

z/OS



MVS Programming: JES Common Coupling Services

z/OS



MVS Programming: JES Common Coupling Services

Note

Before using this information and the product it supports, be sure to read the general information under Appendix B, "Notices" on page 141.

First Edition, March 2001

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

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

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

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

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

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrdfs@us.ibm.com
World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zos/webqs.html>

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

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

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

Contents

About This Book	xi
Who Should Use This Book	xi
How to Use This Book	xi
Programming Books for Assembler Language Programs	xi
Programming Books for High-Level Language Programs	xv
Programming Books for Both Assembler and High-Level Language Programs	xv
Where to Find More Information	xvi
Accessing licensed books on the Web	xvii
Using LookAt to look up message explanations	xvii
Additional Information	xviii
Determining If a Publication Is Current	xviii
Programming Interface Information	xix
Chapter 1. Introduction to the JES Common Coupling Services	1
Your Installation's Goals - Do You Need to Use JES XCF Component Services?	3
JES Benefits from the Use of the Common Coupling Services Component	4
How You Can Affect JES Message Processing	5
Basic JES XCF Terms and Concepts	6
Providing Security for JES XCF	6
Macro Overview	8
Exit Overview	9
Chapter 2. Using the JES XCF Exits and Macros	11
JES and JES XCF Processing Overview	11
JES Initialization Processing	11
JES Message Processing	13
JES Termination Processing	14
Using JES XCF Services	15
The JES-Defined Attachment to JES XCF	16
Determining If You Should Use the JES-Defined Attachment	16
Retrieving the JES XCF Group Token	17
Creating an Installation-Defined Attachment to JES XCF (IXZXIXAT Macro)	17
Building a Mailbox (IXZXIXMB Macro)	18
Mailbox Availability	19
Identifying the Installation-Written POST Exit Routine	20
Requesting XCF Event Monitoring	20
Sending a Message (IXZXIXSM Macro)	21
Recovery Levels	21
Sending an Asynchronous Message Without Acknowledgement or Recovery	24
Sending an Asynchronous Message With Acknowledgement	24
Sending an Asynchronous Message Without Acknowledgement	25
Sending a Synchronous Message	25
Identifying Which Mailbox is to Receive the Acknowledgement	26
Message Segments	26
Providing a Place for Data Returned from the Message Receiver	27
Identifying Which Member is to Receive the Message	27
Restrictions and IBM Recommendations	27
Receiving a Message (IXZXIXRM Macro)	28
Mapping Messages	29
Restrictions	30

Acknowledging Receipt of a Message (IXZXIXAC Macro)	30
Obtaining Information About JES Members in the XCF Group (IXZXIXIF Macro)	31
Collecting JES3 Performance Information Data (IXZXIXPI Macro)	33
Indicating that System Cleanup is Complete (IXZXIXCL Macro)	35
Clearing a Mailbox of Messages (IXZXIXMC Macro)	35
Deleting a Mailbox (IXZXIXMD Macro)	35
System Return and Reason Codes When Messages are Undelivered	36
What Happens to Mailboxes When the JES Session Ends	36
Detaching from a JES XCF Group (IXZXIXDT Macro)	36
Component Trace and Diagnostic Support	37
Controlling the Quantity of Traced Data	38
Obtaining Trace Data	38
Module Footprint Tracing (FLOW Sub-Level)	38
XCF and JES XCF Event Tracing (XCFEVT Sub-Level)	38
Detailed Message Tracing (MSGTRC Sub-Level)	38
Installation Exit Tracing (USRXIT Sub-Level)	39
Ending and Restarting the JESXCF Address Space	39
Chapter 3. Coding the JES XCF Exits	43
Creating Your Own Attachment to JES XCF	44
Determining Your Need to Create Your Own JES XCF Attachment	44
Defining a JES XCF Group	44
Using the JES XCF Exits	45
IXZXIT01 - Transport Exit	51
Installing the Exit Routine	51
Exit Routine Environment	51
Exit Routine Processing	51
Processing Multi-Segment Messages	55
Programming Considerations	56
Entry Specifications	56
Return Specifications	57
Coded Example of Exit Routine	58
IXZXIT02 - Receive Exit	59
Installing the Exit Routine	59
Exit Routine Environment	59
Exit Routine Processing	59
Programming Considerations	62
Entry Specifications	62
Return Specifications	63
Coded Example of Exit Routine	64
IXZXIT03 - Attach/Detach Exit	65
Installing the Exit Routine	65
Exit Routine Environment	65
Exit Routine Processing	66
Programming Considerations	71
Entry Specifications	71
Return Specifications	72
Coded Example of Exit Routine	72
Chapter 4. JES XCF Macro Reference	73
How to Read a Syntax Diagram	73
Controlling Macro Expansion Printing	75
IXZXIXAC - Acknowledge Receipt of a Message	76
Environment	76

Programming Requirements	76
Restrictions	76
Input Register Information	76
Output Register Information	76
Performance Implications	77
Syntax	77
Parameters	77
ABEND Codes	79
Return and Reason Codes	79
IXZXIXAT - Attach to a JES XCF Group	81
Environment	81
Programming Requirements	81
Restrictions	81
Input Register Information	82
Output Register Information	82
Performance Implications	82
Syntax	82
Parameters	83
ABEND Codes	85
Return and Reason Codes	85
IXZXIXCL - System Cleanup Initiated Indicator	86
Environment	86
Programming Requirements	86
Restrictions	86
Input Register Information	86
Output Register Information	87
Performance Implications	87
Syntax	87
Parameters	87
ABEND Codes	89
Return and Reason Codes	89
IXZXIXDT - Detach from a JES XCF Group	90
Environment	90
Programming Requirements	90
Restrictions	90
Input Register Information	90
Output Register Information	91
Performance Implications	91
Syntax	91
Parameters	91
ABEND Codes	92
Return and Reason Codes	93
IXZXIXIF - Obtain Information About Members of an XCF Group	94
Environment	94
Programming Requirements	94
Restrictions	94
Input Register Information	94
Output Register Information	95
Performance Implications	95
Syntax	95
Parameters	95
ABEND Codes	99
Return and Reason Codes	99
IXZXIXMB - Build a Mailbox	101

Environment	101
Programming Requirements	101
Restrictions	101
Input Register Information	101
Output Register information	102
Performance Implications	102
Syntax	102
Parameters	102
ABEND Codes	105
Return and Reason Codes	105
IXZXIXMC - Clear a Mailbox	107
Environment	107
Programming Requirements	107
Restrictions	107
Input Register Information	107
Output Register Information	108
Performance Implications	108
Syntax	108
Parameters	108
ABEND Codes	110
Return and Reason Codes	110
IXZXIXMD - Delete a Mailbox	112
Environment	112
Programming Requirements	112
Restrictions	112
Input Register Information	112
Output Register Information	113
Performance Implications	113
Syntax	113
Parameters	113
ABEND Codes	115
Return and Reason Codes	115
IXZXIXPI - Collect JES3 Performance Information Data	117
Environment	117
Programming Requirements	117
Restrictions	117
Input Register Information	117
Output Register Information	117
Performance Implications	118
Syntax	118
Parameters	118
ABEND Codes	120
Return and Reason Codes	120
IXZXIXRM - Receive a Message	121
Environment	121
Programming Requirements	121
Restrictions	122
Input Register Information	122
Output Register Information	122
Performance Implications	122
Syntax	123
Parameters	123
ABEND Codes	125
Return and Reason Codes	125

IXZXISM - Send a Message	127
Environment	127
Programming Requirements	127
Restrictions	127
Input Register Information	128
Output Register Information	128
Performance Implications	128
Syntax	129
Parameters	129
ABEND Codes	133
Return and Reason Codes	133
Appendix A. POST Exit Routine	137
Installing the Exit Routine	137
Exit Routine Environment	137
Exit Recovery	137
Exit Routine Processing	137
Programming Considerations	138
Entry Specifications	138
Registers at Entry	138
Parameter List Contents	138
Return Specifications	138
Registers at Exit	138
Return Code Meanings	139
Appendix B. Notices	141
Trademarks	142
Glossary	145
Index	147

Figures

1.	Referenced Publications	xvi
2.	A JES XCF Group and Its Relationship to an MVS Sysplex	2
3.	JES XCF Tasks Based on Your Need to Modify JES XCF Communication	3
4.	JES Communication Mechanism	4
5.	JES Common Coupling Services Macros	8
6.	JES Common Coupling Services Exits	9
7.	JES Initialization Processing and Its Relationship to the JES XCF Processing	12
8.	JES Message Processing and Its Relationship to JES XCF Processing	13
9.	JES Termination Processing and Its Relationship to JES XCF Processing	15
10.	JES Control Block from Which to Retrieve the JES XCF Group Token	17
11.	Request Type Characteristics for Send Message Macro	21
12.	Send Message Example Showing Levels of Recovery	22
13.	Example Synchronous Message Request	25
14.	Required Parameter Token Definitions for SEGTYPE and REQTYPE	27
15.	JES XCF Message Mapping Macro Structure	30
16.	Array Format of Data Returned by IXZXIXIF	33
17.	Example JES3 Performance Data Returned by IXZXIXPI	34
18.	System Return and Reason Codes	36
19.	JES XCF Trace Structure	37
20.	Example of Using JES XCF Macros and Exits for an ASYNCAK Message Cycle	45
21.	Access Path From Exit Caller to Message Data	52
22.	IXZ\$XPL Mapping of Message Data and Extents	55
23.	Access Path From Exit Caller to Message Data	60
24.	Example Attach (IXZXIT03) Code	67
25.	Exit IXZXIT03 Attach/Detach Processing	69

About This Book

This book 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 Book

This book 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 books:

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

How to Use This Book

This book 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.

Programming Books for Assembler Language Programs

Book Title	Use this book to:	Order Number
<i>z/OS MVS Programming: Assembler Services Guide</i>	Find out how to use system services provided by macros and callable services available to all assembler language programs. If you are relatively new to assembler language programming, this book is a good place to start. For information about using resource recovery services in an application, see <i>z/OS MVS Programming: Callable Services for HLL</i> .	SA22-7605
<i>z/OS MVS Programming: Assembler Services Reference ABE-HSP</i>	Learn how to code macros and callable services that are available to all assembler language programs. This book is for all assembler language programmers. For information about using resource recovery services in an application, see <i>z/OS MVS Programming: Callable Services for HLL</i> .	SA22-7606
<i>z/OS MVS Programming: Authorized Assembler Services Guide</i>	Find out how to use system services provided by macros and callable services that are available to programs running in supervisor state or with PSW key 0-7 or that are APF-authorized programs. This book is for experienced assembler language programmers; it assumes, for example, that you are familiar with the information in <i>z/OS MVS Programming: Assembler Services Guide</i> .	SA22-7608

Book Title	Use this book to:	Order Number
<p><i>z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN</i></p> <p><i>z/OS MVS Programming: Authorized Assembler Services Reference ENF-IXG</i></p> <p><i>z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU</i></p> <p><i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i></p>	<p>Learn how to code macros and callable services that are available to programs running in supervisor state or with PSW key 0-7 or that are APF-authorized programs. This book is for experienced assembler language programmers.</p> <p>For descriptions of authorized resource recovery services for MVS services (ATRxxxx), authorized registration services (CRGxxxx), and authorized context services (CTXxxxx), see <i>z/OS MVS Programming: Resource Recovery</i>.</p>	<p>SA22-7609</p> <p>SA22-7610</p> <p>SA22-7611</p> <p>SA22-7612</p>
<p><i>z/OS MVS Programming: Extended Addressability Guide</i></p>	<p>Find out how to extend or restrict the storage available to programs through the use of access registers, cross memory services, data spaces, subspaces, and hiperspaces. This book is for experienced assembler language programmers.</p>	<p>SA22-7614</p>
<p><i>z/OS MVS Programming: JES Common Coupling Services</i></p>	<p>Learn how to use the JES common coupling services and exits to affect JES communication processing in an MVS sysplex environment. This book is for experienced JES2, JES3, and BCP assembler language system programmers.</p>	<p>SA22-7615</p>
<p><i>z/OS MVS Programming: Sysplex Services Guide</i></p>	<p>Find out how to use authorized system services to enable your multisystem application or subsystem to run in a sysplex, benefit from automatic restarts, and share data using the coupling facility. This book is for experienced assembler language programmers.</p>	<p>SA22-7617</p>

Book Title	Use this book to:	Order Number
<i>z/OS MVS Programming: Sysplex Services Reference</i>	Learn how to code authorized system services to enable your multisystem application or subsystem to run in a sysplex, benefit from automatic restarts, and share data using the coupling facility. This book is for experienced assembler language programmers.	SA22-7618
<i>z/OS MVS Programming: Workload Management Services</i>	Find out how to use workload management services that are intended for subsystem work manager and performance monitoring programs.	SA22-7619
<i>MVS Batch Local Shared Resources</i>	Find out whether your installation's batch applications can benefit from batch LSR subsystem and how to use batch LSR subsystem. Using this book does not require a knowledge of assembler language programming.	GC28-1469
<i>MVS Hiperbatch Guide</i>	Find out whether your installation's batch applications can benefit from Hiperbatch and how to use Hiperbatch. Using this book does not require a knowledge of assembler language programming.	GC28-1470

Programming Books for High-Level Language Programs

Book Title	Use this book to:	Order Number
<i>z/OS MVS Programming: Callable Services for HLL</i>	Find out how to use and code program CALLs for specific MVS services in high-level language (HLL) programs.	SA22-7613

Programming Books for Both Assembler and High-Level Language Programs

Book Title	Use this book to:	Order Number
<i>z/OS MVS Programming: Product Registration</i>	Learn how to use registration services to register a product or to prepare an optional z/OS element dynamic enablement. This book is for assembler or C language programmers.	SA22-7604
<i>z/OS MVS Programming: Resource Recovery</i>	Learn how to use resource recovery services for MVS services, context services, and registration services in a resource manager, an authorized program that provides access to protected resources. The resource manager, which is not the same as an RTM resource manager that runs during termination, works with an application program and resource recovery services for MVS to protect resources through a two-phase commit protocol. This book is for experienced programmers.	SA22-7616
<i>z/OS MVS Programming: Writing Transaction Programs for APPC/MVS</i>	Find out how to use and code program CALLs for APPC/MVS communication services that are intended for both APPC/MVS transaction programs and servers. This book is a companion to <i>z/OS MVS Programming: Writing Servers for APPC/MVS</i> , which documents APPC/MVS services for servers only.	SA22-7621

Book Title	Use this book to:	Order Number
<i>z/OS MVS Programming: Writing Servers for APPC/MVS</i>	Find out how to use and code program CALLs for APPC/MVS communication services that are intended for servers only. This book is a companion to <i>z/OS MVS Programming: Writing Transaction Programs for APPC/MVS</i> , which documents the APPC/MVS services for both transaction programs and servers.	SA22-7620
<i>z/OS MVS System Messages, Vol 3 (ASB-BPX)</i>	Find out how to use and code program CALLs for APPC/MVS system services that are intended for APPC/MVS transaction schedulers. This book is for programmers who write transaction schedulers other than the one APPC/MVS provides.	SA22-7633

Where to Find More Information

This book references the following publications for further details about specific topics. This following table lists the titles used throughout the book, their full titles, and their order numbers that are not listed in *z/OS Information Roadmap*. Refer to that book for all z/OS publications.

<i>Figure 1. Referenced Publications</i>		
Title Used in This Book	Title	Order Number
<i>American National Standard Vocabulary for Information Processing</i>	<i>American National Standard Vocabulary for Information Processing</i>	N/A
<i>Assembler H Version 2 Application Programming Guide</i>	<i>Assembler H Version 2 Application Programming Guide</i>	SC26-4036
<i>Assembler H Version 2 Application Programming: Language Reference</i>	<i>Assembler H Version 2 Application Programming: Language Reference</i>	SC26-4037
<i>HLASM Language Reference</i>	<i>High Level Assembler for MVS & VM & VSE Language Reference</i>	SC26-4940
<i>HLASM Programmer's Guide</i>	<i>High Level Assembler for MVS & VM & VSE Programmer's Guide</i>	SC26-4940
<i>z/OS Parallel Sysplex Application Migration</i>	<i>z/OS Parallel Sysplex Application Migration</i>	SC22-7662

Accessing licensed books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile**.
4. Click on **Request Access to Licensed books**.
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS**.
6. Access the licensed book by selecting the appropriate element.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages and system abends.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

You can use LookAt on the Internet at:

<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html>

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Web site by clicking on **News and Help** or from the *z/OS Collection*, SK3T-4269.

To find a message explanation from a TSO command line, simply enter: **lookat message-id** as in the following example:

```
lookat iec192i
```

This results in direct access to the message explanation for message IEC192I.

To find a message explanation from the LookAt Web site, simply enter the message ID. You can select the release if needed.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

Additional Information

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

Title	Order Number	Description
<i>z/OS Introduction and Release Guide</i>	GA22-7502	Describes the contents and benefits of z/OS as well as the planned packaging and delivery of this new product.
<i>z/OS Planning for Installation</i>	GA22-7504	Contains information that lets users: <ul style="list-style-type: none"> • Understand the content of z/OS • Plan to get z/OS up and running • Install the code • Take the appropriate migration actions • Test the z/OS system
<i>z/OS Information Roadmap</i>	SA22-7500	Describes the information associated with z/OS including z/OS books and books for the participating elements.
<i>z/OS Summary of Message Changes</i>	SA22-7505	Describes the changes to messages for individual elements of z/OS. Note: This book is provided in <u>softcopy only</u> on the message bookshelf of the z/OS collection kit (SK3T-4269).

Determining If a Publication Is Current

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. Here's how to determine if you are looking at the most current copy of a publication:

1. At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.

2. If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
3. To compare softcopy publications, you can check the last two characters of the publication's filename (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk that indicates whether a publication is new or changed.

Programming Interface Information

This book is intended to help the system programmer to use the JES common coupling services of z/OS MVS. This book primarily documents General-Use Programming Interfaces and Associated Guidance Information provided by z/OS MVS.

General-use programming interfaces allow the customer to write programs that obtain the services of z/OS MVS.

However, this book also documents Product-sensitive Programming Interface and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by z/OS MVS.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of z/OS MVS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service. Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

_____ Product-sensitive programming interface _____

Product-sensitive Programming Interface and Associated Guidance Information...

_____ End of Product-sensitive programming interface _____

Diagnosis, Modification or Tuning information is provided to help the customer diagnose z/OS MVS.

Attention: Do not use this Diagnosis, Modification or Tuning Information as a programming interface.

Diagnosis, Modification or Tuning information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

_____ Product-sensitive programming interface _____

Diagnosis, Modification or Tuning Information...

_____ End of Product-sensitive programming interface _____

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 2. Configuring your JES subsystems and your MVS systems in this way will ease your overall sysplex management task.

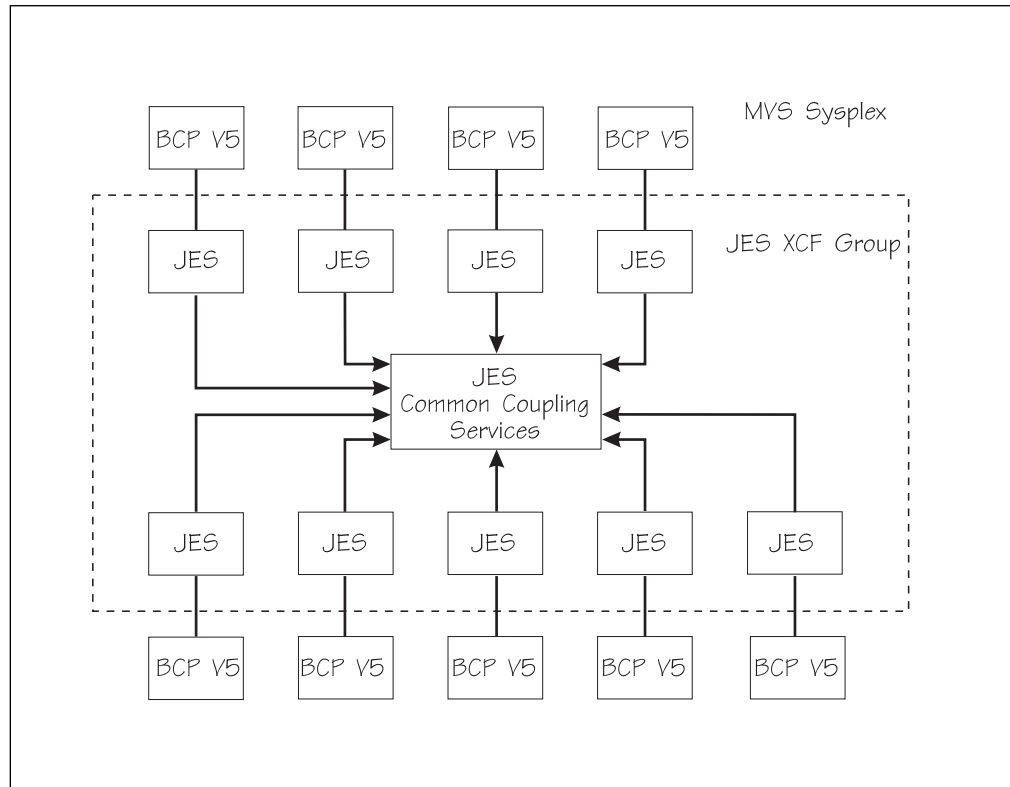


Figure 2. A JES XCF Group and Its Relationship to an MVS Sysplex

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 book. 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. Figure 3 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.

<i>Figure 3. JES XCF Tasks Based on Your Need to Modify JES XCF Communication</i>	
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 4.
Sends and receives messages within standard JES exits or JES3 DSPs	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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 book 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 4 shows a simplistic view of a JES member-to-member link as provided by the XCF and the JES XCF.

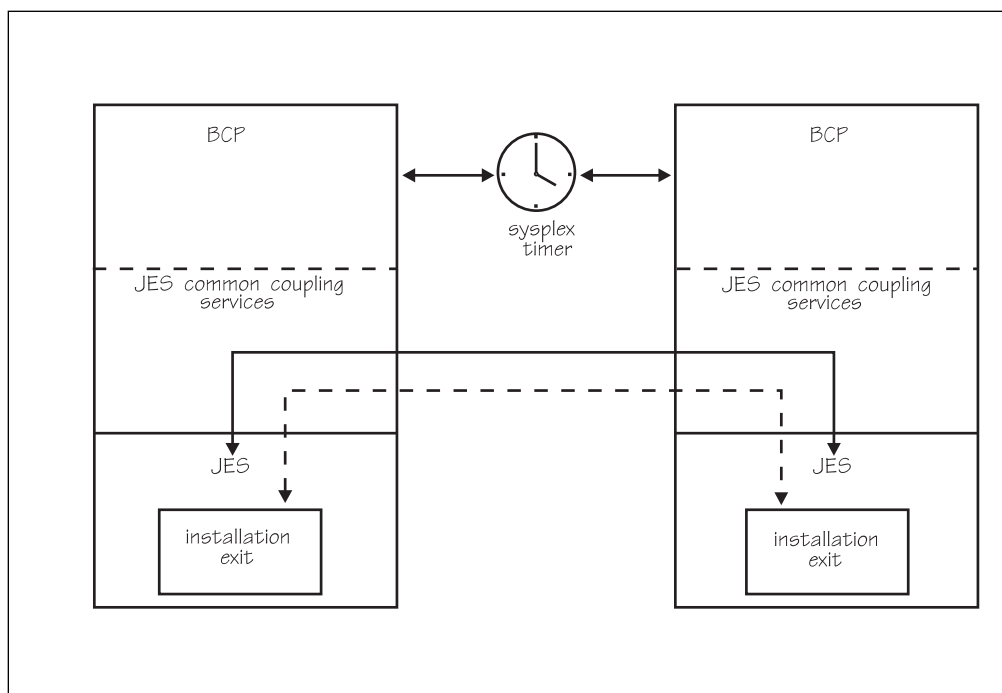


Figure 4. JES Communication Mechanism

There are several ways your installation might choose 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 *z/OS Parallel Sysplex Application Migration*, SC22-7662, 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 43 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 book 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 details the use of the JES XCF exits and macros to performs specific tasks, and Chapter 3, Coding the JES XCF Exits 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 SecureWay 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

Figure 5 lists the set of executable macros that JES XCF provides. Each is fully discussed in Chapter 4, “JES XCF Macro Reference” on page 73.

Figure 5. JES Common Coupling Services Macros

Macro Function	Macro Name	Used In	Used In	Page
		JES XCF Exit	Standard JES Exit	
Acknowledge receipt of a JES XCF message	IXZXIXAC		X	76
Attach to JES XCF	IXZXIXAT	X		81
Build a JES XCF mailbox	IXZXIXMB		X	101
Clear a JES XCF mailbox	IXZXIXMC		X	107
Collect JES3 performance information data	IXZXIXPI		X	117
Delete a JES XCF mailbox	IXZXIXMD		X	112
Detach from JES XCF	IXZXIXDT	X		90
Obtain member information	IXZXIXIF		X	94
Receive a message from a JES XCF mailbox	IXZXIXRM		X	121
Send a message to a JES XCF mailbox	IXZXIXSM		X	127
System cleanup initiated indication	IXZXIXCL		X	86

Exit Overview

Figure 6 lists the IBM-defined exits that JES XCF provides. Each is fully discussed in Chapter 3, “Coding the JES XCF Exits” on page 43. Also, a sample of each exit is provided in SYS1.SAMPLIB, members IXZEX01A, IXZEX02A, and IXZEX03A, respectively.

Figure 6. JES Common Coupling Services Exits

Exit Name	Exit Function	Purpose	Page
IXZXIT01	Transport (send)	View, modify, reroute a message or acknowledgement before the message arrives at the receiving member's mailbox.	51
IXZXIT02	Receive	View or modify a message before it is retrieved from a mailbox.	59
IXZXIT03	Attach/Detach	<ul style="list-style-type: none">• 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.	65

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 and Chapter 4, JES XCF Macro Reference 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 11, “JES Message Processing” on page 13, and “JES Termination Processing” on page 14. 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 65 offers a more complete description of JES XCF processing and Figure 24 on page 67 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 7 for a graphic view of how IXZXIXAT processing relates to overall JES processing.

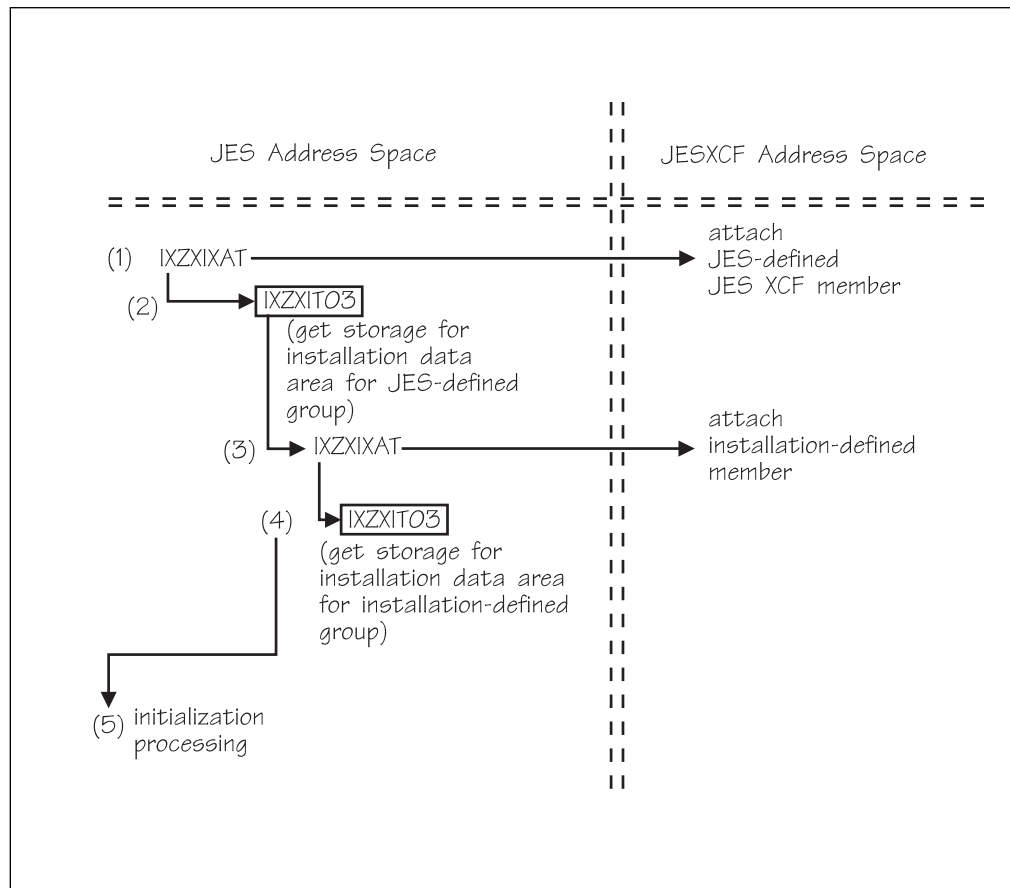


Figure 7. JES Initialization Processing and Its Relationship to the JES XCF Processing

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 8 for a graphic view of message processing.

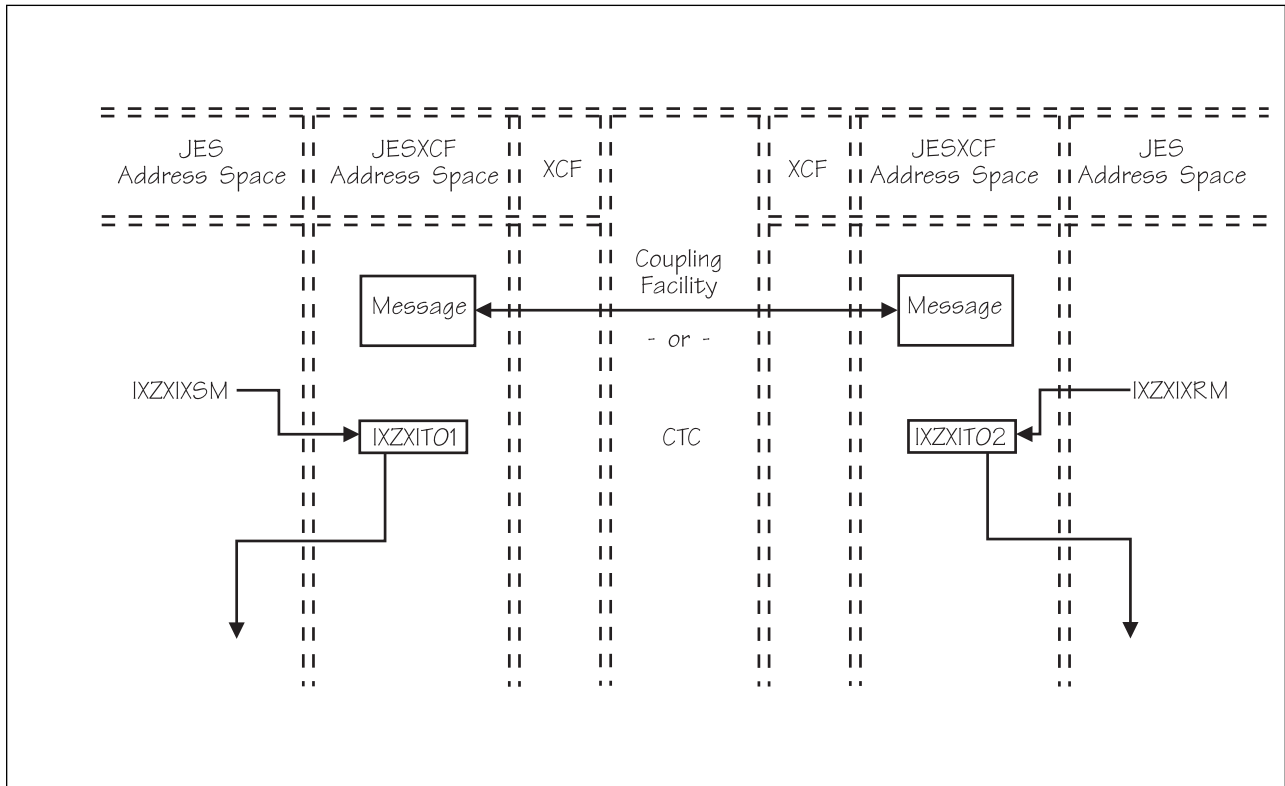


Figure 8. JES Message Processing and Its Relationship to JES XCF Processing

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 9 on page 15. 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.

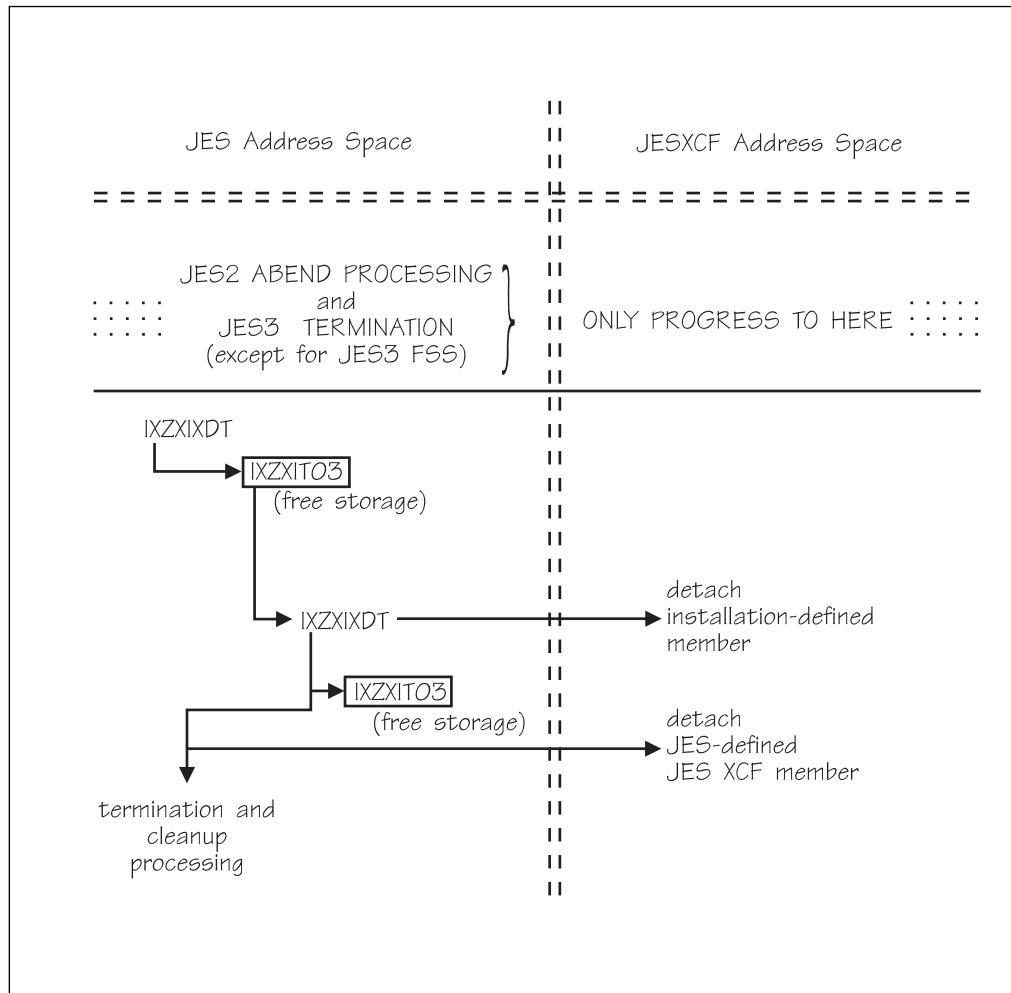


Figure 9. JES Termination Processing and Its Relationship to JES XCF Processing

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 16 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.

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 Figure 10 on page 17. 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 17. Depending upon the JES under which you are running, retrieve the JES XCF token from the appropriate field as follows:

<i>Figure 10. JES Control Block from Which to Retrieve the JES XCF Group Token</i>	
JES	Field Name in Data Area
JES2	<ul style="list-style-type: none"> • \$XCFIXVT in \$HCT • CCTIXVT in \$HCCT
JES3	<ul style="list-style-type: none"> • SVTJXGT in IATYSVT • TVTXJXGT in IATYTVTX
JES3 FSS	<ul style="list-style-type: none"> • 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 of Product-sensitive programming interface

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 65. 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 IXZXIMB 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 17, 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 137 for instructions for coding a POST exit routine.

Requesting XCF Event Monitoring

When you build a mailbox, you have the option to request that you 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. Refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for the IXZYIXEN mapping and the mappings for the other mapping macros discussed here. Also refer to Figure 15 on page 30 for a pictorial view of the message envelope (IXZYIXEN mapping macro), and a system event's message structure.

Sending a Message (IXZXISM Macro)

JDUs running in any JES environment can use the JES XCF macro IXZXISM 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 26 for an explanation of message segments.)

Simply stated, IXZXISM (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 IXZXISM macro. Refer to Figure 11 for a summary of the REQTYPE values and the message processing characteristics of each.

Figure 11. Request Type Characteristics for Send Message Macro

REQTYPE= Parameter	Synchronous Processing	Acknowledgement Requested from Receiver	Multisystem Recovery Provided
COMM	No	No	No
ASYNACK	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 12 on page 22 for an illustration of the levels.

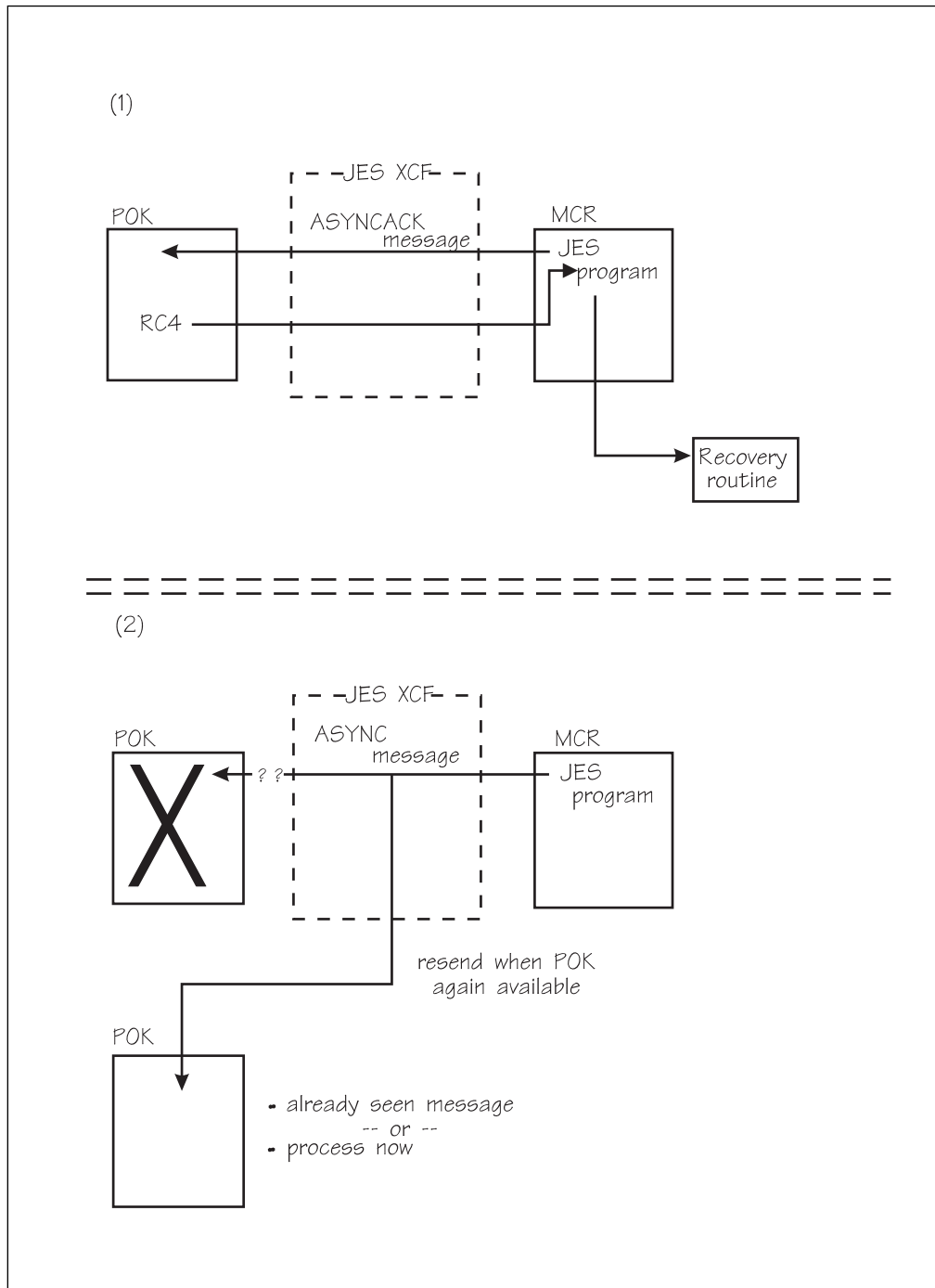


Figure 12. Send Message Example Showing Levels of Recovery

In Part **1** of Figure 12, 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 12 on page 22 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=ASYNACK 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 IXZXISM macro as follows:

COMM	No recovery
ASYNC	JES XCF recovery
ASYNCAACK	JES XCF recovery
SYNC	JES XCF recovery

Sending an Asynchronous Message Without Acknowledgement or Recovery

The default request type for the IXZXISM 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=ASYNCAACK on IXZXISM 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 IXZXIAC 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 ASYNCAACK Message

For ASYNCAACK messages only, you need to associate the acknowledgement with the original message. To do so, specify the REQTOKEN= parameter on the IXZXISM 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 IXZYIAC macro (the mapping macro that maps acknowledgements). Refer to “When a Sender Requires an Acknowledgement of a Multi-Segment Message” on page 27 for further discussion of REQTOKEN= for multi-segment messages.

Sending an Asynchronous Message Without Acknowledgement

Specify REQTYPE=ASYNCR on IXZXISM to indicate that the message being sent is sent asynchronously. This option is similar to REQTYPE=COMM, except that ASYNCR 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 IXZXISM 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 13, 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.

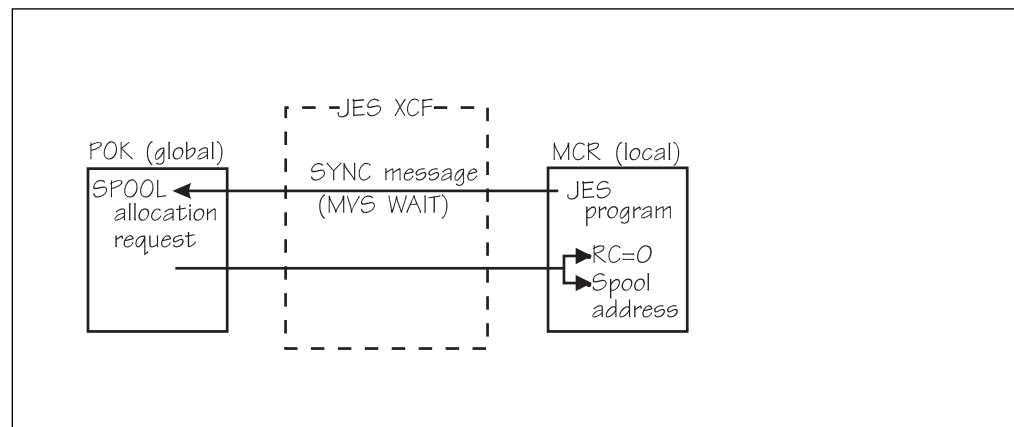


Figure 13. Example Synchronous Message Request

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 28 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=ASYNACK, 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. Figure 14 shows the required parameters for each SEGTYPE= based on the REQTYPE= call.

Figure 14. Required Parameter Token Definitions for SEGTYPE and REQTYPE

REQTYPE=	SEGTYPE= FIRST	SEGTYPE= MIDDLE	SEGTYPE= LAST	SEGTYPE= ABORT
SYNC	--	--	MSGTOKEN=	MSGTOKEN=
ASYNACK	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. ASYNCAACK 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 (IXZXIRM Macro)

JDU's 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 IXZXIMB POSTXIT= parameter of the IXZXISM 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 IXZXIRM macro. When you issue IXZXIRM, 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 IXZXIRM, 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 IXZXIRM 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 ASYNACK 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 ASYNACK 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 Figure 18 on page 36 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 15 on page 30 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 31 for a discussion of the IXZXIXIF macro and the data it returns.

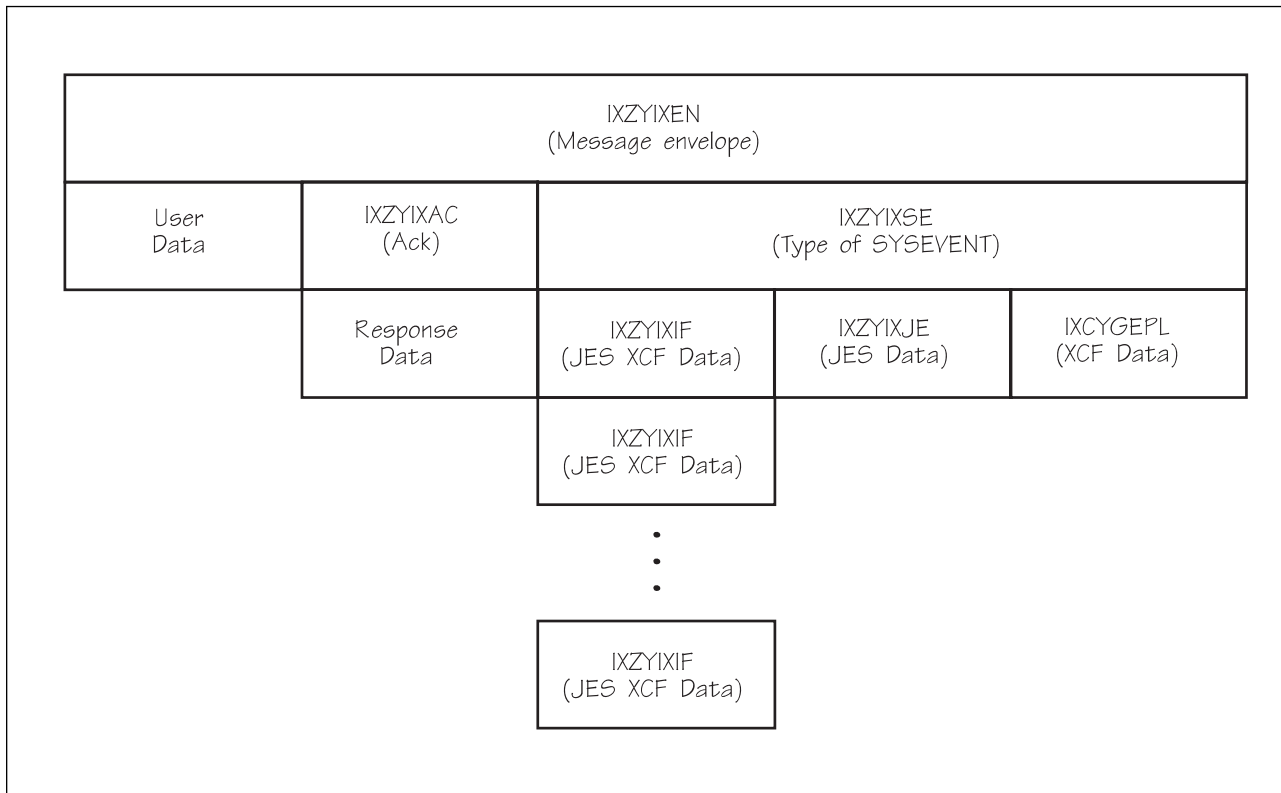


Figure 15. JES XCF Message Mapping Macro Structure

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=ASYNACK messages, to the mailbox specified by the REQMBX= parameter of the IXZXIXSM macro

- For REQTYPE=SYNC messages, to the address as specified by the RESPDATA= parameter of the IXZXISM macro.

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

To acknowledge a message, issue the IXZXIAC macro. When you issue IXZXIAC, 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 IXZXIAC. 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 (IXZXIIF 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 IXZXIIF macro. When you issue IXZXIIF, 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).

IXZXIIF 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 IXZXIIF 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:

IXZXIIF Parameter Limit data based on whether member is:

STATE=	Active (a member on an <i>active</i> MVS system only)
SYSTEM=	Current (a member on the MVS system from which this IXZXIIF 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 IXZYIXIF 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 IXZYIXIF 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 IXZIXRM (receive message) macro. If you choose to have the data returned in a buffer, specify the ANSAREA= and ANSLLEN= parameters on IXZYIXIF 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 ANSLLEN= parameter. You must then free the incorrectly sized buffer, obtain one of the required size, and invoke IXZIXRM again. The data IXZIXRM returns in the buffer is available in an array format as shown in Figure 16 on page 33. 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 IXZIXRM call. Performance can suffer because IXZIXRM requires I/O, therefore, issue IXZIXRM sparingly. When you do issue IXZIXRM, 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.

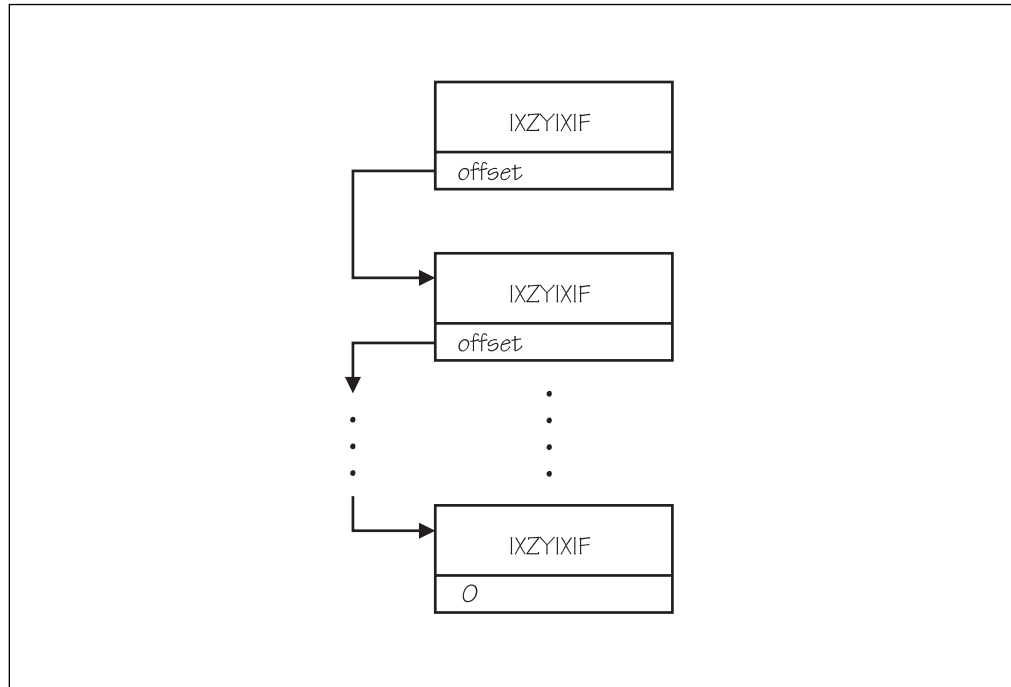


Figure 16. Array Format of Data Returned by IXZYIXIF

Product-sensitive programming interface

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 17 on page 34 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 17. Example JES3 Performance Data Returned by IXZXIXPI

The performance information data stream mapping macro, IXZYPIDS, maps the returned keys. Refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for a description of that mapping.

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. Refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* 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. (FAILED_SYS= 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 ASYNACK, 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 ASYNACK, 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 Figure 18.

Figure 18. 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.

Product-sensitive programming interface

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 19 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.

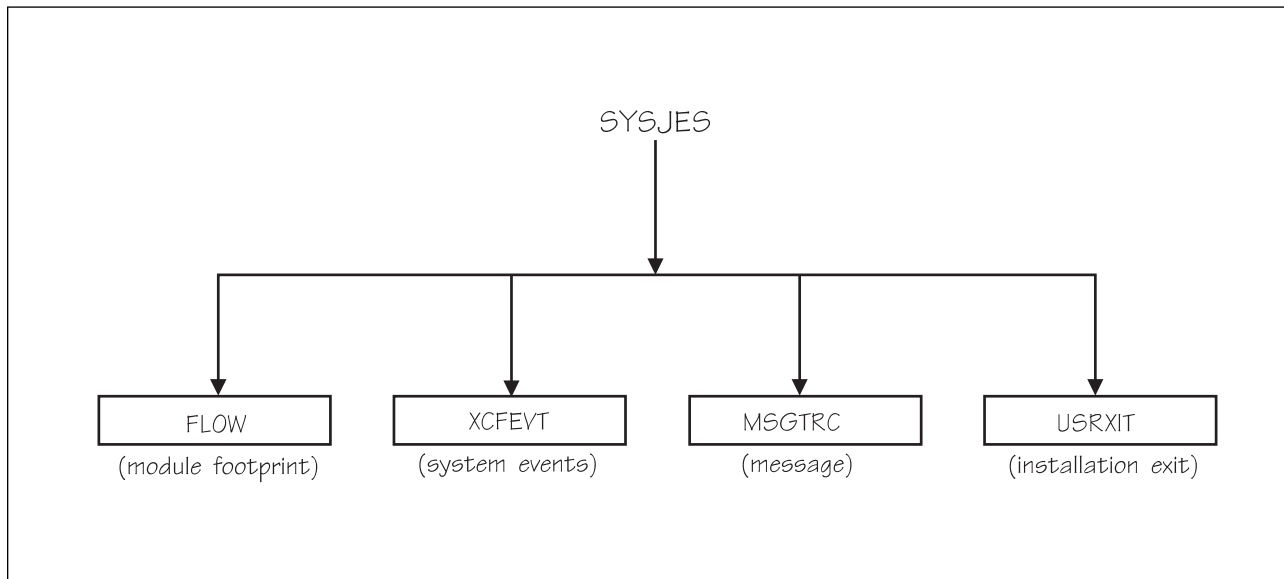


Figure 19. JES XCF Trace Structure

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*.

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.

_____ End of Product-sensitive programming interface _____

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 17.

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 20 on page 45, 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 20 is sample code for each step.

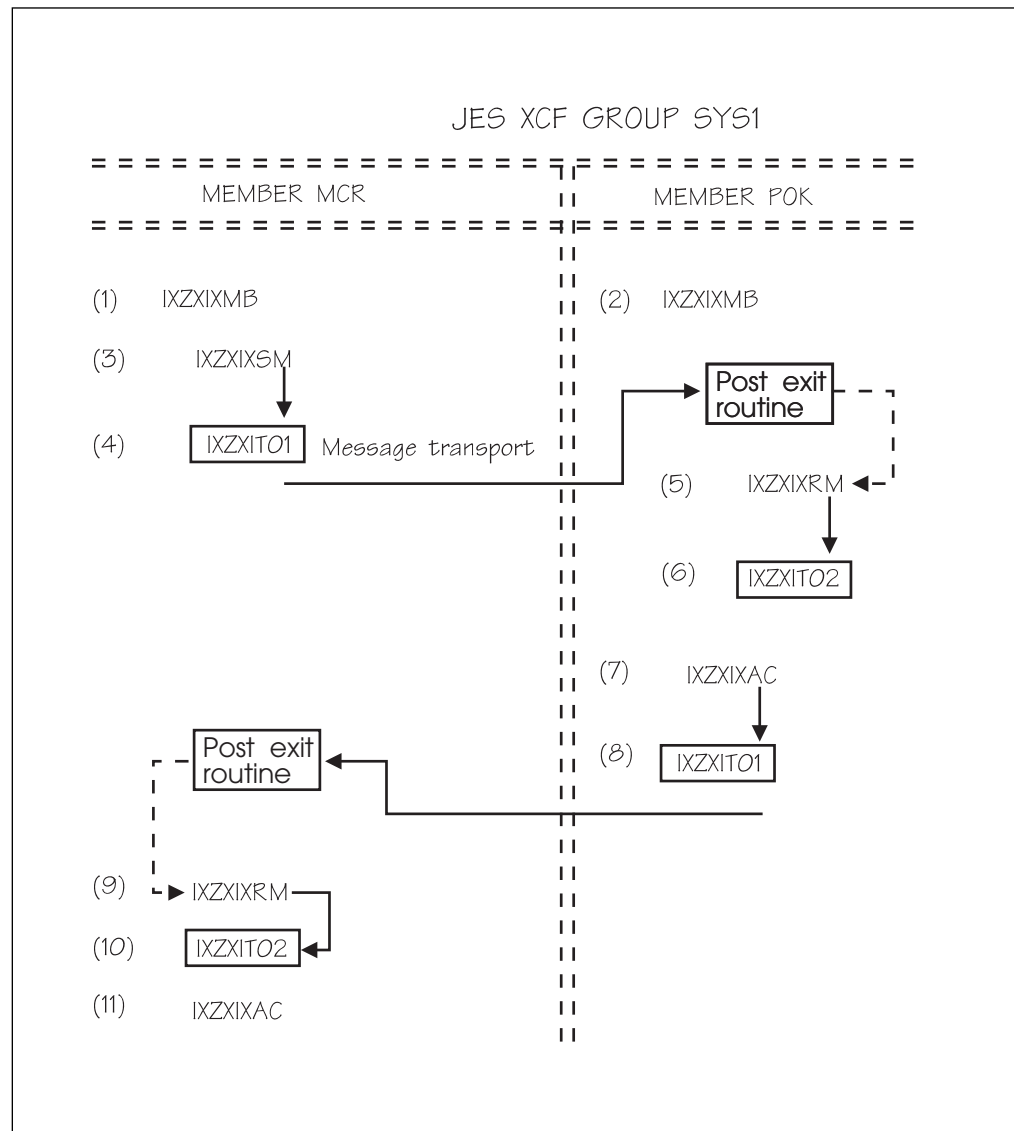


Figure 20. Example of Using JES XCF Macros and Exits for an ASYNACK Message Cycle

The following numbered steps describe the events in Figure 20.

- 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 ASYNACK 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 ASYNACK 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 ASYNACK 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=POKGTOK, X
        RTNCRD=RTNCRD, RSNCRD=RSNCRD
⋮
GROUP DC CL8'IBMVSLAB'
POK DC CL16'POK'
FMID DC CL8'HJE5510'
POKGTOK DS F
RTNCRD DC F'0'
RSNCRD DC F'0'
⋮
```

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

(1) Mailbox Build (IXZXIXMB) for MCR

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

Member *POK* builds mailbox *POKbMAILBOX*. 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=ASYNACK, REQTOKEN=TOKEN, REQ_MBOX=MCR_MBOX, X
      RTNCRD=RTNCRD, RSNCRD=RSNCRD
:

TOMESSAGE DC A(MESSAGE)
DLENGTH DC A(L'MESSAGE)
MESSAGE DC CL80'Message to process'
MCR_MBOX DC CL16'MCRbMAILBOX'
POK_MBOX DC CL16'POKbMAILBOX'
TOTOKEN DS XL8
RTNCRD DS F
RSNCRD DS F
:

```

Member *MCR* sends an ASYNACK message to member *POK*'s mailbox (*POKbMAILBOX*). 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 *MCRbMAILBOX*.

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

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

```

:
receive IXZXIXRM MBOXNAME=POK_MBOX, GROUPTOKEN=POKGTOK, X
        MSGTOKEN=MTOKEN, DATA=DATA, DATALEN=DLEN, ...
:
POK_MBOX DC CL16'POKbMAILBOX'
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 ASYNACK message from *POKbMAILBOX*. 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=POKGTOK,MSGTOKEN=MTOKEN, X  
        USERRC=USERRC,...  
⋮  
USERRC DC F'0'  
⋮
```

Member *POK* acknowledges the *ASYNACK* 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=MCRGTOK, 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=MCRGTOK,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 linked 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 ASYNACK). 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 21:

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.

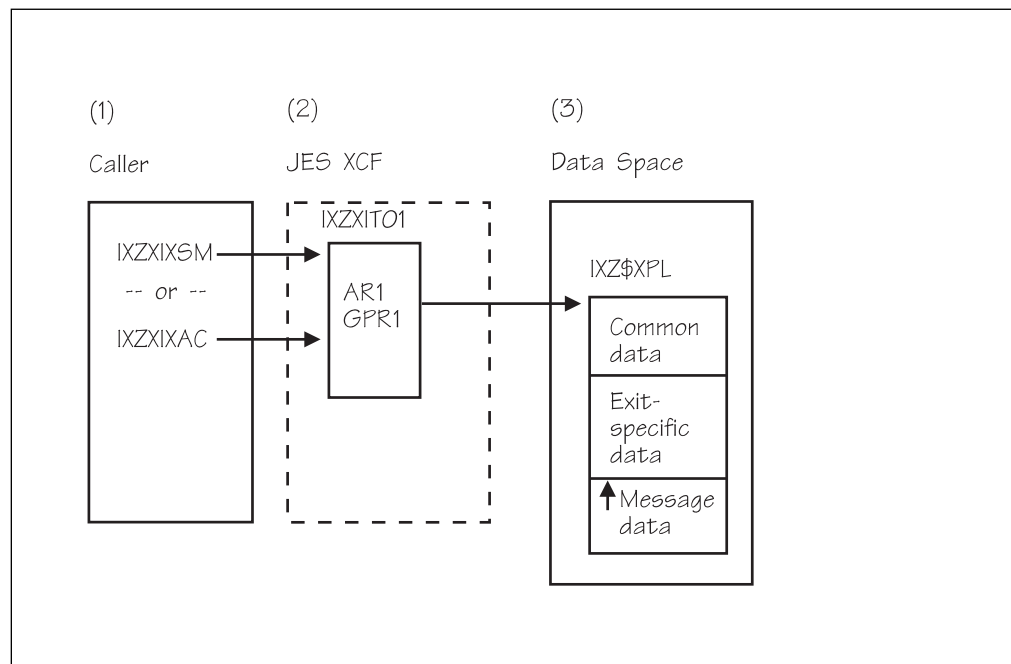


Figure 21. Access Path From Exit Caller to Message Data

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 REQMBX= 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_UPLLEN
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 22 on page 55 shows the IXZ\$XPL counters, XIT01_MESSAGE_UPLEN and XIT01_MAX_ADD.

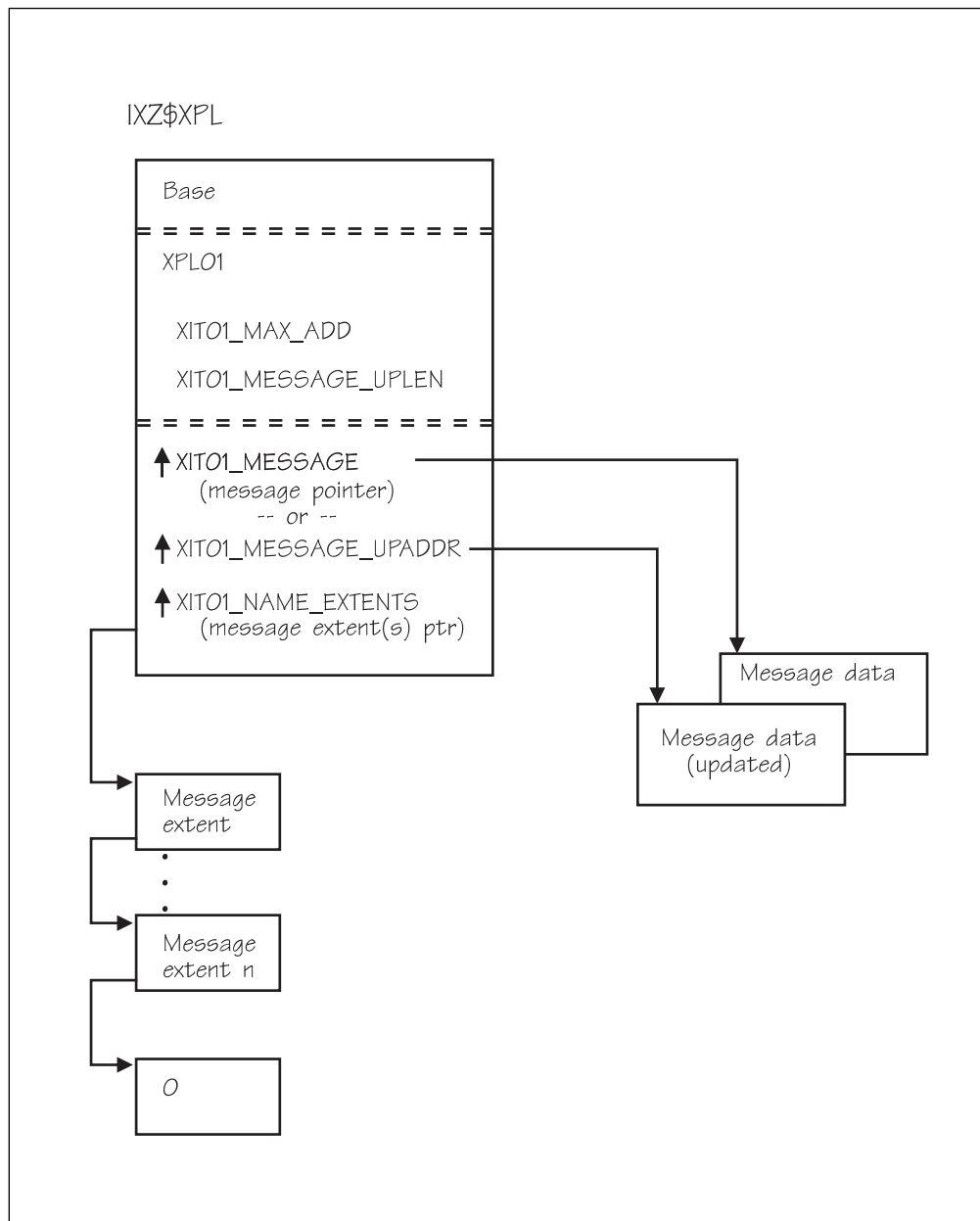


Figure 22. IXZ\$XPL Mapping of Message Data and Extents

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, refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

The parameter list consists of three sections:

- 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 linked 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 23:

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.

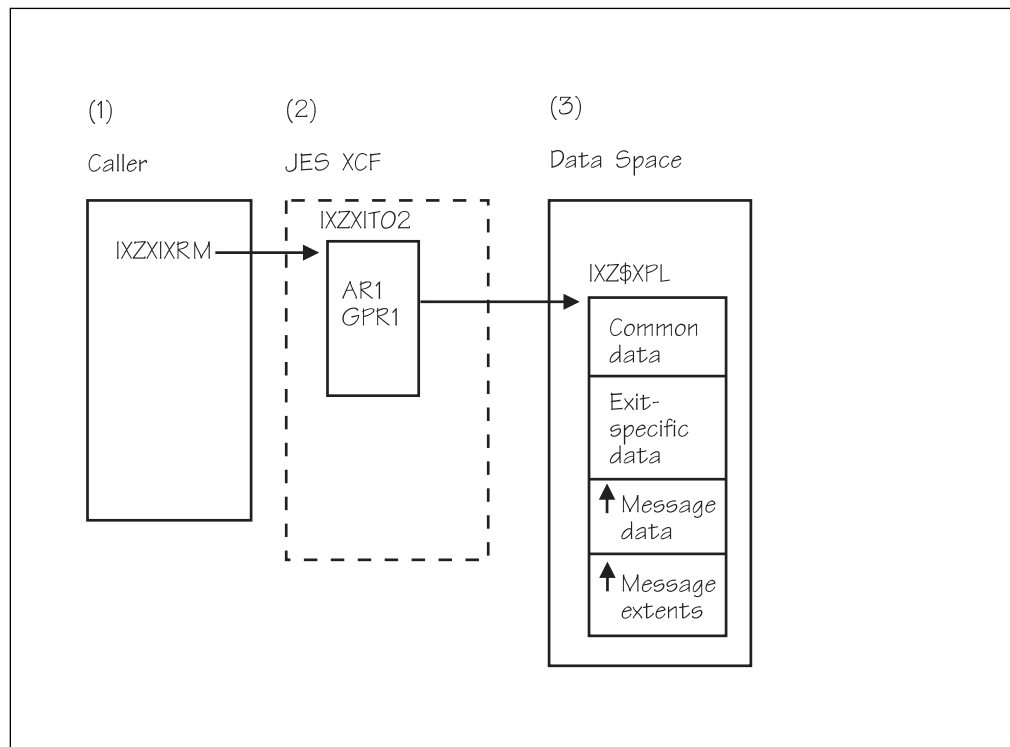


Figure 23. Access Path From Exit Caller to Message Data

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 ASYNCAK Messages

The message acknowledgement destination can be changed for ASYNCAK messages only. The sender codes the REQMBX= 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, refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

The parameter list consists of four sections:

- 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 if 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 linked 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	<ul style="list-style-type: none"> • 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 25 on page 69. 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 24 on page 67 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.

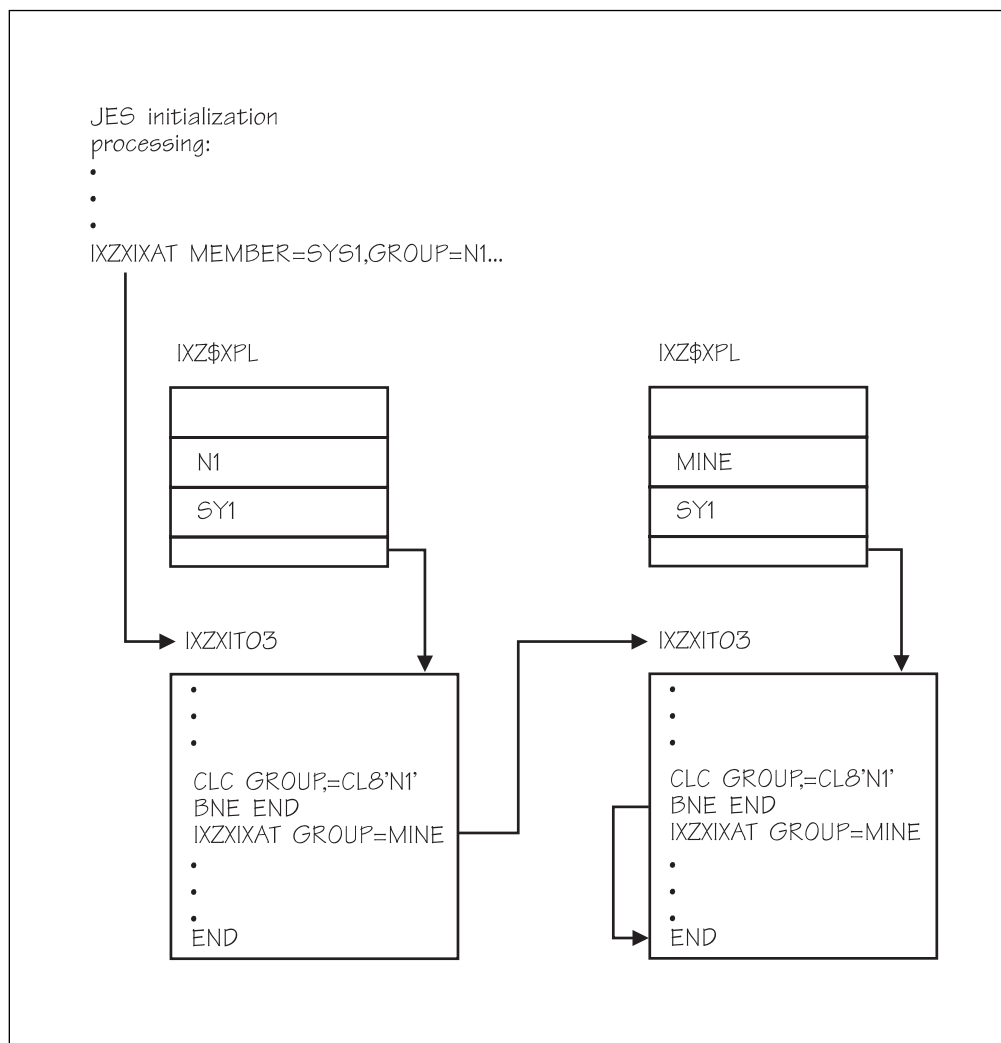


Figure 24. Example Attach (IXZXIT03) Code

Detach Processing

JES2 and JES3 detach processing are quite different as shown in section B of Figure 25 on page 69. 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 25 on page 69) 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:

IXZXIT03

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 24 on page 67 if you detach an installation-defined group.

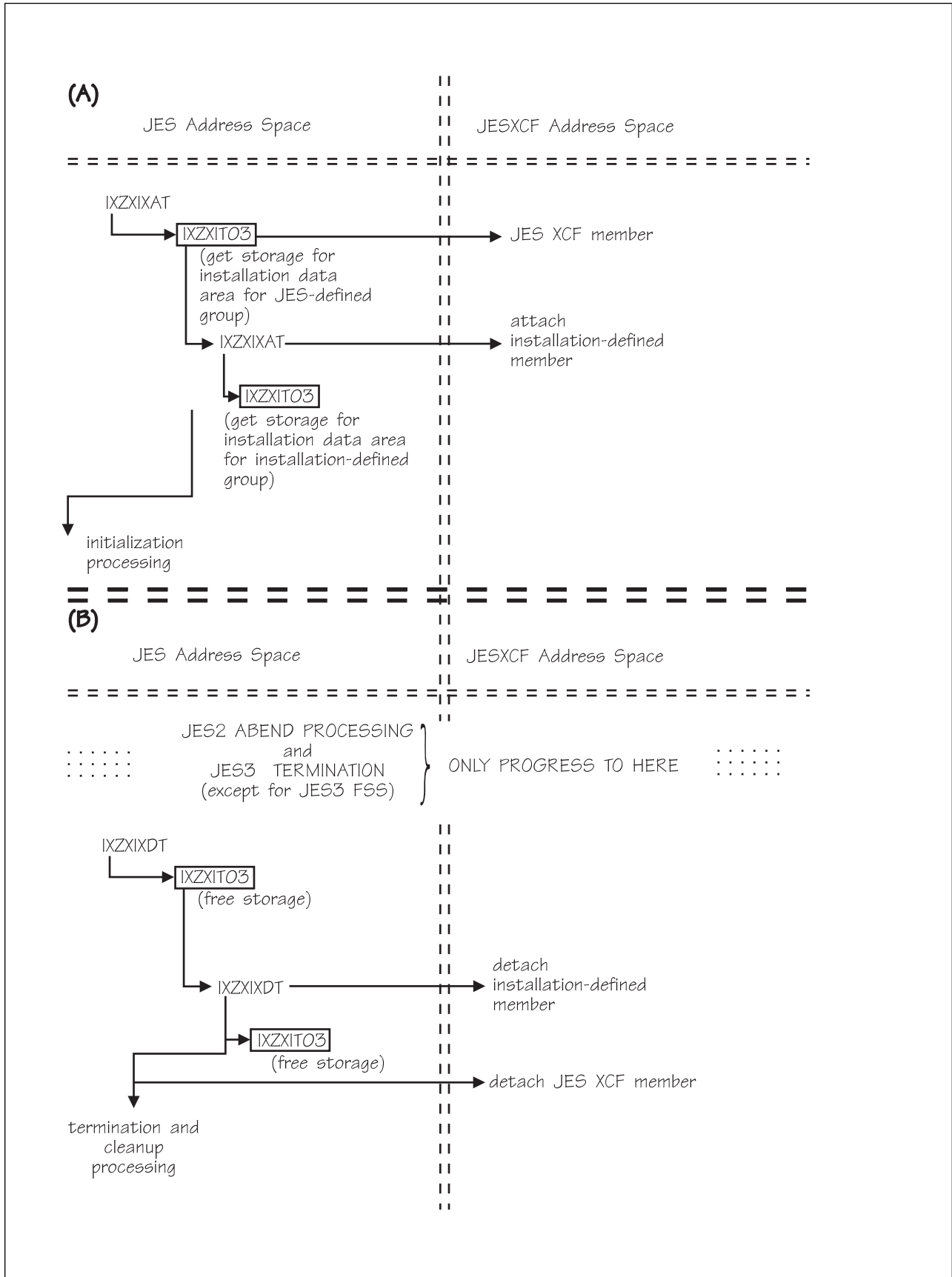


Figure 25. Exit IXZXIT03 Attach/Detach Processing

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 ENF-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, refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

The parameter list consists of two sections:

- 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

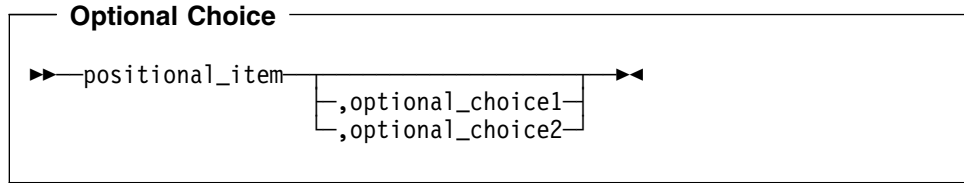
0-15 The routine must restore the contents to what they were when the routine received control

Return Code Meanings

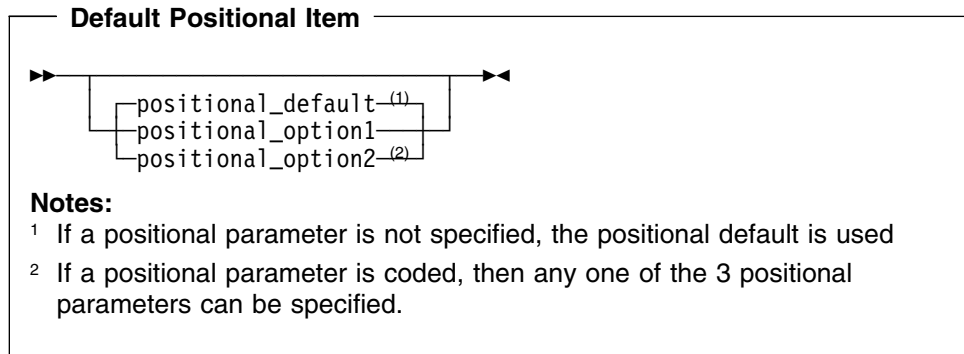
If processing is successful, IXZXIT03 returns control to its caller without a return code.

Coded Example of Exit Routine

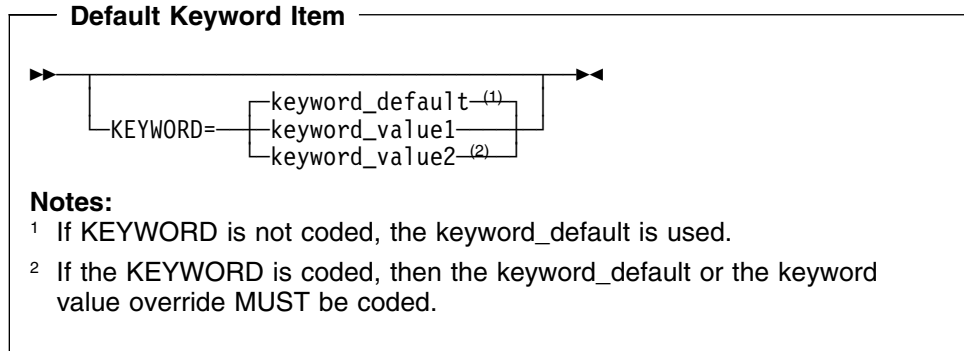
A sample exit IXZXIT03 is provided in SYS1.SAMPLIB member IXZEX03A. The sample code illustrates coding techniques only; it **does not** perform actual function so that you can use it as written.



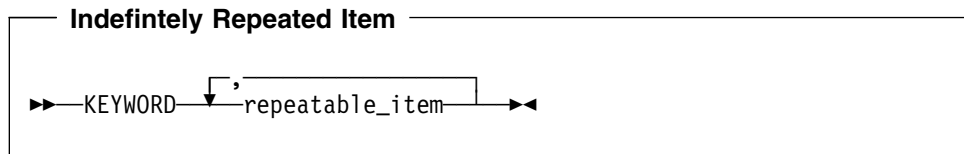
- If one of the items has a default, it appears above the main line (for that item) and the overriding choices will be shown on and/or below the main line. For POSITIONAL items the syntax is as follows:



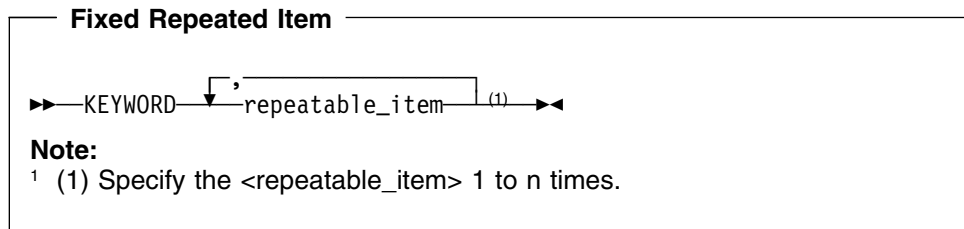
For KEYWORD items the syntax is as follows:



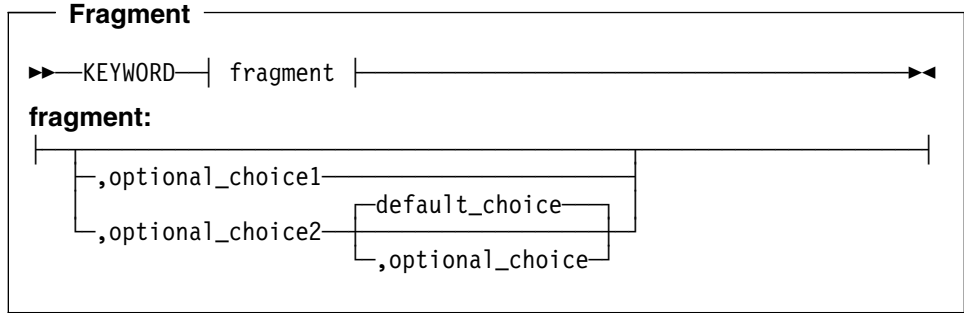
- An arrow returning to the left above the main line indicates an item that can be repeated indefinitely.



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



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



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 30 for a description of using IXZXIXAC to acknowledge messages.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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 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:

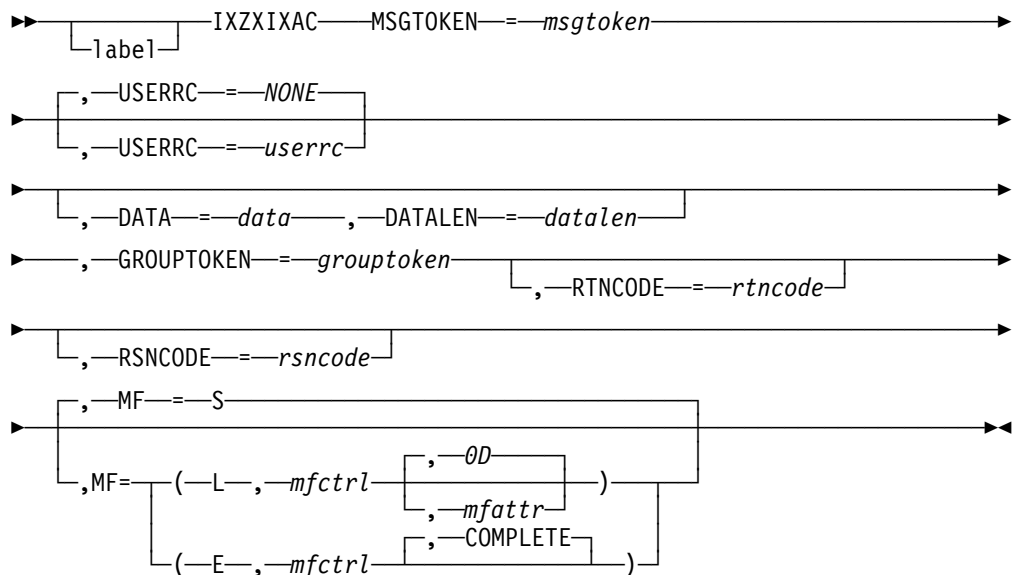
Register	Content
0,1	Used as a work register by the system
2-13	Unchanged
14,15	Used as a work register by the system

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 17 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	<p>Meaning: Processing was successful. The message was acknowledged.</p> <p>Action: None</p>
4	4	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXAC service contained an eyecatcher that was not valid.</p> <p>Action: Check the parameter list to be certain it was not overlaid with other data.</p>

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	8	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXAC service had a version indicator that was not valid.</p> <p>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.</p>
4	C	<p>Meaning: Processing failed because the group token passed to the IXZXIXAC service was not valid.</p> <p>Action: Verify that the correct group token is being passed to IXZXIXAC.</p>
4	10	<p>Meaning: Processing failed because the IXZXIXAC macro was issued on a member that is detaching.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
4	14	<p>Meaning: Processing failed because the message token passed to the IXZXIXAC service is not valid.</p> <p>Action: Verify that the correct message token is being passed to IXZXIXAC.</p>
4	18	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXAC service did not contain a DATALEN= specification.</p> <p>Action: Be certain that DATALEN= is specified when the DATA= parameter is used.</p>
4	1C	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXAC service contains a DATALEN= value that is not valid.</p> <p>Action: Be certain that the DATALEN= value was not negative or too large. DATALEN= must be a positive decimal integer up to 60 kilobytes.</p>
8	0	<p>Meaning: Exit IXZXIT01 failed. The acknowledgement being processed at the time IXZXIT01 failed is lost.</p> <p>Action: Review your IXZXIT01 code for coding errors, correct the error, and rerun the program that issues IXZXIXAC.</p>
C	0	<p>Meaning: Processing failed because the IXZXIXAC service ended abnormally.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

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 16.

Refer to “Creating an Installation-Defined Attachment to JES XCF (IXZXIXAT Macro)” on page 17 for a description of using IXZXIXAT to attach to a JES XCF group.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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.

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 initialization.

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	<p>Meaning: Processing was successful. The subtask has been attached and the JES XCF group has been activated.</p> <p>Action: None</p>
8	0	<p>Meaning: Processing failed because the IXZXIXAT service ended abnormally.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

IXZXIXCL - System Cleanup Initiated Indicator

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

Refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for a description of the IXCYGEPL mapping macro.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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 has established an attachment to the JES XCF group; that is, after IXZXIT03 processing has 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:

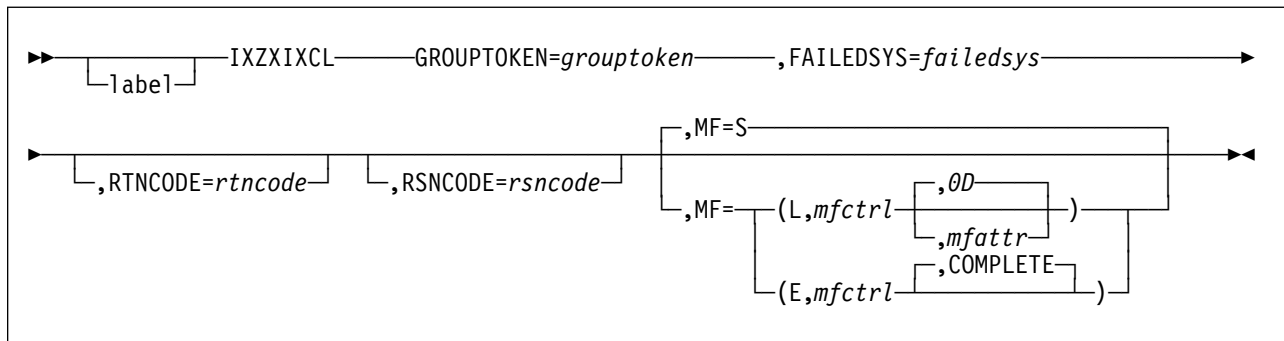
Register	Content
0,1	Used as a work register by the system
2-13	Unchanged
14,15	Used as a work register by the system

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; 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 IXZXIXCL 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 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 <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.

End of Product-sensitive programming interface

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 held by that attachment.

Refer to “Detaching from a JES XCF Group (IXZXIXDT Macro)” on page 36 for a description of using IXZXIXDT to detach from a JES XCF group.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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 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:

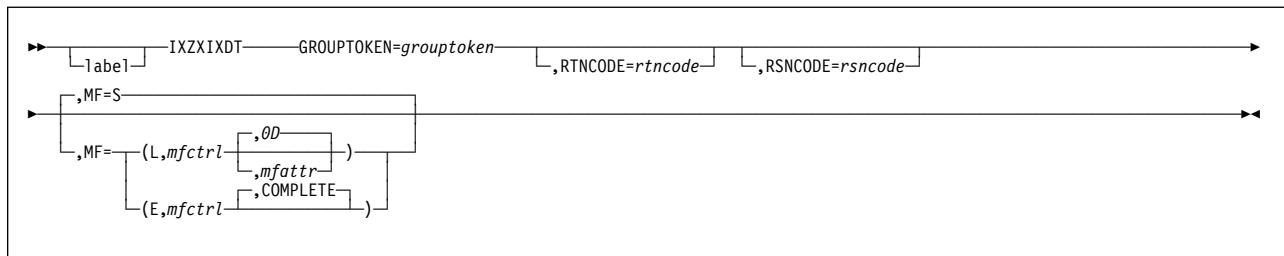
Register	Content
0,1	Used as a work register by the system
2-13	Unchanged
14,15	Used as a work register by the system

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

Syntax



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=group token

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

RTN CODE=rtn code

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 OD})

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 OD

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 OD, which forces the parameter list to a doubleword boundary.

Default: OD

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'8', it indicates that processing failed because the IXZXIXDT service ended abnormally, and you will also receive an associated ABEND code. 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	<p>Meaning: Processing was successful. JES has been detached from JES XCF and is no longer an active member of the corresponding XCF group.</p> <p>Action: None</p>
8	0	<p>Meaning: Processing failed because the IXZXIXDT service ended abnormally.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

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 15 on page 30 provides the JES XCF message mapping structure that shows the format in which JES XCF returns the requested information.

Refer to “Obtaining Information About JES Members in the XCF Group (IXZXIXIF Macro)” on page 31 for a description of using IXZXIXIF to obtain JES XCF information.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • JES2 or JES3 main task • JES2 or JES3 subtask • JES3 FSS
Minimum authorization:	<ul style="list-style-type: none"> • If you specify GROUPTOKEN= <ul style="list-style-type: none"> – Supervisor state with PSW key 0 or 1 • If you specify GROUPNAME= <ul style="list-style-type: none"> – 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, Vol 3 (IVT-RCWK)*, and Figure 16 on page 33 provides a graphic view of its structure.

Restrictions

Do not issue IXZXIXIF 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 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:

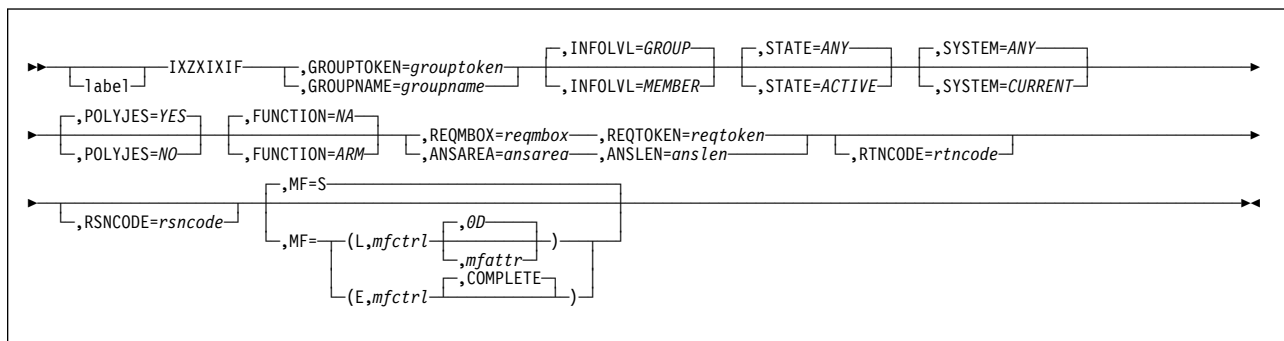
Register	Content
0,1	Used as a work register by the system
2-13	Unchanged
14,15	Used as a work register by the system

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 REQMBX causes an asynchronous call to JES XCF, whereas using ANSAREA= and ANSLEN= causes a synchronous call. Therefore, IBM recommends using REQMBX= 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=group token

Specifies the name (RS-type) or address (in GPR2-GPR12) of an optional fullword input field that contains the group token. Refer to "Retrieving the JES

XCF Group Token” on page 17 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 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, 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 IXZXIXIF 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 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	<p>Meaning: Processing was successful. The requested group or member information was returned.</p> <p>Action: none</p>
0	4	<p>Meaning: Processing failed because there are no members that match the filtering parameters (FUNCTION=, POLYJES=, STATE=, and SYSTEM=) specified on this macro request.</p> <p>Action: Verify that the filtering parameter specifications are set correctly. If appropriate, revise their settings and rerun the program that issues the IXZXIXIF macro.</p>

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

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 be notified of system events. JES XCF stores messages in the mailbox until the JDU receives the messages.

Refer to “Building a Mailbox (IXZXIXMB Macro)” on page 18 for a description of using IXZXIXMB to build a mailbox.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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, Vol 3 (IVT-RCWK)*.

Restrictions

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

Do not begin mailbox names with the character string *SYSJES*. This string is reserved for IBM.

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:

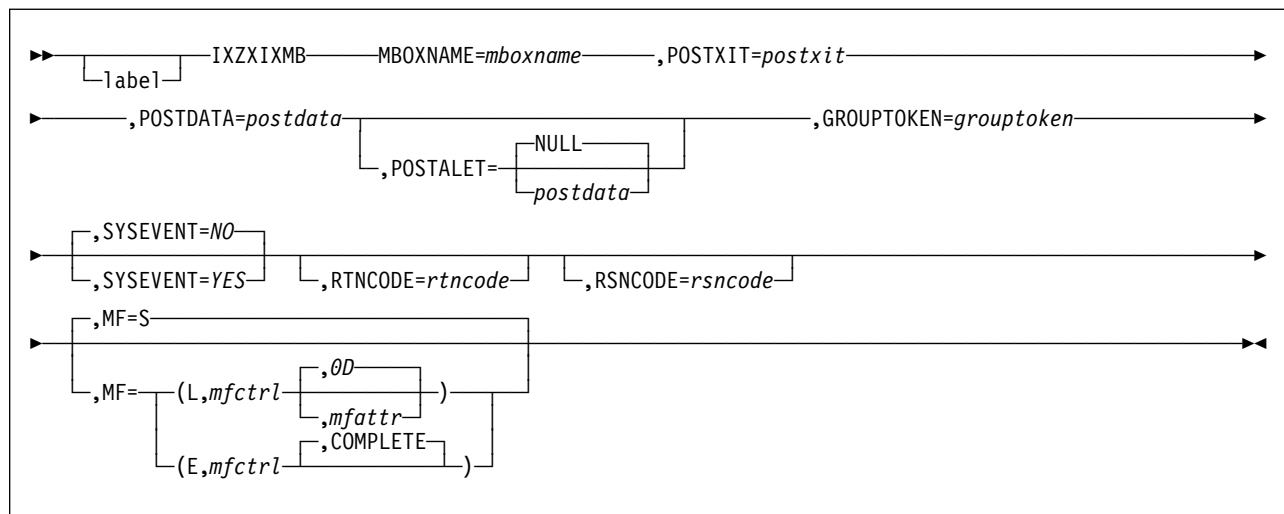
Register	Content
0,1	Used as work registers by the system
2-13	Unchanged
14,15	Used as work registers by the system

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*.

IBM recommends the use of 2-part mailbox names. The first part of the name might identify your company or department. The second part might identify the function 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:

```
DASDCORPbACCOUNT
```

```
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 137 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. Refer to "Retrieving the JES XCF Group Token" on page 17 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; 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, 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 IXZXIXMB service ended abnormally, and you will also receive an associated ABEND code. Refer to *z/OS MVS System Codes* 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.

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	C	<p>Meaning: Processing failed because the POSTXIT= parameter was not specified.</p> <p>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.</p>
4	10	<p>Meaning: Processing failed because the POSTDATA= parameter was not specified.</p> <p>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.</p>
4	14	<p>Meaning: Processing failed because the group token passed to the IXZXIXMB service was not valid.</p> <p>Action: Verify that the correct group token is being passed to IXZXIXMB.</p>
4	18	<p>Meaning: Processing failed because the IXZXIXMB service was issued on a member that is detaching.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
4	1C	<p>Meaning: Processing failed because the mailbox name specified contains characters that are not valid.</p> <p>Action: Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, \$, #, @, and embedded and trailing blanks are valid.</p>
4	20	<p>Meaning: Processing failed because the mailbox specified is already defined.</p> <p>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.</p>
C	0	<p>Meaning: Processing failed because the IXZXIXMB service abended.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

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 acknowledgement 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 35 for a description of using IXZXIXMC to clear a mailbox.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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:

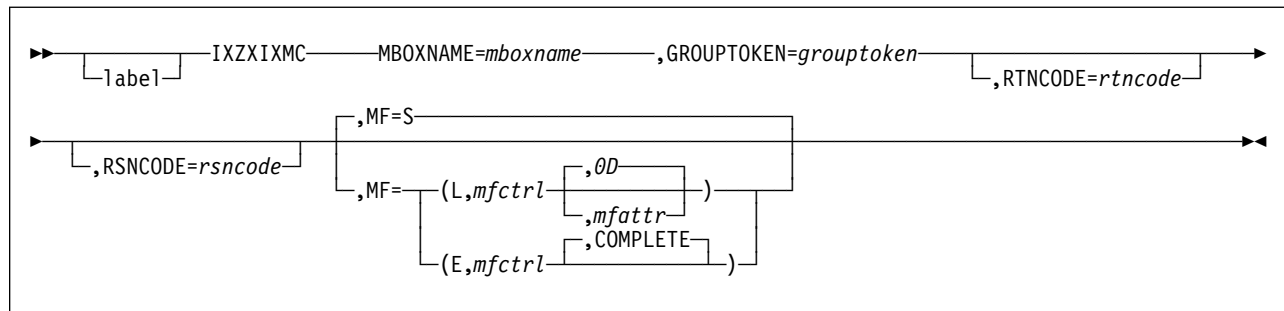
Register	Content
0,1	Used as work registers by the system
2-13	Unchanged
14,15	Used as work registers by the system

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

Syntax



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 17 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 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 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	<p>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.</p> <p>Action: None</p>
4	4	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXMC service contains an eyecatcher that is not valid.</p> <p>Action: Check the parameter list to be certain it was not overlaid with other data.</p>
4	8	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXMC service contains a version indicator that is not valid.</p> <p>Action: The caller might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.</p>

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	C	<p>Meaning: Processing failed because the group token passed to the IXZXIXMC service was not valid.</p> <p>Action: Verify that the correct group token is being passed to IXZXIXMC.</p>
4	10	<p>Meaning: Processing failed because the mailbox name passed to the IXZXIXMC service routine does not exist.</p> <p>Action: Verify that the correct mailbox name is being passed to IXZXIXMC.</p>
4	14	<p>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).</p> <p>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.</p>
4	18	<p>Meaning: Processing failed because the IXZXIXMC service was issued on a member that is detaching.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
4	1C	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXMC service routine specified a mailbox name that is not valid.</p> <p>Action: Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, \$, #, @, and embedded and trailing blanks are valid.</p>
C	0	<p>Meaning: Processing failed because the IXZXIXMC macro service abended.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

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 acknowledgement is returned to the JES members that sent the messages.

Refer to “Deleting a Mailbox (IXZXIXMD Macro)” on page 35 for a description of using IXZXIXMD to delete a mailbox.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none">• 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:

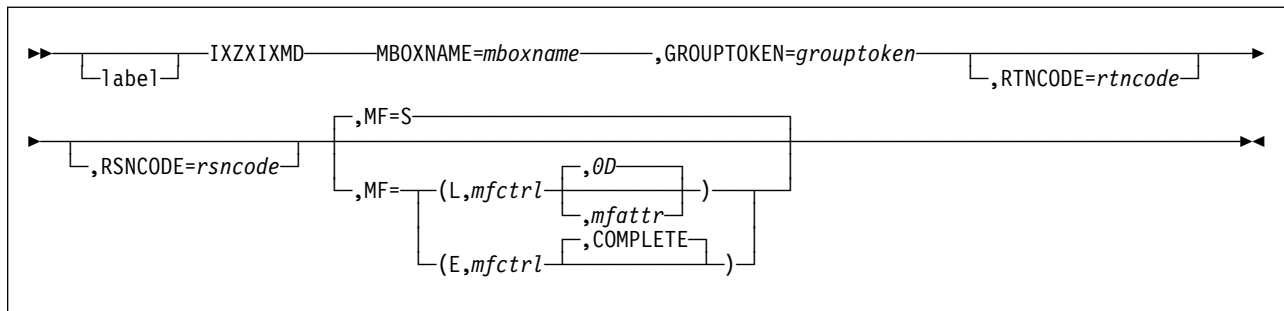
Register	Content
0,1	Used as work registers by the system
2-13	Unchanged
14,15	Used as work registers by the system

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

Syntax



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=mbxname

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 17 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 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 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	<p>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.</p> <p>Action: None</p>
4	4	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXMD service contains an eyecatcher that is not valid.</p> <p>Action: Check the parameter list to be certain it was not overlaid with other data.</p>
4	8	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXMD service contains a version indicator that is not valid.</p> <p>Action: The caller of IXZXIXAC may need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.</p>

IXZXIXMD Macro

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	C	<p>Meaning: Processing failed because the group token passed to the IXZXIXMD macro service is not valid.</p> <p>Action: Verify that the correct group token is being passed to IXZXIXMD.</p>
4	10	<p>Meaning: Processing failed because the mailbox name specified does not exist.</p> <p>Action: Verify that the correct mailbox name is being passed to IXZXIXMD.</p>
4	14	<p>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.</p> <p>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.</p>
4	18	<p>Meaning: Processing failed because the IXZXIXMD service was issued on a member that is detaching.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
4	1C	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXMD service contained a mailbox name that was not valid.</p> <p>Action: Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, \$, #, @, and embedded and trailing blanks are valid.</p>
C	0	<p>Meaning: Processing failed because the IXZXIXMD service abended.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

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 33 for a description of using IXZXIXPI to access JES XCF message queues.

Environment

The requirements for the caller are:

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

IXZXIXPI Macro

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

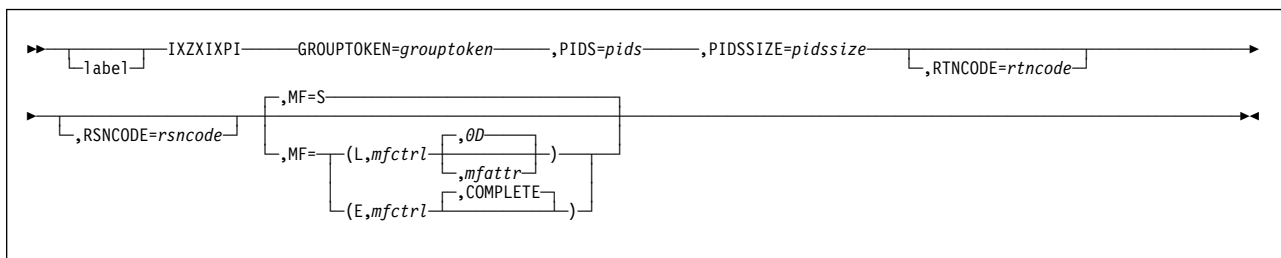
Register	Content
0,1	Used as a work register by the system
2-13	Unchanged
14,15	Used as a work register by the system

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=group token

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=pid size

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	<p>Meaning: Processing was successful. JES XCF accessed and returned the JES3 performance data.</p> <p>Action: None</p>
4	0	<p>Meaning: Processing failed because the size of the storage requested (PIDSSIZE=) was too small.</p> <p>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.</p>
8	4	<p>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.</p> <p>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.</p>
C	0	<p>Meaning: Processing failed because the IXZXIXPI service abended.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

_____ End of Product-sensitive programming interface _____

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.

Refer to “Receiving a Message (IXZXIXRM Macro)” on page 28 for a description of using IXZXIXRM to receive messages.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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 has defined the format and content of the message envelope, system event messages, and acknowledgement 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 returned by IXZXIXRM. For a description of the following mapping macros, refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

Mapping Macro Maps

IXCYGEPL	XCF data
IXZYIXAC	Acknowledgement messages
IXZYIXEN	The message envelope
IXZYIXIF	JES XCF data
IXZYIXJE	JES data
IXZYIXSE	System event messages

Also, refer to Figure 15 on page 30 for a graphic view of the message envelope and associated mapping macro structure, and *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for the IXZYIXEN macro mapping.

Restrictions

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

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:

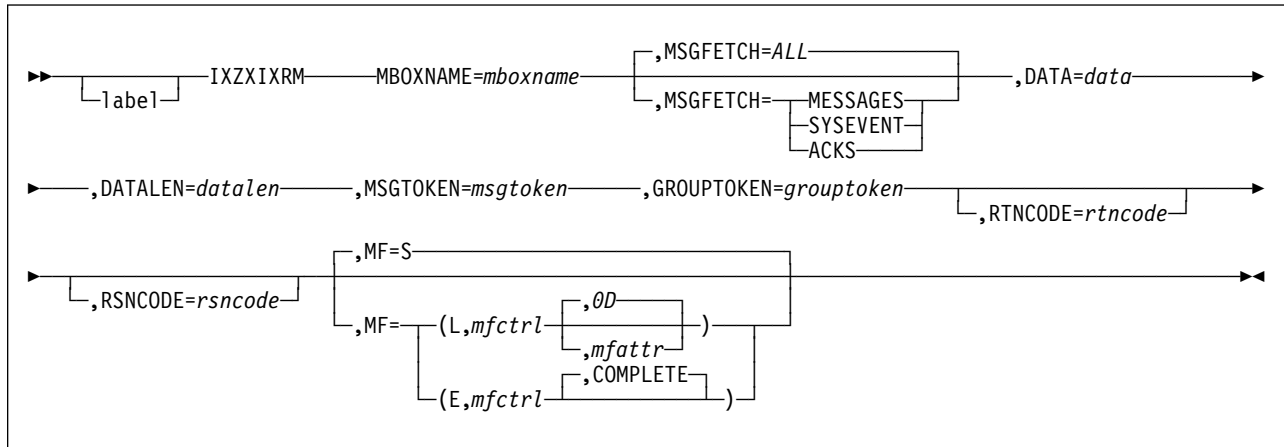
Register	Content
0,1	Used as work registers by the system
2-13	Unchanged
14,15	Used as work registers by the system

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

Syntax



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 will be 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. Refer to “Retrieving the JES XCF Group Token” on page 17 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 in-line parameter list and invoke the process acknowledgement service. Processing also includes checking for all required keywords and supplying defaults, if any, for omitted optional parameters.

MF=(L,mfctrl{,mfattr OD})

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 OD

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 IXZXIXRM 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 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	<p>Meaning: Processing was successful. The address of an envelope containing a message was returned.</p> <p>Action: None</p>
4	4	<p>Meaning: Processing failed because the parameter list passed to the IXZXIXRM service contained an eyecatcher that was not valid.</p> <p>Action: Check the parameter list to be certain it was not overlaid with other data.</p>

IXZXIRM Macro

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	8	<p>Meaning: Processing failed because the parameter list passed to the IXZXIRM service contained a version indicator that was not valid.</p> <p>Action: The caller might need to be recompiled. Check the parameter list to be certain it was not overlaid with other data.</p>
4	C	<p>Meaning: Processing failed because the group token passed to the IXZXIRM service was not valid.</p> <p>Action: Verify that the correct group token is being passed to IXZXIAC.</p>
4	10	<p>Meaning: Processing failed because the parameter list passed to the IXZXIRM service contained a mailbox name that was not valid.</p> <p>Action: Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, \$, #, @, and embedded and trailing blanks are valid.</p>
4	14	<p>Meaning: Processing failed because there are no messages in the mailbox specified on the IXZXIRM macro.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
4	18	<p>Meaning: Processing failed because the ASCB or TCB of the caller invoking the IXZXIRM service do not match those of the specified mailbox (those of the caller of the IXZXIMB service).</p> <p>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 IXZXIMB was issued. Be certain that both the ASCB and TCB of the caller match those of the mailbox builder, and rerun the program.</p>
4	1C	<p>Meaning: Processing failed because the mailbox specified on the IXZXIRM macro does not exist.</p> <p>Action: Verify that the correct mailbox name is being passed to IXZXIRM.</p>
4	20	<p>Meaning: Processing failed because the IXZXIRM service was called on a member that is detaching.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
8	0	<p>Meaning: IXZXIT02 failed. The message being processed at the time IXZXIT02 failed is lost.</p> <p>Action: Review your IXZXIT01 code for coding errors, correct the error and rerun the program that issues the IXZXIRM macro.</p>
C	0	<p>Meaning: Processing failed because the IXZXIRM service abended.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

IXZXISM - Send a Message

Use the IXZXISM 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 (IXZXISM Macro)” on page 21 for a description of using IXZXISM to send a message.

Environment

The requirements for the caller are:

JES environments:	<ul style="list-style-type: none"> • 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 IXZXISM is invoked.

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

Restrictions

Do not issue IXZXISM 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 IXZXISM 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 IXZXISM 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 IXZXISM returns control, the access registers (ARs) contain:

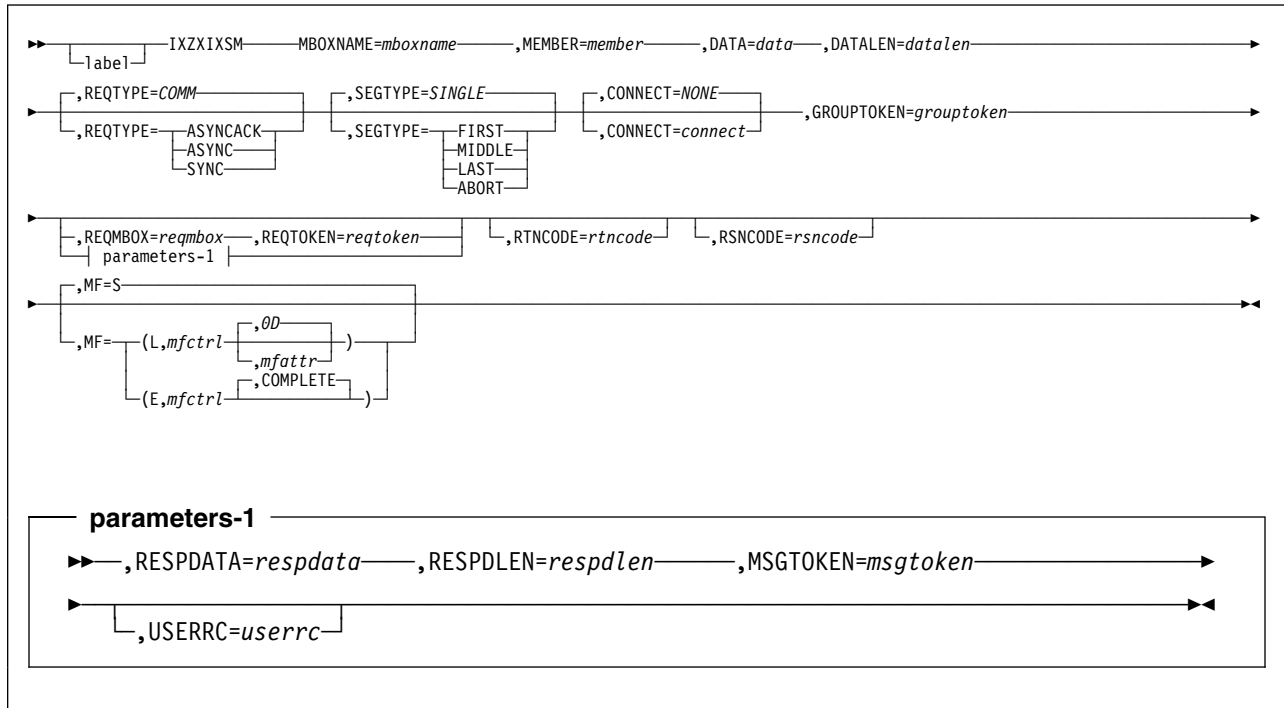
Register	Content
0,1	Used as work registers by the system
2-13	Unchanged
14,15	Used as work registers by the system

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 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 ASYNCAK 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.

ASYNCAK

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 REQMBX= for REQTYPE=ASYNCAK 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

REQMBX=reqmbx

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=ASYNCAK, you must also specify REQMBX.

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=ASYNCAK, 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 17 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 IXZXISM 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 IXZXISM 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 IXZXISM 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 IXZXISM 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 IXZXISM service was not valid. Action: Verify that the correct group token is being passed to IXZXISM.
4	10	Meaning: Processing failed because the member name passed to the IXZXISM service contains characters that are not valid. Action: Be certain you specified MEMBER= correctly. Only A-Z, 0-9, \$, #, and @ are valid.

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	14	<p>Meaning: Processing failed because the mailbox name passed to the IXZXIXSM service contains characters that are not valid.</p> <p>Action: Be certain you specified MBOXNAME= correctly. Only A-Z, 0-9, \$, #, @ and embedded and trailing blanks are valid.</p>
4	18	<p>Meaning: Processing failed because the mailbox name specified on the REQMBOX= parameter contains characters that are not valid.</p> <p>Action: Be certain you specified REQMBOX= correctly. Only A-Z, 0-9, \$, #, @, and embedded and trailing blanks are valid.</p>
4	20	<p>Meaning: Processing failed because the IXZXIXSM service was called on a member that is detaching.</p> <p>Action: Check to be certain this is not a symptom of a serialization problem within your JES XCF group.</p>
4	24	<p>Meaning: Processing failed because CONNECT= was not specified.</p> <p>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.</p>
4	28	<p>Meaning: Processing failed because MSGTOKEN= was not specified. service.</p> <p>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.</p>
4	2C	<p>Meaning: Processing failed because you did not also specify REQMBOX= on this IXZXIXSM call. REQMBOX= is required for ASYNCAK call to identify the mailbox into which the acknowledgement message is returned.</p> <p>Action: Add REQMBOX= to the IXZXIXSM macro and rerun the program that issues the send message call.</p>
4	34	<p>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.</p> <p>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.</p>
4	38	<p>Meaning: Processing failed because the DATALEN= value is not valid.</p> <p>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.</p>

Return Code (hex)	Reason Code (hex)	Meaning and Action
4	3C	<p>Meaning: Processing failed because the message was cleared (by a IXZXIMC call) from the target mailbox before the message was processed.</p> <p>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.</p>
4	48	<p>Meaning: Processing failed because the message was deleted (by a IXZXIMD call) from the target mailbox before the message was processed.</p> <p>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.</p>
4	4C	<p>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 IXZXIMD service.</p> <p>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.</p>
8	0	<p>Meaning: IXZXIT01 failed. The message being processed at the time IXZXIT01 failed is lost.</p> <p>Action: Review your IXZXIT01 code for coding errors, correct the error, and rerun the program that issues the IXZXISM macro.</p>
C	0	<p>Meaning: Processing failed because the IXZXISM service abended.</p> <p>Action: Refer to <i>z/OS MVS System Codes</i> for a description of the abend code and its associated reason codes.</p>

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 20 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 linked into an authorized library in the LNKLIST 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, refer to *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

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.

POST Exit Routine

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

- AD/Cycle
- CBIPO
- CICS
- CICS/ESA
- DB2
- DFSMS/MVS
- ESCON

- Extended Services
- Hardware Configuration Definition
- Hiperbatch
- IBM
- IBMLink
- IMS/ESA
- MVS/ESA
- OpenEdition
- OS/2
- OS/390
- Print Services Facility
- PR/SM
- RACF
- Resource Measurement Facility
- RMF
- S/390
- SAA
- Sysplex Timer
- System/390
- VTAM
- z/OS

The following terms are trademarks of other companies as follows:

GUIDE Guide International Corporation

SHARE Share Incorporated

Other company, product, or service names may be trademarks or service marks of others.

Glossary

This glossary defines technical terms and abbreviations used in JES2 documentation. If you do not find the term you are looking for, refer to the index of the appropriate manual or view *IBM Dictionary of Computing*, located at:

<http://www.ibm.com/ibm/terminology>

This glossary includes terms and definitions from:

American National Standard Dictionary for Information Systems, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by an asterisk (*) that appears between the term and the beginning of the definition; a single definition taken from ANSI is identified by an asterisk after the item number for that definition.

ALET. access list entry token

cross-system coupling facility (XCF). XCF provides the MVS coupling services that allow authorized programs on MVS systems in a multisystem environment to communicate (send and receive data) with programs on MVS systems.

data packet. The data (messages, system event messages, and acknowledgements) that one JDU running on one member sends to itself or another member within the JES XCF group. The maximum size of the data is 60 kilobytes.

envelope. Header data associated with a message that contains the delivery information used to deliver the associated message to the receiver.

FMID. Function modification identifier. The IBM release-specific product identifier such as HJE6610 for OS/390 Release 1 JES2.

FSS. functional subsystem. An address space uniquely identified as performing a specific function related to the JES. An example of an FSS is the program Print Services Facility that operates the 3800 Model 3 and 38xx printers.

JDU (JES dispatchable unit). A JES-independent term for the processor control element (PCE in JES2)

and the dynamic support program or function control table (DSP and FCT, respectively, in JES3). The JES subdispatcher running under the subsystem's main task gives control to JDUs.

JES common coupling services. A set of macro-driven services that provide the communication interface between JES members of a sysplex. Synonymous with *JES XCF*.

JES XCF. JES cross-system coupling services. The MVS component, common to both JES2 and JES3, that provides the cross-system coupling services to either JES2 multi-access spool members or JES3 complex members, respectively.

JES2. job entry subsystem 2. An MVS subsystem that receives jobs into the system and processes all output data produced by the job.

JES3. job entry subsystem 3. An MVS subsystem that receives jobs into the system and processes all output data produced by the job.

mailbox. A logical queue of ordered messages that is maintained by JES XCF in a location associated with a JES dispatchable unit (JDU) where messages are held until the receiving JDU is able to receive them.

message. In the JES XCF context, any data that comprises a data packet. to include system messages (single- or multi-segment), system event messages, and acknowledgements and the data associated with them.

messaging. The ability for one JDU running on one system to send a data packet to itself or another system within the JES XCF group.

multi-segment message. A message containing more than one part. Each part is associated to the others to form the entire message so that it can be sent and received as if it were a single message.

POST exit routine. A routine that receives control when JES XCF places a message in a mailbox. The routine informs (posts) the associated JDU that mail is waiting to be received.

sysplex. A sysplex (systems complex) is the set of one or more MVS systems that is given an XCF sysplex name and in which programs in the systems can then use XCF services.

XCF. see *cross-system coupling facility*.

XCF group. A set of related members defined to XCF. The members can communicate and share data through the cross-system coupling facility (XCF) services.

Index

A

abend processing

for mailboxes 36

abort

multi-segment message 26

acknowledgement destination

change using IXZXIT01 53

change using IXZXIT02 61

acknowledgement of message

asynchronous 24

receipt 30

requirement 30

synchronous 25

ALET (access list entry token)

definition 145

append message extents

using IXZXIT01 53

asynchronous message

with acknowledgement 24

without acknowledgement 24, 25

attach/detach exit

IXZXIT03 65

B

build a mailbox

using IXZXIXMB 101

C

change message

content

using IXZXIT02 61

destination

using IXZXIT01 52

clear a mailbox 35

using IXZXIXMC macro 107

component trace

for JES XCF processing 37

SYSJES COMP id 37

control macro expansion printing

using ZPRINT 75

cross-system coupling facility (XCF)

definition 145

D

data packet

definition 145

delete a mailbox

using IXZXIXMD macro 112

detach JES

from a JES XCF group 36

diagnostic support

for JES XCF 37

discontinue sending

a multi-segment message 26

E

entry specifications

for IXZXIT01 56

for IXZXIT02 62

for IXZXIT03 71

for POST exit routine 138

envelope

definition 145

for messages 29

environment

IXZXIT01 51

IXZXIT02 59

IXZXIT03 65

POST exit routine 137

exit

for JES XCF 43

IXZXIT01 51

IXZXIT02 59

IXZXIT03 65

POST exit routine 137

F

filter data

returned by IXZXIXIF macro 31

FLOW trace record 38

FSS (functional subsystem)

definition 145

FUNCTION

IXZXIXIF parameter 98

G

GROUPNAME

IXZXIXIF parameter 96

I

information about JES XCF group members

obtain using IXZXIXIF 31

initialization processing

for JES 11

install

IXZXIT01 51

IXZXIT02 59

install *(continued)*

IXZXIT03 65
POST exit routine 137

installation exit

identifying the POST exit routine 20
trace record 39

installation-defined attachment to common coupling services

creating 17

IXZ\$XPL mapping macro

for IXZXIT01 processing 52
for IXZXIT02 processing 60

IXZXIT01 installation exit

appending message extents 53
changing
 message content 53
 message destination 52
entry specifications 56
environment 51
installing 51
parameter list description 56
processing 51
programming considerations 56
recovery 51
registers
 entry 56
 exit 57
rerouting an acknowledgement 53
return codes 57

IXZXIT02 installation exit

changing message content 61
entry specifications 62
environment 59
installing 59
parameter list description 62
processing 59
programming considerations 62
recovery 59
registers
 entry 62
 exit 63
rerouting an acknowledgement 61
retrieving message extents 61
return codes 63

IXZXIT03 installation exit

attach/detach exit 65
entry specifications 71
environment 65
exit processing 66
installing 65
parameter list description 71
programming considerations 71
recovery 65
registers
 entry 71
 exit 72

IXZXIT03 installation exit *(continued)*

return codes 72

IXZXIXAC macro 80

acknowledge message 76

IXZXIXAT macro 85

attach to JES XCF group 81

IXZXIXCL macro 89

system cleanup initiated indicator 86

IXZXIXDT macro 93

detach from a JES XCF group 36, 90

IXZXIXIF

to obtain information 31

IXZXIXIF macro 100

filtering data 31
request member information 94

IXZXIXIF parameter

FUNCTION 98
GROUPNAME 96
POLYJES 97
STATE 97
SYSTEM 97

IXZXIXMB macro

build a mailbox 101, 106

IXZXIXMD macro 116

delete a mailbox 112

IXZXIXPI macro 120

collect JES3 performance data 117
JES3 processing delays 33

IXZXIXRM macro 126

receive a message 121

IXZXIXSM macro 135

send a message 127

IXZYIXIF mapping macro 32

IXZYIXPE mapping macro 101

IXZYIXSE mapping macro 20

J

JDU (JES dispatchable unit)

definition 145

JES

group token retrieval 17
initialization processing 11
members
 obtaining information about 31
 message processing 13

JES abend processing

for mailboxes 36

JES common coupling services

definition 145
using 11

JES XCF

affecting message processing 5
definition 145
diagnostic support 37
exits
 implementing 43

JES XCF *(continued)*

- getting system cleanup status from JES 35
- introduction 1
- providing security 6
- trace support 37
- uses 5

JES XCF group token

- retrieve 17

JES XCF macro

- IXZXIXAC 76
- IXZXIXAT 81
- IXZXIXCL 86
- IXZXIXDT 90
- IXZXIXIF 94
- IXZXIXMB 101
- IXZXIXMC 107
- IXZXIXMD 112
- IXZXIXPI 117
- IXZXIXRM 121

JES XCF trace 37

JES-defined attachment

- determining if you should use the attachment 16
 - to XCF
 - description 16
 - IBM recommendation 16

JES2

- definition 145

JES2 MAS

- relationship to sysplex 1

JES3

- collect performance data 117
- definition 145
- delays in processing 33

JES3 complex

- relationship to sysplex 1

M

macro

- IXZXIXSM 127

macro syntax

- how to read 73

mailbox

- availability 19
- building 18
- clearing messages 35
- definition 145
- deleting 35
- POST exit routine 137

mailbox processing

- for JES abends 36

mapping macro

- IXCYGEPL 20
- IXZ\$XPL
 - use with IXZXIT01 processing 52
 - use with IXZXIT02 processing 60

mapping macro *(continued)*

- IXZYIXAC 29
- IXZYIXEN 29
- IXZYIXJE 20
- IXZYIXPE 101
- IXZYIXSE 29

message

- acknowledgement
 - reroute using IXZXIT01 53
 - reroute using IXZXIT02 61
- asynchronous with acknowledgement 24
- asynchronous without acknowledgement 24, 25
- content
 - changing using IXZXIT01 53
 - changing using IXZXIT02 61
- definition 145
- destination
 - change using IXZXIT01 52
- envelope 29
- extents
 - add using IXZXIT01 53
 - retrieve using IXZXIT02 61
- mapping macros 29
- receiving 28
- sending 21
- sending a multiple segment 26
- sending a single segment 26
- synchronous with acknowledgement 25
- trace record 38

message processing

- affect using JES XCF 5
- for JES 13

messaging

- definition 145

module footprint

- trace record 38

monitoring of XCF events

- requesting 20

MSGTRC trace record 38

multi-segment message

- abort 26
- definition 145
- discontinue 26
- send 26

N

Notices 141

P

parameter list description

- for IXZXIT01 56
- for IXZXIT02 62
- for IXZXIT03 71
- for POST exit routine 138

performance delays
due to JES3 processing 33

POLYJES

IXZXIXIF parameter 97

POST exit routine

coding requirements 20, 137
definition 145
entry specifications 138
environment 137
identifying 20
installing 137
parameter list description 138
processing 137
programming considerations 138
recovery 137
registers
entry 138
exit 138
return codes 139

printing macro expansions

control with ZPRINT 75

programming considerations

for IXZXIT01 56
for IXZXIT02 62
for IXZXIT03 71
for POST exit routine 138

R

RACF

profile for JES XCF 6
security for JES XCF 6
started procedures table 6

receipt of a message

acknowledging 30

receive exit

IXZXIT02 59

receiving a message 28

recovery

for IXZXIT01 51
for IXZXIT02 59
for IXZXIT03 65
for mailbox POST exit routine 137

retrieve JES XCF group token 17

retrieve message extents

using IXZXIT02 61

RMF

collecting performance data 33

S

send a message in JES environments 21

single-segment message

sending 26

STATE

IXZXIXIF parameter 97

synchronous message

with acknowledgement 25

SYSJES component trace id 37

sysplex

definition 145
relationship to JES XCF group 1
relationship to JES2 MAS 1
relationship to JES3 complex 1

SYSTEM

IXZXIXIF parameter 97

system cleanup

indicate status to JES XCF 35

system event message

IXZYIXSE mapping macro 20

T

trace record

FLOW (module footprint) 38
for JES XCF 37
installation exit (USRXIT) 39
module footprint (FLOW) 38
MSGTRC (message) 38
SYSJES COMP id 37
USRXIT (installation exit) 39
XCF and JES XCF event (XCFEVT) 38
XCFEVT (system event) 38

transport exit

IXZXIT01 51

U

user exit trace

see installation exit trace 39

uses for JES XCF 5

USRXIT

installation exit trace 39

X

XCF (cross-system coupling facility)

definition 146

XCF and JES XCF event

trace record 38

XCF event message

requesting 20

XCF group

definition 146

XCFEVT trace record 38

Z

ZPRINT

to control macro expansion printing 75

Communicating Your Comments to IBM

z/OS
MVS Programming:
JES Common Coupling Services
Publication No. SA22-7615-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
 - FAX: (International Access Code)+1+845+432-9405
- If you prefer to send comments electronically, use the following e-mail address:
 - mhvrdfs@us.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

z/OS

MVS Programming:

JES Common Coupling Services

Publication No. SA22-7615-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: _____

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

- | | | | |
|--------------------------|-------------------------------|--------------------------|------------------------|
| <input type="checkbox"/> | As an introduction | <input type="checkbox"/> | As a text (student) |
| <input type="checkbox"/> | As a reference manual | <input type="checkbox"/> | As a text (instructor) |
| <input type="checkbox"/> | For another purpose (explain) | | |

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number:

Comment:

Name

Address

Company or Organization

Phone No.



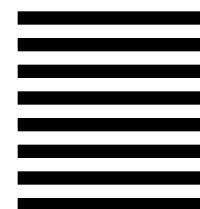
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400



Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5694-A01



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SA22-7615-00

