the other filters (FUNCTION=, STATE=, and SYSTEM=). If JES XCF cannot return information for a primary member, it returns information for a secondary member if one matches the other filters.

Note: Only JES2 supports a poly-JES environment; this parameter has no meaning on a JES3 system.

Default: YES

FUNCTION=NA | ARM
Optionally specifies whether the information request is filtered based on whether the member supports the automatic restart manager (available on OS/390 MVS Version 2 Release 6 or higher).

If you specify FUNCTION=, you must also specify GROUPNAME=.

NA
Indicates that the FUNCTION= parameter is *not applicable*, meaning that JES XCF should not limit the information it returns based on whether the member supports the automatic restart function.

ARM
Indicates that JES XCF should limit the information it returns only for members that support the automatic restart function.

Default: NA

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the inline parameter list and invoke the wanted service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXIF service ended abnormally, and also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and reason codes

When the IXZXIXIF macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return code (hex)	Reason code (hex)	Meaning and action
0	0	**Meaning:** Processing was successful. The requested group or member information was returned. **Action:** none
0	4	**Meaning:** Processing failed because there are no members that match the filtering parameters (FUNCTION=, POLYJES=, STATE=, and SYSTEM=) specified on this macro request. **Action:** Verify that the filtering parameter specifications are set correctly. If appropriate, revise their settings and rerun the program that issues the IXZXIXIF macro.

Return code (hex)	Reason code (hex)	Meaning and action
4	4	**Meaning:** Processing failed because the ANSAREA= specification is too small to hold the requested information. The system updates the area identified by the ANSLEN= parameter to indicate the length necessary. **Action:** Obtain a new buffer of the length returned in the area identified by the ANSLEN= parameter and rerun the program that issues IXZXIXIF.
4	40	**Meaning:** Processing failed because the parameter list passed to the IXZXIXIF service contained an eyecatcher that is not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.
4	44	**Meaning:** Processing failed because the parameter list passed to the IXZXIXIF service contained a version indicator that is not valid. **Action:** You might need to recompile the calling program. Check the parameter list to be certain it was not overlaid with other data.
4	48	**Meaning:** Processing failed because the IXZXIXIF service was issued for a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	4C	**Meaning:** Processing failed because of a JES XCF internal error. **Action:** Contact the IBM support center, and supply the return and reason codes.
4	54	**Meaning:** Processing failed because of a JES XCF internal error. **Action:** Contact the IBM support center, and supply the return and reason codes.
4	58	**Meaning:** Processing failed because the parameter list passed to the IXZXIXIF service had a REQMBOX= name that is not valid. **Action:** Be certain you specified the REQMBOX= name correctly.
8	50	**Meaning:** Processing failed because of a JESXCF internal error. **Action:** Contact the IBM support center, and supply the return and reason codes.
C	0	**Meaning:** Processing failed because the IXZXIXIF service ended abnormally. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

IXZXIXMB - Build a mailbox

Use the IXZXIXMB macro to build a mailbox for a JES dispatchable unit (JDU) that intends to receive mail. In addition to defining a mailbox, a JDU can also use IXZXIXMB to request that it is notified of system events. JES XCF stores messages in the mailbox until the JDU receives the messages.

See “Building a Mailbox (IXZXIXMB Macro)” on page 15 for a description of using IXZXIXMB to build a mailbox.

Environment

The requirements for the caller are:

Table 13. Environment	
Variable	**Value**
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXMB is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXMB is invoked.

When the exit routine identified by POSTXIT= receives control, it is passed a parameter list. To map this parameter list, use mapping macro IXZYIXPE. IXZYIXPE is described in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Restrictions

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXMB is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXMB is invoked.

When the exit routine identified by POSTXIT= receives control, it is passed a parameter list. To map this parameter list, use mapping macro IXZYIXPE. IXZYIXPE is described in *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Input register information

Before issuing the IXZXIXMB macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output register information

When IXZXIXMB returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXMB returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as work registers by the system

2-13
Unchanged

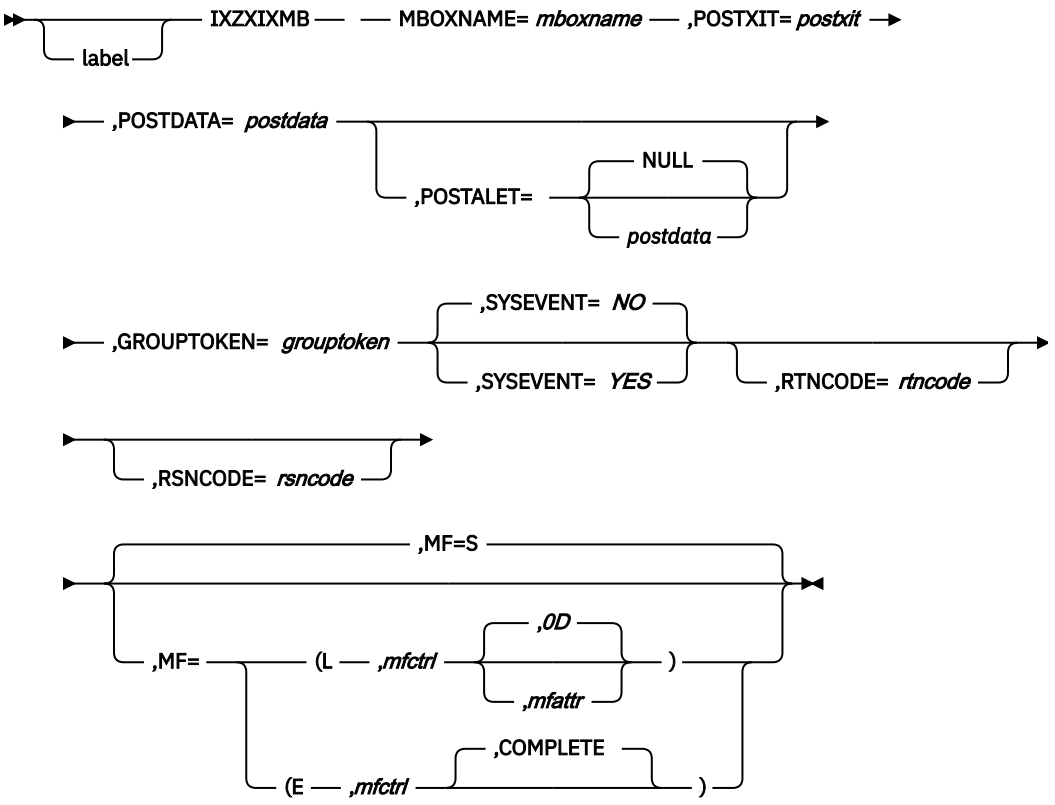
14,15
Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None

Syntax



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXMB macro invocation.

Default: no name

MBOXNAME=mboxname
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the mailbox name. Valid characters are A-Z, 0-9, $, #, @, and embedded and trailing blanks. The mailbox name must **not** duplicate another name on the member where the IXZXIXMB service is invoked, nor should the name begin with *SYSJES*.

You should use 2-part mailbox names. The first part of the name might identify your company or department. The second part might identify the function that you are providing. For example, if the Dave, Alan, Susan, and Don Corporation used JES XCF to provide an accounting function, they might name the mailbox:

```
DASDCORP_ACCOUNT
```

```
DASDCORP ACCOUNT
```

Default: none

POSTXIT=postxit
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the address of the POST exit routine. A POST exit routine (provided by the caller) receives control whenever a message is placed in the mailbox. This exit allows the caller to post a routine to retrieve the message from the mailbox and process its contents. Refer to Appendix A, “POST Exit Routine,” on page 121 for details on coding a POST exit routine.

Default: none

POSTDATA=postdata
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field needed by the POST exit routine to process post requests. The data is produced by the creator of the mailbox, and a pointer is returned in the parameter list passed to the POST exit routine. Use the POSTALET= parameter to specify an access list entry table (ALET) if you want to store this data in a data space or another address space.

Default: none

POSTALET=postalet | NULL
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword input field that contains an ALET if you choose to store the POSTDATA information in either a data space or another address space which is specified by the POSTDATA= keyword. Be certain to put this ALET on the primary address space number-access list (PASN-AL).

Default: NULL, indicating the data resides in the primary address space.

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token. See “Retrieving the JES XCF Group Token” on page 13 for the procedure for retrieving the JES XCF group token.

Default: none

SYSEVENT=NO | YES
Specifies whether (YES) or not (NO) JES XCF should inform the JDU of sysplex system events. Select YES only if the JDU needs to monitor sysplex system events.

Default: NO

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro, which is to build the inline parameter list and invoke the wanted service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXMB service ended abnormally, and also received an associated ABEND code. Refer to *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for a description of the possible following ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and reason codes

When the IXZXIXMB macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. The mailbox was built. **Action:** None
4	4	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMB service contained an eyecatcher that was not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.
4	8	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMB service had a version indicator that was not valid. **Action:** The caller might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.
4	C	**Meaning:** Processing failed because the POSTXIT= parameter was not specified. **Action:** Be certain you specified POSTXIT= on the IXZXIXMB call. POSTXIT= is required and **must** provide an address of a routine to be called when a message is placed in the mailbox.
4	10	**Meaning:** Processing failed because the POSTDATA= parameter was not specified. **Action:** Be certain you specified POSTDATA= on the IXZXIXMB call. POSTDATA= is required and **must** provide the address of the data area to be passed to the POST exit routine.
4	14	**Meaning:** Processing failed because the group token passed to the IXZXIXMB service was not valid. **Action:** Verify that the correct group token is being passed to IXZXIXMB.
4	18	**Meaning:** Processing failed because the IXZXIXMB service was issued on a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	1C	**Meaning:** Processing failed because the mailbox name specified contains characters that are not valid. **Action:** Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, $, #, @, and embedded and trailing blanks are valid.
4	20	**Meaning:** Processing failed because the mailbox specified is already defined. **Action:** Be certain you named the mailbox correctly. If you specified an incorrect name, correct the error and rerun the program that issues IXZXIXMB. If the specified mailbox has already been built, you need not rebuild it unless it was built with incorrect parameter specifications. If this is case, delete the mailbox and then rebuild it.

Return Code (hex)	Reason Code (hex)	Meaning and Action
C	0	**Meaning:** Processing failed because the IXZXIXMB service abended. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

IXZXIXMC - Clear a Mailbox

Use the IXZXIXMC macro to clear a specified mailbox of any messages that have been placed into the mailbox but have not been received. IXZXIXMC returns an acknowledgment message to the sending JES member indicating that the message was cleared from the mailbox.

Typically, IXZXIXMC might be used by a JES dispatchable unit (JDU) that has restarted and does not want to process messages that were received from the previous JES start. IXZXIXMC also might be used by a JDU that becomes available after being unavailable and does not want to process messages that were placed in its mailbox during that time.

Refer to “Clearing a Mailbox of Messages (IXZXIXMC Macro)” on page 30 for a description of using IXZXIXMC to clear a mailbox.

Environment

The requirements for the caller are:

Table 14. Environment

Variable	Value
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXMC is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXMC is invoked.

Restrictions

Do not issue IXZXIXMC until JES initialization processing has established an attachment to the JES XCF group; that is, after IXZXIT03 processing has completed.

When clearing a mailbox, you must do so under the same task in which the mailbox was built.

Input Register Information

Before issuing the IXZXIXMC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXMC returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXMC returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as work registers by the system

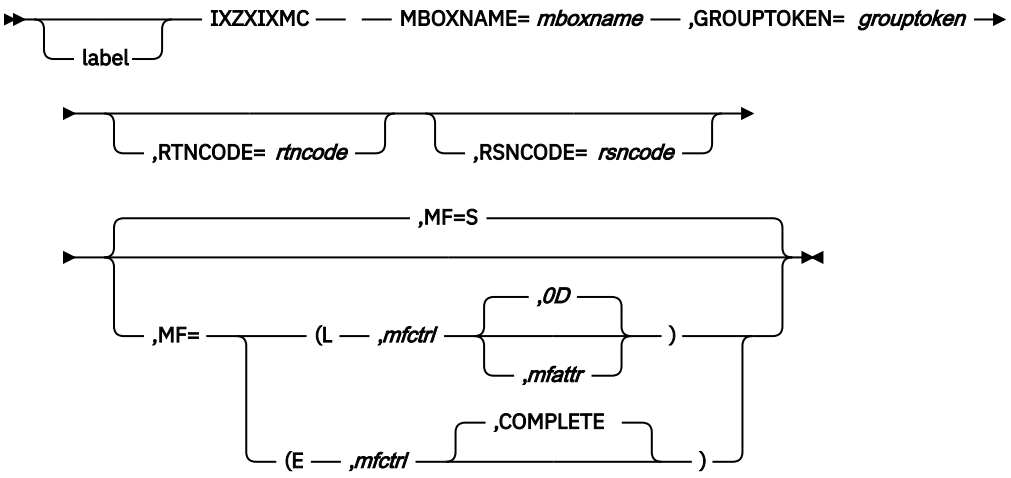
2-13
Unchanged

14,15
Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXMC macro invocation.

Default: no name

MBOXNAME=mboxname
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the name of the mailbox to be cleared. Valid characters are A-Z, 0-9, $, #, @, and embedded and trailing blanks.

Default: none

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token. Refer to "Retrieving the JES XCF Group Token" on page 13 for the procedure for retrieving the JES XCF group token.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the inline parameter list and invoke the wanted service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXMC service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and Reason Codes

When the IXZXIXMC macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. The mailbox was cleared. All messages that were in the mailbox were acknowledged with a return code that indicates the messages were not received. **Action:** None
4	4	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMC service contains an eyecatcher that is not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	8	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMC service contains a version indicator that is not valid. **Action:** The caller might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.
4	C	**Meaning:** Processing failed because the group token passed to the IXZXIXMC service was not valid. **Action:** Verify that the correct group token is being passed to IXZXIXMC.
4	10	**Meaning:** Processing failed because the mailbox name passed to the IXZXIXMC service routine does not exist. **Action:** Verify that the correct mailbox name is being passed to IXZXIXMC.
4	14	**Meaning:** Processing failed because the ASCB or TCB of the caller of the IXZXIXMC service does not match that of the mailbox builder (the caller of IXZXIXMB). **Action:** You attempted to clear a mailbox under a different task than that used to build the mailbox. You can only clear the mailbox when under the task from which IXZXIXMB was issued. Be certain that both the ASCB and TCB of the caller match those of the mailbox builder, and rerun the program.
4	18	**Meaning:** Processing failed because the IXZXIXMC service was issued on a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	1C	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMC service routine specified a mailbox name that is not valid. **Action:** Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, $, #, @, and embedded and trailing blanks are valid.
C	0	**Meaning:** Processing failed because the IXZXIXMC macro service abended. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

IXZXIXMD - Delete a Mailbox

Use the IXZXIXMD macro to delete a mailbox and free any resources associated with using the mailbox. Use IXZXIXMD when there is no longer a need to communicate with other JES members or when the JES dispatchable unit (JDU) that owns the mailbox terminates. Any messages still in the mailbox are deleted and an appropriate acknowledgment is returned to the JES members that sent the messages.

Refer to "Deleting a Mailbox (IXZXIXMD Macro)" on page 30 for a description of using IXZXIXMD to delete a mailbox.

Environment

The requirements for the caller are:

Table 15. Environment	
Variable	**Value**
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXMD is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXMD is invoked.

Restrictions

Do not issue IXZXIXMD until JES initialization processing has established an attachment to the JES XCF group; that is, after IXZXIT03 processing has completed.

When deleting a mailbox, you must do so under the same task in which the mailbox was built.

Input Register Information

Before issuing the IXZXIXMD macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXMD returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXMD returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as work registers by the system

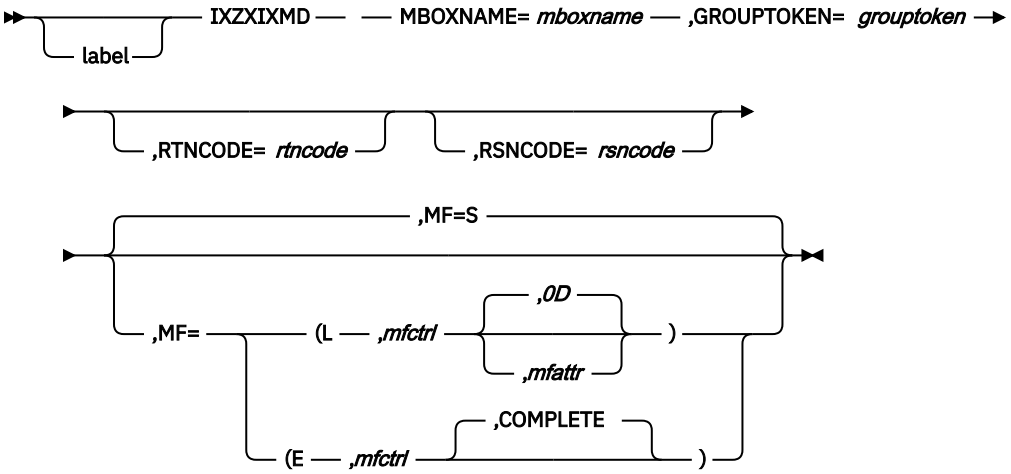
2-13
Unchanged

14,15
Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXMD macro invocation.

Default: no name

MBOXNAME=mboxname
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the name of the mailbox to be deleted. Valid characters are A-Z, 0-9, $, #, @, and embedded and trailing blanks.

Default: none

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token. Refer to "Retrieving the JES XCF Group Token" on page 13 for the procedure for retrieving the JES XCF group token.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the inline parameter list and invoke the wanted service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXMD service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and Reason Codes

When the IXZXIXMD macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. The mailbox has been deleted. All messages that were in the mailbox were acknowledged with a return code that indicates the messages were not received. **Action:** None
4	4	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMD service contains an eyecatcher that is not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.
4	8	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMD service contains a version indicator that is not valid. **Action:** The caller of IXZXIXAC may need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.
4	C	**Meaning:** Processing failed because the group token passed to the IXZXIXMD macro service is not valid. **Action:** Verify that the correct group token is being passed to IXZXIXMD.
4	10	**Meaning:** Processing failed because the mailbox name specified does not exist. **Action:** Verify that the correct mailbox name is being passed to IXZXIXMD.
4	14	**Meaning:** Processing failed because the ASCB and/or TCB of the caller of the IXZXIXMD service do not match those of the mailbox builder (the caller of IXZXIXMB). Be certain that both the ASCB and TCB of the caller match those of the mailbox builder, and rerun the program. **Action:** You attempted to delete a mailbox under a different task than that used to build the mailbox. You can only delete the mailbox when under the task from which IXZXIXMB was issued.
4	18	**Meaning:** Processing failed because the IXZXIXMD service was issued on a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	1C	**Meaning:** Processing failed because the parameter list passed to the IXZXIXMD service contained a mailbox name that was not valid. **Action:** Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, $, #, @, and embedded and trailing blanks are valid.
C	0	**Meaning:** Processing failed because the IXZXIXMD service abended. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

[Programming Interface Information] IXZXIXPI - Collect JES3 Performance Information Data

Use the IXZXIXPI macro to provide an interface for resource monitoring products such as Resource Measurement Facility (RMF) to obtain required data. Use this macro to extract information from the JES XCF message queues concerning address space delays encountered due to JES3 processing delays.

Refer to "Collecting JES3 performance information data (IXZXIXPI macro)" on page 27 for a description of using IXZXIXPI to access JES XCF message queues.

Environment

The requirements for the caller are:

Table 16. Environment

Variable	Value
JES environments:	JES3 User
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXPI is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXPI is invoked.

Restrictions

Do not issue IXZXIXPI until JES3 initialization processing has established an attachment to the JES XCF group; that is, after IXZXIT03 processing has completed.

Input Register Information

Before issuing the IXZXIXPI macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXPI returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXPI returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as a work register by the system

2-13
Unchanged

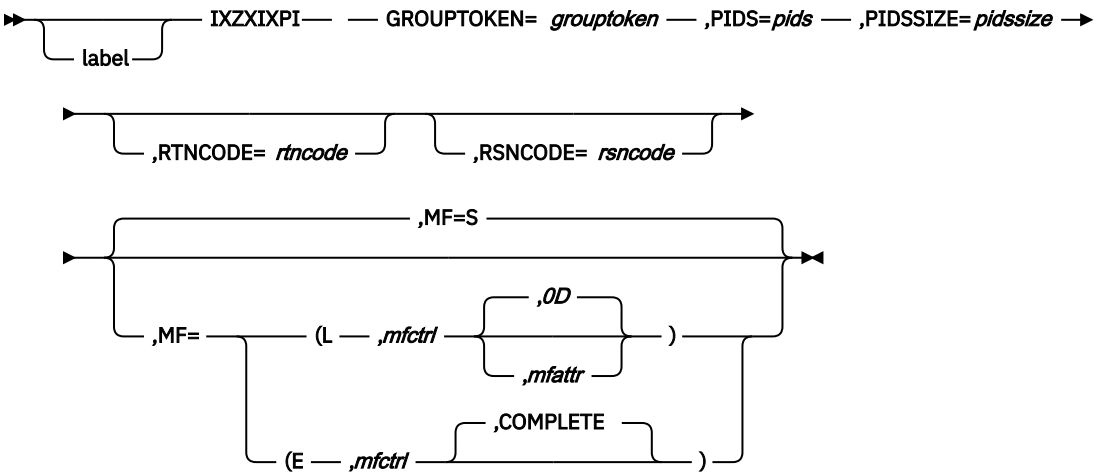
14,15
Used as a work register by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

None

Syntax



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXPI macro invocation.

Default: no name

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token returned by the IXZXIXAT macro.

Default: none

PIDS=pids
Specifies the name (RS-type) of a required fullword input/output field or register (GPR2-GPR12) into which JES XCF puts the address of the JES3 performance output data. This storage is in the caller's address space (key 0).

Default: none

PIDSSIZE=pidssize
Specifies the name (RS-type) of a required fullword input/output field or register (GPR2-GPR12) that contains the size of the JES3 performance output data area to which the PIDS= parameter points. If the size of the area requested is too small, JES XCF returns the correct size requirement to the caller in the field specified by this parameter.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the in-line parameter list and invoke the desired service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the desired degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXPI service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of possible ABEND codes.

Return and Reason Codes

When the IXZXIXPI macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. JES XCF accessed and returned the JES3 performance data. **Action:** None
4	0	**Meaning:** Processing failed because the size of the storage requested (PIDSSIZE=) was too small. **Action:** Change the PIDSSIZE= specification to the value IXZXIXPI returned in the field pointed to by PIDS= and rerun the program that issues the IXZXIXPI macro.
8	4	**Meaning:** Processing failed because the IXZXIXPI service exceeded its retry count. The IXZXIXPI service routine is unable to gather data due to the lack of serialization. **Action:** This return code is temporary. Rerun the program that issues the IXZXIXPI macro or assume there is no data to be returned at this point in the processing cycle.
C	0	**Meaning:** Processing failed because the IXZXIXPI service abended. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

[End Programming Interface Information]

IXZXIXRM - Receive a message

Use the IXZXIXRM macro to retrieve messages from the specified mailbox. IXZXIXRM retrieves all messages that are in the mailbox or only the specific kinds of messages you select.

See “Receiving a Message (IXZXIXRM Macro)” on page 23 for a description of using IXZXIXRM to receive messages.

Environment

The requirements for the caller are:

Table 17. Environment	
Variable	**Value**
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXRM is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXRM is invoked.

For each message you receive, you must issue IXZXIXAC to acknowledge receipt of the message to JES XCF.

IBM defined the format and content of the message envelope, system event messages, and acknowledgment messages. The format and content of all other messages must be previously agreed to by the sender and the receiver.

IBM provides several mapping macros that you can use to map the information that is returned by IXZXIXRM. For a description of the following mapping macros, see *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Mapping Macro
Maps

IXCYGEPL
XCF data

IXZYIXAC
Acknowledgment messages

IXZYIXEN
The message envelope

IXZYIXIF
JES XCF data

IXZYIXJE
JES data

IXZYIXSE
System event messages

Also, see Figure 8 on page 25 for a graphic view of the message envelope and associated mapping macro structure, and *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) IXZYIXEN macro mapping.

Restrictions

Do not issue IXZXIXRM until JES initialization processing established an attachment to the JES XCF group, which is after IXZXIT03 processing completes.

When retrieving a message from a mailbox, you must do so under the same task in which the mailbox was built.

Input register information

Before issuing the IXZXIXRM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output register information

When IXZXIXRM returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXRM returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as work registers by the system

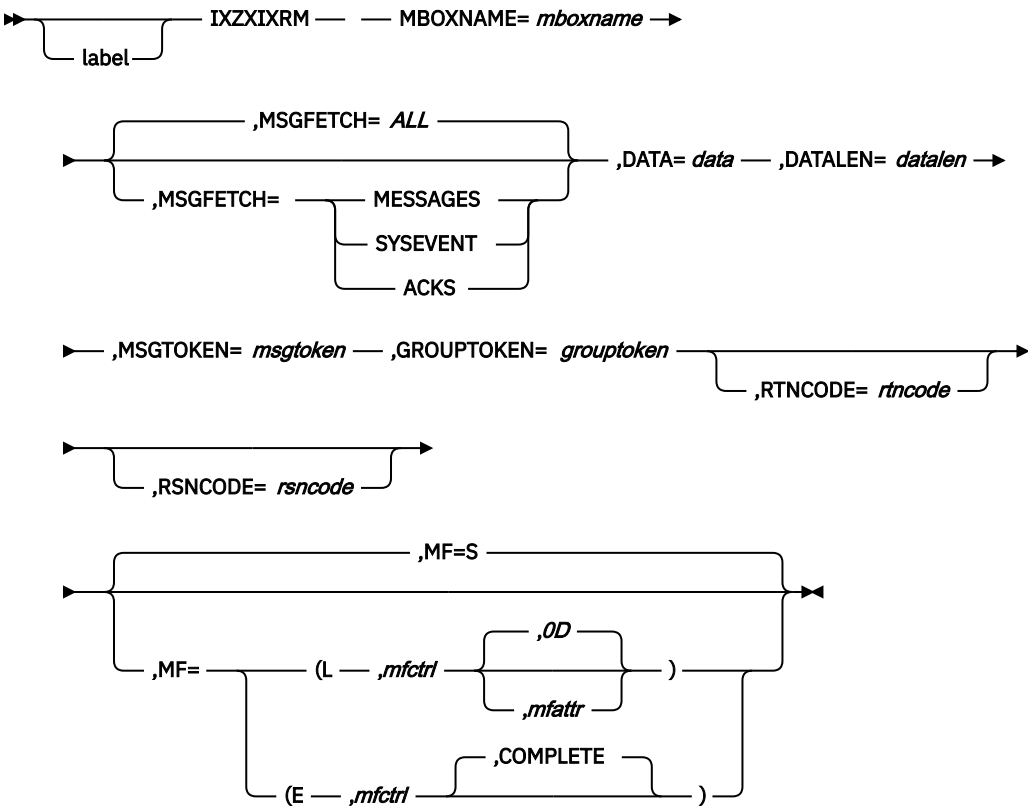
2-13
Unchanged

14,15
Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None



Parameters

label
Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXRM macro invocation.

Default: no name

MBOXNAME=mboxname
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the name of the mailbox where messages are placed for processing. Valid characters are A-Z, 0-9, $ # @, and embedded and trailing blanks.

Default: none

MSGFETCH=ALL | MESSAGES | SYSEVENT | ACKS
Optionally specifies the type of messages to be retrieved from the mailbox. Messages are ordered chronologically within the message type as follows:

ALL
Retrieve all messages in the following order:

1. System events (SYSEVENT)
2. Acknowledgements (ACKS)
3. Messages (MESSAGES)

MESSAGES
Only retrieve messages sent by IXZXIXSM from this or other sysplex members.

SYSEVENTS
Only retrieve system event messages, for example, to monitor system state changes.

ACKS
Only retrieve acknowledgements of previous IXZXIXSM (send message) invocations.

Default: ALL

DATA=data
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword output field that contains the address of the message data received.

Default: none

DATALEN=datalen
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword output field that contains the length (up to 60 kilobytes of data) of the message received.

Default: none

MSGTOKEN=msgtoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 64-bit output field that contains the token that is used when acknowledging receipt of a message.

Default: none

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token. See "Retrieving the JES XCF Group Token" on page 13 for the procedure for retrieving the JES XCF group token.

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the inline parameter list and invoke the process acknowledgment service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the wanted degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND codes

When control returns to your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXRM service ended abnormally, and also received an associated ABEND code. See *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and reason codes

When the IXZXIXRM macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return code (hex)	Reason code (hex)	Meaning and action
0	0	**Meaning:** Processing was successful. The address of an envelope containing a message was returned. **Action:** None
4	4	**Meaning:** Processing failed because the parameter list passed to the IXZXIXRM service contained an eyecatcher that was not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.
4	8	**Meaning:** Processing failed because the parameter list passed to the IXZXIXRM service contained a version indicator that was not valid. **Action:** The caller might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.
4	C	**Meaning:** Processing failed because the group token passed to the IXZXIXRM service was not valid. **Action:** Verify that the correct group token is being passed to IXZXIXAC.
4	10	**Meaning:** Processing failed because the parameter list passed to the IXZXIXRM service contained a mailbox name that was not valid. **Action:** Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, $, #, @, and embedded and trailing blanks are valid.

Return code (hex)	Reason code (hex)	Meaning and action
4	14	**Meaning:** Processing failed because there are no messages in the mailbox specified on the IXZXIXRM macro. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	18	**Meaning:** Processing failed because the ASCB or TCB of the caller invoking the IXZXIXRM service do not match those of the specified mailbox (those of the caller of the IXZXIXMB service). **Action:** You attempted to receive mail under a different task than that used to build the mailbox. You can only receive mail from a mailbox when under the task from which IXZXIXMB was issued. Be certain that both the ASCB and TCB of the caller match those of the mailbox builder, and rerun the program.
4	1C	**Meaning:** Processing failed because the mailbox specified on the IXZXIXRM macro does not exist. **Action:** Verify that the correct mailbox name is being passed to IXZXIXRM.
4	20	**Meaning:** Processing failed because the IXZXIXRM service was called on a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
8	0	**Meaning:** IXZXIT02 failed. The message being processed at the time IXZXIT02 failed is lost. **Action:** Review your IXZXIT01 code for coding errors, correct the error and rerun the program that issues the IXZXIXRM macro.
C	0	**Meaning:** Processing failed because the IXZXIXRM service abended. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

IXZXIXSM - Send a Message

Use the IXZXIXSM macro to send a message from one JES dispatchable unit (JDU) to another JDU. The JDUs can be running on the same JES member or on different JES members of the XCF group.

You can have more than one multi-segment message active at a time. That is, message segments from more than one multi-segment message can be in the process of being sent. The sending member sends complete multi-segment messages in the order that it receives their last segments.

If you start a multi-segment message, and then send single-segment messages before sending the last segment of the multi-segment message, all of the intervening single-segment messages will be sent before the multi-segment message is sent. A multi-segment message is considered incomplete and is not sent until the last segment (SEGTYPE=LAST or SEGTYPE=ABORT) has been sent.

Refer to “Sending a Message (IXZXIXSM Macro)” on page 16 for a description of using IXZXIXSM to send a message.

Environment

The requirements for the caller are:

Table 18. Environment	
Variable	**Value**
JES environments:	• JES2 or JES3 main task • JES2 or JES3 subtask • JES2 or JES3 FSS • User
Minimum authorization:	Supervisor state with PSW key 0 or 1
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held
Control parameters:	None

Programming Requirements

JES2 programs must include the $MODULE macro so it is invoked before IXZXIXSM is invoked.

JES3 programs must include the ENVIRON= keyword on the IATYMOD macro so it is invoked before IXZXIXSM is invoked.

Restrictions

Do not issue IXZXIXSM until JES initialization processing has established an attachment to the JES XCF group; that is, after IXZXIT03 processing has completed.

If two or more JES members are involved in the "send message" operation, both must belong to the same JES XCF group.

Input Register Information

Before issuing the IXZXIXSM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When IXZXIXSM returns control, the general purpose registers (GPRs) contain:

Register
Content

0
Reason code

1
Used as a work register by the system

2-13
Unchanged

14
Used as a work register by the system

15
Return code

When IXZXIXSM returns control, the access registers (ARs) contain:

Register
Content

0,1
Used as work registers by the system

2-13
Unchanged

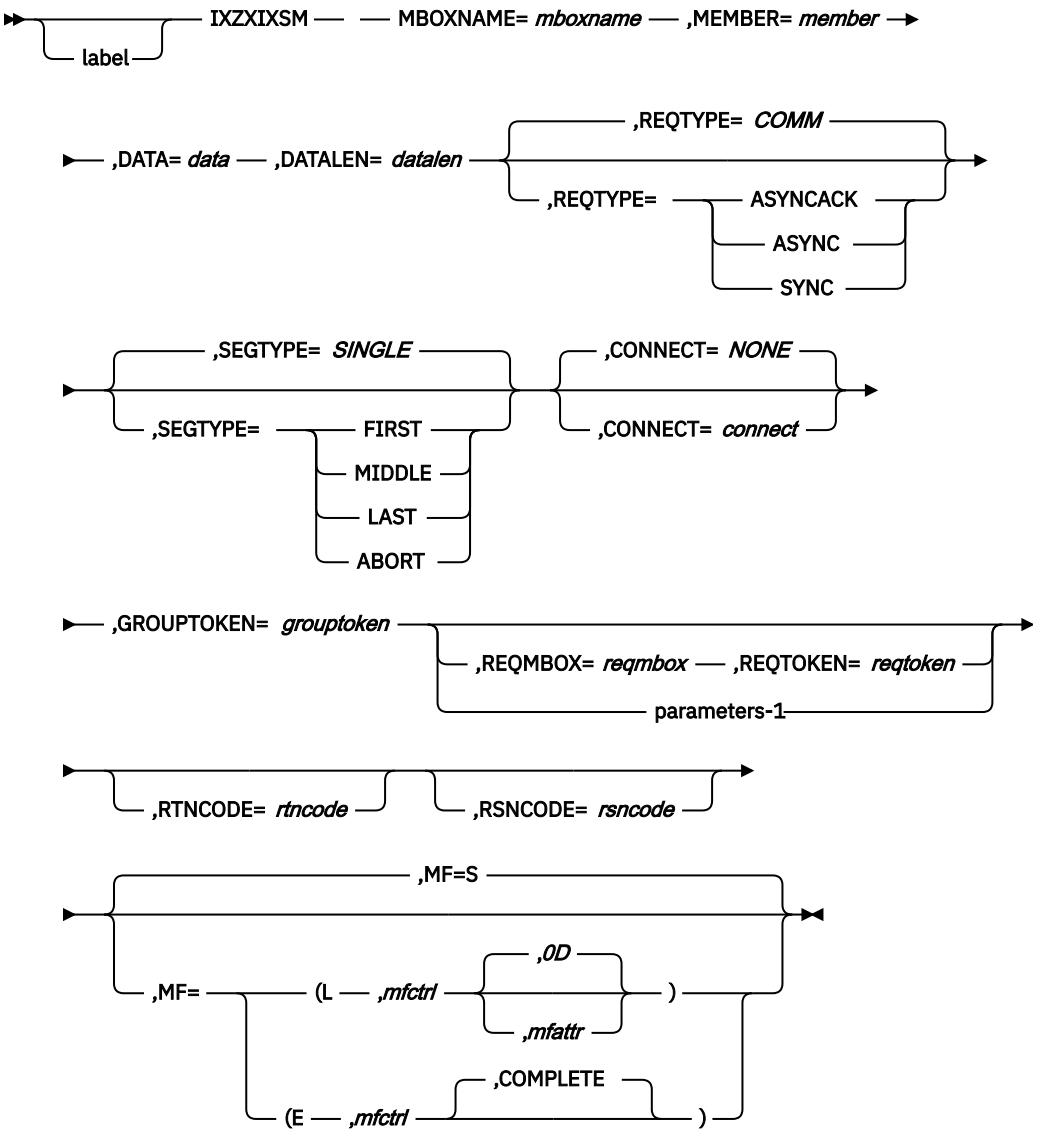
14,15
Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

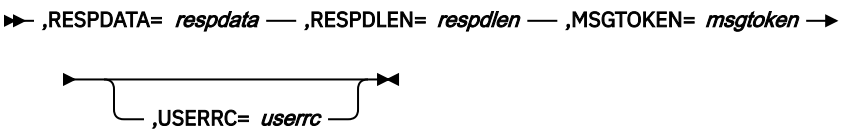
Performance Implications

None

Syntax



parameters-1



Parameters

label

Specifies an optional symbol, starting in column 1, to be used as the name on the IXZXIXSM macro invocation.

Default: no name

MBOXNAME=mboxname

Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the name of the mailbox where messages are placed for processing by another JDU. Valid characters are A-Z, 0-9, $, #, @, and embedded and trailing blanks.

Default: none

MEMBER=member
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required 16-character input field that contains the name of the XCF member to which a message is to be sent. This name must respect the following naming standards for XCF groups: characters A-Z, 0-9, and $, #, @; and must be left justified, and padded on the right with blanks. Avoid names beginning with A-I, the character string *SYS*, or the reserved name *UNDESIG*. These are reserved for IBM use only.

Default: none

DATA=data
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the address of the message data to be sent.

Default: none

DATALEN=datalen
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the length (up to 60 kilobytes in decimal) of the message data to be sent.

Default: none

REQTYPE=COMM | ASYNCACK | ASYNC | SYNC
Optionally specifies how you prefer acknowledgements to be handled.

COMM
Process the message asynchronously. There is no recovery processing if either the sending or receiving system fail before the message is processed. This option is used for most typical communication, such as WTO message processing, between members of the XCF group.

ASYNCACK
Process the message asynchronously. The sender receives an acknowledgement message from the receiver, thus providing the sender with potential recovery situation information.

Note: You must also specify REQTOKEN= and REQMBOX= for REQTYPE=ASYNCACK messages.

ASYNC
Process the message asynchronously. No acknowledgement message is returned to the sender. JES XCF provides recovery processing.

SYNC
Process the message synchronously. The sender waits until the receiver returns an acknowledgement before continuing further processing. If you specify SYNC, and this is a multi-segment message of SEGTYPE=LAST or SEGTYPE=ABORT, you must also specify MSGTOKEN=.

Default: COMM

REQMBOX=reqmbox
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 16-character input field that contains the name of the mailbox into which the acknowledgement message is returned. Valid characters are A-Z, 0-9, $, #, @, and embedded and trailing blanks. If you specify REQTYPE=ASYNCACK, you must also specify REQMBOX.

Default: none

REQTOKEN=reqtoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 64-bit output field that contains a token used to associate the message acknowledgement with an issued message.

Note: If you specify REQTYPE=ASYNCACK, you must also specify REQTOKEN; all other REQTYPE specifications receive a binary 0 if you specify REQTOKEN.

Default: none

SEGTYPE=SINGLE | FIRST | MIDDLE | LAST | ABORT
Optionally specifies the type of message that is being sent.

SINGLE
The message being sent is a single-segment message.

FIRST
The message is the first segment of a multi-segment message. Use the CONNECT= keyword to receive a token to associate all following segments of the message.

MIDDLE
The message is a segment other than the first or last of a multi-segment message.

LAST
The message is the last segment of a multi-segment message.

ABORT
The multi-segment message is terminated, and all held resources are freed.

Default: SINGLE

CONNECT=connect | NONE
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 64-bit input/output field used to associate the segments of a multi-segment message. This token is returned for SEGTYPE=FIRST and must be passed as input for SEGTYPE=MIDDLE | LAST | ABORT. CONNECT= has no meaning for SEGTYPE=SINGLE.

Default: NONE

GROUPTOKEN=grouptoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of a required fullword input field that contains the group token. Refer to "Retrieving the JES XCF Group Token" on page 13 for the procedure for retrieving the JES XCF group token.

Default: none

RESPDATA=respdata
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword output field that contains the address of a message buffer that contains the response message returned by the acknowledger of the message; that is, the response message specified by the DATA= keyword on the IXZXIXAC macro.

Default: none

RESPDLEN=respdlen
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword output field that contains the length of the response message returned by the acknowledger of the message; that is, the response message length specified by the DATALEN= keyword on the IXZXIXAC macro.

Default: none

MSGTOKEN=msgtoken
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional 64-bit output field that contains the token that must be used when acknowledging that the response data has been received. This token will be used as input to the MSGTOKEN= keyword on the IXZXIXAC macro.

Default: none

USERRC=userrc
Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword output field that returns the processing results as specified by the message receiver (through the USERRC= parameter on the IXZXIXAC macro). USERRC only has meaning for REQTYPE=SYNC because only synchronous messages receive a direct response from the receiver. All asynchronous messages receive a binary 0, and receive the user return code in the message envelope (mapped by IXZYIXEN).

Default: none

RTNCODE=rtncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the return code is copied from GPR 15.

Default: none

RSNCODE=rsncode
Specifies the name (RS-type) of an optional fullword output field or register (GPR2-GPR12) into which the reason code is copied from GPR 0.

Default: none

MF=S | L | E
Specifies the form of the macro as standard (S), list (L), or execute (E). This keyword is optional.

MF=S
Specifies the standard form of the macro; that is, to build the in-line parameter list and invoke the desired service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr | 0D})
Specifies the list form of the macro; that is, defining an area to be used for the parameter list. If you code MF=L, do not code anything else except a label, the macro name, and the following values:

mfctrl
Specifies the name of a storage area to contain the parameters.

mfattr | 0D
Specifies an optional 1- to 60-character input string, which can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *mfattr*, the system provides a value of 0D, which forces the parameter list to a doubleword boundary.

Default: 0D

MF=(E,mfctrl,COMPLETE)
Specifies the execute form of the macro; that is, builds the parameter list specified by *mfctrl*. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

mfctrl
Specifies the name (RS-type) or address (in GPR1-GPR12) of a storage area for the parameter list.

COMPLETE
Specifies the desired degree of macro parameter syntax checking. Syntax checking includes checking for all required keywords and supplying default values, if any, for all omitted optional parameters.

Default: COMPLETE

Default: S

ABEND Codes

When control returns from your program, if you receive a return code of X'C', it indicates that processing failed because the IXZXIXSM service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* for a description of the following possible ABEND codes:

DC5
JES XCF detected an unrecoverable error during attach or detach processing.

EC5
JES XCF detected an unrecoverable error while attempting recovery from a previous abend.

Return and Reason Codes

When the IXZXIXSM macro returns control to your program, GPR 15 (and *rtncode* if you coded RTNCODE) contains the return code and GPR 0 (and *rsncode* if you coded RSNCODE) contains the reason code.

Return Code (hex)	Reason Code (hex)	Meaning and Action
0	0	**Meaning:** Processing was successful. If the call was asynchronous, the message was queued to be sent. If the call was synchronous, the message was delivered to the receiver and acknowledged. **Action:** None
4	4	**Meaning:** Processing failed because the parameter list passed to the IXZXIXSM service contains an eyecatcher that is not valid. **Action:** Check the parameter list to be certain it was not overlaid with other data.
4	8	**Meaning:** Processing failed because the parameter list passed to the IXZXIXSM service does not have a valid version indicator. **Action:** The caller might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.
4	C	**Meaning:** Processing failed because the group token passed to the IXZXIXSM service was not valid. **Action:** Verify that the correct group token is being passed to IXZXIXSM.
4	10	**Meaning:** Processing failed because the member name passed to the IXZXIXSM service contains characters that are not valid. **Action:** Be certain you specified MEMBER= correctly. Only A-Z, 0-9, $, #, and @ are valid.
4	14	**Meaning:** Processing failed because the mailbox name passed to the IXZXIXSM service contains characters that are not valid. **Action:** Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, $, #, @ and embedded and trailing blanks are valid.
4	18	**Meaning:** Processing failed because the mailbox name specified on the REQMBOX= parameter contains characters that are not valid. **Action:** Be certain you specified REQMBOX= correctly. Only A-Z, 0-9, $, #, @, and embedded and trailing blanks are valid.
4	20	**Meaning:** Processing failed because the IXZXIXSM service was called on a member that is detaching. **Action:** Check to be certain this is not a symptom of a serialization problem within your JES XCF group.
4	24	**Meaning:** Processing failed because CONNECT= was not specified. **Action:** Be certain that for SEGTYPE=FIRST | LAST | ABORT send message calls, you also include the CONNECT= parameter. If it is missing, add it and rerun the program that issues the IXZXIXSM macro.
4	28	**Meaning:** Processing failed because MSGTOKEN= was not specified. service. **Action:** Be certain that for this REQTYPE=SYNC message send call, you also include the MSGTOKEN= parameter. If it is missing, add it and rerun the program that issues the IXZXIXSM macro.

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	2C	**Meaning:** Processing failed because you did not also specify REQMBOX= on this IXZXIXSM call. REQMBOX= is required for ASYNCACK call to identify the mailbox into which the acknowledgement message is returned. **Action:** Add REQMBOX= to the IXZXIXSM macro and rerun the program that issues the send message call.
4	34	**Meaning:** Processing failed because the CONNECT= specification is not valid. A SEGTYPE=MIDDLE | LAST | ABORT was specified, but the first segment was not found or the last segment was already sent. **Action:** Be certain the CONNECT= parameter was specified on all but SEGTYPE=SINGLE messages. Resend the message and its segments using the CONNECT= token returned on a SEGTYPE=FIRST call as input to all successive SEGTYPE=MIDDLE | LAST | ABORT calls.
4	38	**Meaning:** Processing failed because the DATALEN= value is not valid. **Action:** DATALEN= must be a positive decimal integer of 0 to 60 kilobytes. Validate the size of the data packet and rerun the program that issues the IXZXIXSM macro.
4	3C	**Meaning:** Processing failed because the message was cleared (by a IXZXIXMC call) from the target mailbox before the message was processed. **Action:** You must be aware of message processing unique to REQTYPE=SYNC messages. To prevent this error, consider and provide necessary code in your routine to handle potential situations in which a mailbox could be cleared or deleted during synchronous message processing.
4	48	**Meaning:** Processing failed because the message was deleted (by a IXZXIXMD call) from the target mailbox before the message was processed. **Action:** You must be aware of message processing unique to REQTYPE=SYNC messages. To prevent this error, consider and provide necessary code in your routine to handle potential situations in which a mailbox could be cleared or deleted during synchronous message processing.
4	4C	**Meaning:** Processing failed because, although the message was delivered, it was not received by the intended recipient. Likely the recipient previously deleted the mailbox using the IXZXIXMD service. **Action:** You must be aware of message processing unique to REQTYPE=SYNC messages. To prevent this error, consider providing necessary code in your routine to handle situations in which a mailbox could potentially be deleted prematurely during synchronous message processing.
8	0	**Meaning:** IXZXIT01 failed. The message being processed at the time IXZXIT01 failed is lost. **Action:** Review your IXZXIT01 code for coding errors, correct the error, and rerun the program that issues the IXZXIXSM macro.
C	0	**Meaning:** Processing failed because the IXZXIXSM service abended. **Action:** Refer to *z/OS MVS System Codes* for a description of the abend code and its associated reason codes.

Appendix A. POST Exit Routine

The POST exit routine is invoked when JES XCF places a message into a mailbox. The function of this exit routine is to post the JES JDU that is responsible for retrieving messages from the mailbox. Refer to “Identifying the Installation-Written POST Exit Routine” on page 16 for information on the use and requirements for this routine.

Installing the Exit Routine

The POST exit routine does not have a specific name, so there are no installation requirements other than ensuring that the module which contains the POST exit routine is linkedited into an authorized library in the LNKLST concatenation. Define the exit as follows:

- AMODE 31
- Reentrant

Exit Routine Environment

The environment at the time the exit routine receives control is as follows:

Address space
JES address space

AMODE
31

ASC Mode
Access register (AR)

Authorization
Supervisor state and key 1

Cross memory mode
PASN=HASN=SASN

Dispatchable unit mode
Task

JES environments
None

Locks held
None

RMODE
ANY

Interrupt status
Enabled for I/O and external interrupts

Storage
Pageable, key 1

Exit Recovery

JES XCF provides recovery for the POST exit routine. If the exit routine fails, the calling program provides a dump and will continue processing the next POST exit request.

Exit Routine Processing

The POST exit routine receives control under a special JES subtask of the task which issued the IXZXIXAT macro. The exit routine posts the JDU that owns the mailbox and returns control to the calling program.

Programming Considerations

- The POST exit routine should not use any system services other than those required to post the JDU that owns the mailbox.
- Do not use services that can cause an MVS wait.

Entry Specifications

When the POST exit routine receives control, GPR1 contains the address of the parameter list mapped by the IXZYIXPE mapping macro. For a description of IXZYIXPE, see *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Registers at Entry

When the POST exit routine receives control, the general purpose registers (GPRs) contain the following information:

Register
Content

0
Does not contain any information for use by this routine

1
Address of a pointer to the IXZYIXPE parameter list

2-13
Do not contain any information for use by this routine

14
Return address

15
Entry point address

When the POST exit routine receives control, the access registers (ARs) contain the following information:

Register
Content

0-15
Do not contain any information for use by this routine

Parameter List Contents

The parameter list contains the following data:

- XCF group name
- JES XCF member name
- Mailbox name
- POST exit data address
- POST exit data address ALET.

Return Specifications

When the mailbox POST exit returns control to JES XCF, it must return control to the address that was specified in register 14 at the time the exit was entered.

Registers at Exit

On return from exit processing, the contents of the general purpose registers (GPRs) must be:

Register
Content

0-15
The routine must restore the contents to what they were when the routine received control

On return from exit processing, the contents of the access registers (ARs) must be:

Register
Content

0-15
The routine must restore the contents to what they were when the routine received control

Return Code Meanings

There are no return codes from the POST exit routine.

Appendix B. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact z/OS web page (www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with 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 are considered 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 the 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, 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 is placed 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 to provide 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, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark 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 the 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.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate 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 dotted decimal number can specify the ! 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 the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the 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 applies only 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.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero 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, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

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 can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the 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 repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the `*` symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
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 document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which 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.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, 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 (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface Information

This document is intended to help the system programmer use the JES common coupling services of z/OS MVS. This document manual also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS.

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 (www.ibm.com/legal/copytrade.shtml).

Index

Special Characters

<MVS and JES>
    summary of changes xv

A

abend processing
    for mailboxes 31
abort
    multi-segment message 22
accessibility
    contact IBM 125
    features 125
acknowledgement destination
    change using IXZXIT01 45
    change using IXZXIT02 51
acknowledgement of message
    asynchronous 20
    receipt 25
    requirement 25
    synchronous 20
append message extents
    using IXZXIT01 45
assistive technologies 125
asynchronous message
    with acknowledgement 20
    without acknowledgement 20
attach/detach exit
    IXZXIT03 54

B

build a mailbox
    using IXZXIXMB 88

C

change message
    content
        using IXZXIT02 51
    destination
        using IXZXIT01 44
clear a mailbox
    using IXZXIXMC macro 94
component trace
    for JES XCF processing 32
    SYSJES COMP id 32
contact
    z/OS 125
control macro expansion printing
    using ZPRINT 65

D

delete a mailbox
delete a mailbox *(continued)*
    using IXZXIXMD macro 98
detach JES
    from a JES XCF group 31
diagnostic support
    for JES XCF 31
discontinue sending
    a multi-segment message 22

E

entry specifications
    for IXZXIT01 47
    for IXZXIT02 52
    for IXZXIT03 59
    for POST exit routine 122
envelope
    for messages 24
environment
    IXZXIT01 43
    IXZXIT02 49
    IXZXIT03 54
    POST exit routine 121
exit
    for JES XCF 37
    IXZXIT01 43
    IXZXIT02 49
    IXZXIT03 54
    POST exit routine 121

F

feedback xiii
filter data
    returned by IXZXIXIF macro 26
FLOW trace record 32
FUNCTION
    IXZXIXIF parameter 86

G

GROUPNAME
    IXZXIXIF parameter 84

I

information about JES XCF group members
    obtain using IXZXIXIF 26
initialization processing
    for JES 9
install
    IXZXIT01 43
    IXZXIT02 49
    IXZXIT03 54
    POST exit routine 121
installation exit

installation exit *(continued)*
identifying the POST exit routine 16
trace record 33
installation-defined attachment to common coupling services
creating 14
IXZ$XPL mapping macro
for IXZXIT01 processing 44
for IXZXIT02 processing 50
IXZXIT01 installation exit
appending message extents 45
changing
message content 45
message destination 44
entry specifications 47
environment 43
installing 43
parameter list description 47
processing 43
programming considerations 47
recovery 43
registers
entry 47
exit 48
rerouting an acknowledgement 45
return codes 49
IXZXIT02 installation exit
changing message content 51
entry specifications 52
environment 49
installing 49
parameter list description 52
processing 50
programming considerations 52
recovery 50
registers
entry 52
exit 53
rerouting an acknowledgement 51
retrieving message extents 52
return codes 54
IXZXIT03 installation exit
attach/detach exit 54
entry specifications 59
environment 54
exit processing 55
installing 54
parameter list description 60
programming considerations 59
recovery 55
registers
entry 59
exit 60
return codes 61
IXZXIXAC macro
acknowledge message 65
IXZXIXAT macro
attach to JES XCF group 70
IXZXIXCL macro
system cleanup initiated indicator 75
IXZXIXDT macro
detach from a JES XCF group 31, 78
IXZXIXIF
to obtain information 26
IXZXIXIF macro
filtering data 26
request member information 82
IXZXIXIF parameter
FUNCTION 86
GROUPNAME 84
POLYJES 86
STATE 85
SYSTEM 85
IXZXIXMB macro
build a mailbox 88, 93, 94
IXZXIXMD macro
delete a mailbox 98
IXZXIXPI macro
collect JES3 performance data 103
JES3 processing delays 27
IXZXIXRM macro
receive a message 106
IXZXIXSM macro
send a message 112
IXZYIXIF mapping macro 27
IXZYIXPE mapping macro 89
IXZYIXSE mapping macro 16

J

JES
group token retrieval 13
initialization processing 9
members
obtaining information about 26
message processing 10
JES abend processing
for mailboxes 31
JES common coupling services
using 9
JES XCF
affecting message processing 5
diagnostic support 31
exits
implementing 37
getting system cleanup status from JES 30
introduction 1
providing security 6
trace support 31
uses 5
JES XCF group token
retrieve 13
JES XCF macro
IXZXIXAC 65
IXZXIXAT 70
IXZXIXCL 75
IXZXIXDT 78
IXZXIXIF 82
IXZXIXMB 88
IXZXIXMC 94
IXZXIXMD 98
IXZXIXPI 103
IXZXIXRM 106
JES XCF trace 32
JES-defined attachment
determining if you should use the attachment 13
to XCF
description 13

JES-defined attachment *(continued)*
to XCF *(continued)*
IBM recommendation 13
JES2 MAS
relationship to sysplex 1
JES3
collect performance data 103
delays in processing 27
JES3 complex
relationship to sysplex 1

K

keyboard
navigation 125
PF keys 125
shortcut keys 125

M

macro
IXZXIXSM 112
macro syntax
how to read 63
mailbox
availability 15
building 15
clearing messages 30
deleting 30
POST exit routine 121
mailbox processing
for JES abends 31
mapping macro
IXCYGEPL 16
IXZ$XPL
use with IXZXIT01 processing 44
use with IXZXIT02 processing 50
IXZYIXAC 24
IXZYIXEN 24
IXZYIXJE 16
IXZYIXPE 89
IXZYIXSE 24
message
acknowledgement
reroute using IXZXIT01 45
reroute using IXZXIT02 51
asynchronous with acknowledgement 20
asynchronous without acknowledgement 20
content
changing using IXZXIT01 45
changing using IXZXIT02 51
destination
change using IXZXIT01 44
envelope 24
extents
add using IXZXIT01 45
retrieve using IXZXIT02 52
mapping macros 24
receiving 23
sending 16
sending a multiple segment 21
sending a single segment 21
synchronous with acknowledgement 20
message *(continued)*
trace record 33
message processing
affect using JES XCF 5
for JES 10
module footprint
trace record 32
monitoring of XCF events
requesting 16
MSGTRC trace record 33
multi-segment message
abort 22
discontinue 22
send 21

N

navigation
keyboard 125

P

parameter list description
for IXZXIT01 47
for IXZXIT02 52
for IXZXIT03 60
for POST exit routine 122
performance delays
due to JES3 processing 27
POLYJES
IXZXIXIF parameter 86
POST exit routine
coding requirements 16, 121
entry specifications 122
environment 121
identifying 16
installing 121
parameter list description 122
processing 122
programming considerations 122
recovery 121
registers
entry 122
exit 123
return codes 123
printing macro expansions
control with ZPRINT 65
programming considerations
for IXZXIT01 47
for IXZXIT02 52
for IXZXIT03 59
for POST exit routine 122

R

RACF
profile for JES XCF 6
security for JES XCF 6
started procedures table 6
receipt of a message
acknowledging 25
receive exit
IXZXIT02 49

receiving a message 23
recovery
  for IXZXIT01 43
  for IXZXIT02 50
  for IXZXIT03 55
  for mailbox POST exit routine 121
retrieve JES XCF group token 13
retrieve message extents
  using IXZXIT02 52
RMF
  collecting performance data 27

S

send a message in JES environments 16
sending to IBM
  reader comments xiii
shortcut keys 125
single-segment message
  sending 21
STATE
  IXZXIXIF parameter 85
summary of changes
  <MVS and JES> xv
synchronous message
  with acknowledgement 20
SYSJES component trace id 32
sysplex
  relationship to JES XCF group 1
  relationship to JES2 MAS 1
  relationship to JES3 complex 1
SYSTEM
  IXZXIXIF parameter 85
system cleanup
  indicate status to JES XCF 30
system event message
  IXZYIXSE mapping macro 16

T

trace record
  FLOW (module footprint) 32
  for JES XCF 32
  installation exit (USRXIT) 33
  module footprint (FLOW) 32
  MSGTRC (message) 33
  SYSJES COMP id 32
  USRXIT (installation exit) 33
  XCF and JES XCF event (XCFEVT) 33
  XCFEVT (system event) 33
trademarks 132
transport exit
  IXZXIT01 43

U

user exit trace
  see installation exit trace 33
user interface
  ISPF 125
  TSO/E 125
uses for JES XCF 5
USRXIT
USRXIT *(continued)*
  installation exit trace 33

X

XCF and JES XCF event
  trace record 33
XCF event message
  requesting 16
XCFEVT trace record 33

Z

ZPRINT
  to control macro expansion printing 65

SA23-1387-40