IMS

# IMS Connect Guide and Reference

*Version 9*

SC18-9287-00

IMS

# IMS Connect Guide and Reference

*Version 9*

SC18-9287-00

**Note:**

Before using this information and the product it supports, be sure to read the information in “Notices” on page 303.

**First Edition (October 2004) (Softcopy Only)**

This edition applies to Version 9 of IMS (product number 5655–J38) and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# Figures

# Tables

# About This Book

This information is available as part of the DB2 Information Management Software Information Center for z/OS Solutions. To view the information within the DB2 Information Management Software Information Center for z/OS Solutions, go to http://publib.boulder.ibm.com/infocenter/dzichelp. This information is also available in PDF and BookManager® formats. To get the most current versions of the PDF and BookManager formats, go to the IMS™ Library page at www.ibm.com/software/data/ims/library.html.

IMS Version 9 provides an integrated IMS Connect function, which offers a functional replacement for the IMS Connect tool (program number 5655-K52). In this information, the term IMS *Connect* refers to the integrated IMS Connect function that is part of IMS Version 9, unless otherwise indicated.

This book is designed to help programmers, operators, and system support personnel perform these tasks:

- Plan for and design the installation of IMS Connect.
- Install and operate IMS Connect.
- Diagnose and recover from IMS Connect system problems.
- Write an IMS Connect client.
- Use IMS Connect with IMS Connector for Java™.

## Prerequisite Knowledge

Before using this book, you may need to understand:

- Basic IMS concepts
- Basic Open Transaction Manager Access (OTMA) concepts
- The IMS environment
- TCP/IP configuration concepts
- Basic OS/390® MVS™ Extended Coupling Facility (XCF) concepts
- Basic RACF® (or equivalent product) concepts
- Basic Secure Sockets Layer (SSL) concepts

For a list of references to related publications, refer to the “Bibliography” on page 307.

## Summary of Contents

This book has three parts.

- Part 1, “User’s Guide and Reference,” on page 1 contains information on how to set up, install, and operate IMS Connect.
- Part 2, “Programmer’s Guide and Reference,” on page 97 provides guidelines for how to write an IMS Connect client and how to diagnose and recover from system problems.
- Part 3, “Messages and Codes,” on page 165 describes the IMS Connect error codes, abend codes, and their associated messages.

## IBM Product Names Used in This Information

In this information, the licensed programs shown in Table 1 are referred to by their short names.

*Table 1. Licensed Program Full Names and Short Names*

| Licensed program full name | Licensed program short name |
|---|---|
| IBM® Application Recovery Tool for IMS and DB2® | Application Recovery Tool |
| IBM CICS® Transaction Server for OS/390 | CICS |
| IBM CICS Transaction Server for z/OS® | CICS |
| IBM DB2 Universal Database™ | DB2 Universal Database |
| IBM DB2 Universal Database for z/OS | DB2 UDB for z/OS |
| IBM Enterprise COBOL for z/OS and OS/390 | Enterprise COBOL |
| IBM Enterprise PL/I for z/OS and OS/390 | Enterprise PL/I |
| IBM High Level Assembler for MVS & VM & VSE | High Level Assembler |
| IBM IMS Advanced ACB Generator | IMS Advanced ACB Generator |
| IBM IMS Batch Backout Manager | IMS Batch Backout Manager |
| IBM IMS Batch Terminal Simulator | IMS Batch Terminal Simulator |
| IBM IMS Buffer Pool Analyzer | IMS Buffer Pool Analyzer |
| IBM IMS Command Control Facility for z/OS | IMS Command Control Facility |
| IBM IMS Connect for z/OS | IMS Connect |
| IBM IMS Connector for Java | IMS Connector for Java |
| IBM IMS Database Control Suite | IMS Database Control Suite |
| IBM IMS Database Recovery Facility for z/OS | IMS Database Recovery Facility |
| IBM IMS Database Repair Facility | IMS Database Repair Facility |
| IBM IMS DataPropagator™ for z/OS | IMS DataPropagator |
| IBM IMS DEDB Fast Recovery | IMS DEDB Fast Recovery |
| IBM IMS Extended Terminal Option Support | IMS ETO Support |
| IBM IMS Fast Path Basic Tools | IMS Fast Path Basic Tools |
| IBM IMS Fast Path Online Tools | IMS Fast Path Online Tools |
| IBM IMS Hardware Data Compression-Extended | IMS Hardware Data Compression-Extended |
| IBM IMS High Availability Large Database (HALDB) Conversion Aid for z/OS | IBM IMS HALDB Conversion Aid |
| IBM IMS High Performance Change Accumulation Utility for z/OS | IMS High Performance Change Accumulation Utility |
| IBM IMS High Performance Load for z/OS | IMS HP Load |
| IBM IMS High Performance Pointer Checker for OS/390 | IMS HP Pointer Checker |
| IBM IMS High Performance Prefix Resolution for z/OS | IMS HP Prefix Resolution |
| IBM Tivoli® NetView® for z/OS | Tivoli NetView for z/OS |
| IBM WebSphere® Application Server for z/OS and OS/390 | WebSphere Application Server for z/OS |

*Table 1. Licensed Program Full Names and Short Names (continued)*

| **Licensed program full name** | **Licensed program short name** |
|---|---|
| IBM WebSphere MQ for z/OS | WebSphere MQ |
| IBM WebSphere Studio Application Developer Integration Edition | WebSphere Studio |
| IBM z/OS | z/OS |

## How to Read Syntax Diagrams

The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

```
►►──required_item──────────────────────────────────────────────────►◄
```

- Optional items appear below the main path.

```
►►──required_item──┬───────────────┬───────────────────────────────►◄
                   └─optional_item─┘
```

  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

```
                   ┌─optional_item─┐
►►──required_item──┴───────────────┴───────────────────────────────►◄
```

- If you can choose from two or more items, they appear vertically, in a stack.

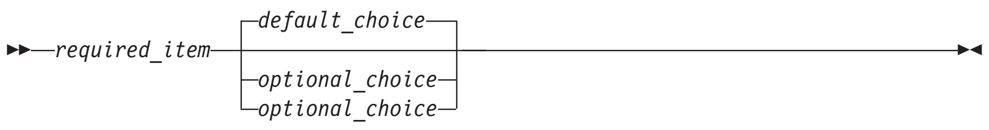
  If you *must* choose one of the items, one item of the stack appears on the main path.

```
►►──required_item──┬─required_choice1─┬────────────────────────────►◄
                   └─required_choice2─┘
```

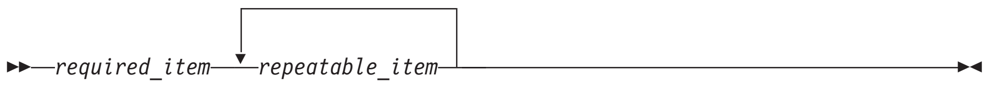
  If choosing one of the items is optional, the entire stack appears below the main path.

```
►►──required_item──┬──────────────────┬────────────────────────────►◄
                   ├─optional_choice1─┤
                   └─optional_choice2─┘
```

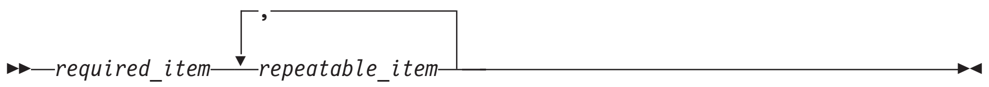
  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



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

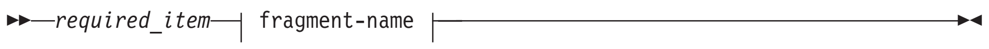


If the repeat arrow contains a comma, you must separate repeated items with a comma.

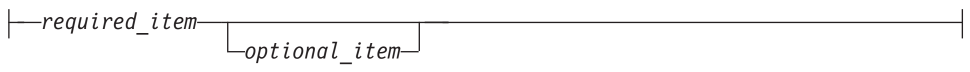


A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



**fragment-name:**



- In IMS, a b symbol indicates one blank position.
- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

## How to Send Your Comments

Your feedback is important in helping us provide the most accurate and highest quality information. If you have any comments about this or any other IMS information, you can take one of the following actions:

- Go to the IMS Library page at www.ibm.com/software/data/ims/library.html and click the Library Feedback link, where you can enter and submit comments.
- Send your comments by e-mail to imspubs@us.ibm.com. Be sure to include the title, the part number of the title, the version of IMS, and, if applicable, the

specific location of the text on which you are commenting (for example, a page number in the PDF or a heading in the Information Center).

# Summary of Changes

This section summarizes the significant improvements or enhancements for IMS Connect and refers you to relevant sections of this book for more information.

## Changes to This Book for IMS Version 9

For IMS Connect, the improvements and enhancements include:
- Commit mode 0 persistent socket support
- Purge not deliverable support
- New and enhanced IMS Connect commands
- Cancel timer support
- New RESUME_TPIPE single with wait option
- IMS Connect event recording support
- The MAXSOC= parameter has changed and requires APAR PQ90051

The following chapters in this book have been modified to reflect new or changed product features:
- Chapter 4, "IMS Connect Definition and Tailoring," on page 11
- Chapter 15, "User Message Exits for IMS Connect," on page 161
- Chapter 10, "Protocols," on page 99
- Chapter 11, "Security Support," on page 127
- Chapter 13, "IMS Connect Commands," on page 137
- Chapter 16, "IMS Connect Error Codes and Messages," on page 167
- Chapter 17, "IMS Connect Return and Reason Codes," on page 221

The following chapters in this book are new:
- Chapter 14, "IMS Connect z/OS Commands," on page 151
- Appendix G, "HWSTECL0 User Exit," on page 267.

IMS Connect Version 9 is the final release of IMS Connect user message exits, HWSIMSO0 and HWSIMSO1. These two user message exits will not be available in any future IMS Connect release.

## Library Changes for IMS Version 9

Changes to the IMS Library for IMS Version 9 include the addition of one title, a change of one title, organizational changes, and a major terminology change. Changes are indicated by a vertical bar (|) to the left of the changed text.

The IMS Version 9 information is now available in the DB2 Information Management Software Information Center for z/OS Solutions, which is available at http://publib.boulder.ibm.com/infocenter/dzichelp. The DB2 Information Management Software Information Center for z/OS Solutions provides a graphical user interface for centralized access to the product information for IMS, IMS Tools, DB2 Universal Database (UDB) for z/OS, DB2 Tools, and DB2 Query Management Facility (QMF™).

## New and Revised Titles

The following list details the major changes to the IMS Version 9 library:

- *IMS Version 9: IMS Connect Guide and Reference*

  The library includes new information: *IMS Version 9: IMS Connect Guide and Reference*. This information is available in softcopy format only, as part of the DB2 Information Management Software Information Center for z/OS Solutions, and in PDF and BookManager formats.

  IMS Version 9 provides an integrated IMS Connect function, which offers a functional replacement for the IMS Connect tool (program number 5655-K52). In this information, the term *IMS Connect* refers to the integrated IMS Connect function that is part of IMS Version 9, unless otherwise indicated.
- The information formerly titled *IMS Version 8: IMS Java User's Guide* is now titled *IMS Version 9: IMS Java Guide and Reference*. This information is available in softcopy format only, as part of the DB2 Information Management Software Information Center for z/OS Solutions, and in PDF and BookManager formats.
- To complement the IMS Version 9 library, a new book, *An Introduction to IMS* by Dean H. Meltz, Rick Long, Mark Harrington, Robert Hain, and Geoff Nicholls (ISBN # 0-13-185671-5), is available starting February 2005 from IBM Press. Go to the IMS Web site at www.ibm.com/ims for details.

## Organizational Changes

Organization changes to the IMS Version 9 library include changes to:

- *IMS Version 9: IMS Java Guide and Reference*
- *IMS Version 9: Messages and Codes, Volume 1*
- *IMS Version 9: Utilities Reference: System*

The chapter titled ″DLIModel Utility″ has moved from *IMS Version 9: IMS Java Guide and Reference* to *IMS Version 9: Utilities Reference: System*.

The DLIModel utility messages that were in *IMS Version 9: IMS Java Guide and Reference* have moved to *IMS Version 9: Messages and Codes, Volume 1*.

## Terminology Changes

IMS Version 9 introduces new terminology for IMS commands:

**type-1 command**
A command, generally preceded by a leading slash character, that can be entered from any valid IMS command source. In IMS Version 8, these commands were called *classic* commands.

**type-2 command**
A command that is entered only through the OM API. Type-2 commands are more flexible than type-2 commands and can have a broader scope. In IMS Version 8, these commands were called *IMSplex* commands or *enhanced* commands.

## Accessibility Enhancements

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products. The major accessibility features in z/OS products, including IMS, enable users to:

- Use assistive technologies such as screen readers and screen magnifier software

- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

## User Assistive Technologies

Assistive technology products, such as screen readers, function with the IMS user interfaces. Consult the documentation of the assistive technology products for specific information when you use assistive technology to access these interfaces.

## Accessible Information

Online information for IMS Version 9 is available in BookManager format, which is an accessible format. All BookManager functions can be accessed by using a keyboard or keyboard shortcut keys. BookManager also allows you to use screen readers and other assistive technologies. The BookManager READ/MVS product is included with the z/OS base product, and the BookManager Softcopy Reader (for workstations) is available on the IMS Licensed Product Kit (CD), which you can download from the Web at www.ibm.com.

## Keyboard Navigation of the User Interface

Users can access IMS user interfaces using TSO/E or ISPF. Refer to the *z/OS V1R1.0 TSO/E Primer*, the *z/OS V1R5.0 TSO/E User's Guide*, and the *z/OS V1R5.0 ISPF User's Guide, Volume 1*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

# Part 1. User’s Guide and Reference

# Chapter 1. Overview of IMS Connect

This chapter provides an overview of the IMS Connect product and how it works, and describes each of the IMS Connect components.

**In this chapter:**

## Introduction to IMS Connect

IMS Connect provides high performance communications for IMS between one or more TCP/IP or local OS/390 or z/OS clients and one or more IMS systems. IMS Connect provides the following features:

- Provides commands to manage the communication environment.
- Assists with workload balancing.
- Reduces design and coding efforts for client applications.
- Offers easier e-business access to IMS applications and operations with advanced security and transactional integrity.

As shown in Figure 1, IMS Connect enables TCP/IP or local OS/390 or z/OS clients to exchange messages through the IMS Open Transaction Manager Access (OTMA) facility or to exchange commands through the IMS Structured Call Interface (SCI) to the IMS Operations Manager (OM).

IMS Connect runs on an OS/390 or z/OS platform. For environmental details, see Chapter 4, “IMS Connect Definition and Tailoring,” on page 11.

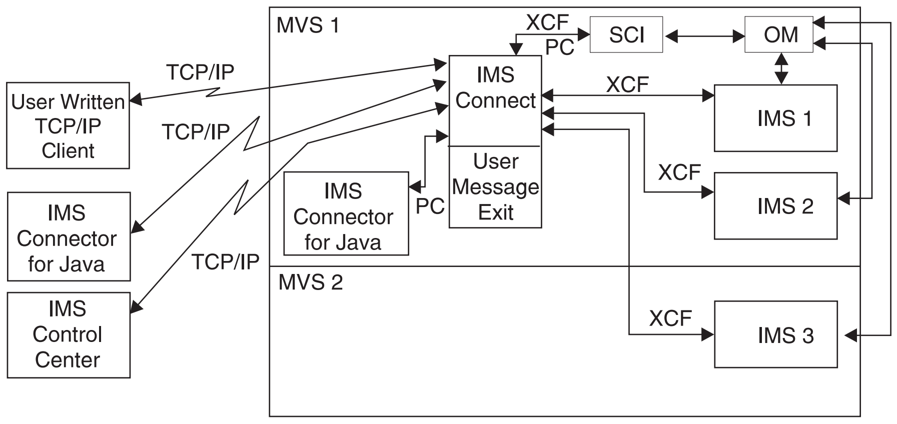


*Figure 1. System Overview*

IMS Connect performs router functions between TCP/IP clients and local option clients with datastores and IMSplex resources. Request messages received from TCP/IP clients, using TCP/IP connections, or local option clients, using the MVS Program Call (PC), are passed to a datastore through cross-system coupling facility (XCF) sessions. IMS Connect receives response messages from the datastore and then passes them back to the originating TCP/IP or local option clients.

IMS Connect supports TCP/IP clients communicating with socket calls, but it can also support any TCP/IP client that communicates with a different input data stream format. User-written message exits can execute in the IMS Connect address space to convert customer message format to OTMA message format before IMS Connect sends the message to IMS. The user-written message exits also convert OTMA message format to customer message format before sending a message back to IMS Connect. IMS Connect then sends output to the client.

If the datastore goes down, the status of the datastore is sent to IMS Connect from IMS OTMA through XCF. When the datastore is brought back up and restarted, IMS Connect is notified and automatically reconnects to the datastore. You do not need to manually reconnect to the datastore. (Note: IMS Connect will automatically reconnect only if it was originally connected to the datastore before the datastore went down.)

In addition to TCP/IP client communications, IMS Connect also supports local communication through the “local option”. This option provides a non-socket (non-TCP/IP) communication protocol for use between IBM WebSphere and IMS Connect in the OS/390 environment. Servlets that run in IBM WebSphere Application Server (WAS) for z/OS and use IMS Connector for Java, can communicate with IMS Connect through the local option.

IMS Connect also supports TCP/IP connections from the DB2 V8.1 UDB Control Center to exchange IMS Control Center commands and responses. A single IMS Connect can support communication between the IMS Control Center and any IMS within the sysplex.

**Requirement:** To use local option client communications, both IMS Connect and the IBM WebSphere instance where IMS Connector for Java is running must reside in the same MVS image.

**Restriction:** Local option client communications are supported only through IMS Connector for Java and the IMS Connect HWSJAVA0 user message exit.

IMS Connect supports IMS Version 7 and IMS Version 8. See Chapter 3, “IMS Connect and IMS Coexistence,” on page 9 for more information about IMS Connect and IMS coexistence.

## IMS Connect Components

As shown in Figure 2 on page 5, IMS Connect consists of eleven core components. These eleven components are:

- **Client communication component (CCC)**

  The Client communication component processes communication requests between the front-end driver(s) and the back-end driver(s).
- **Command component (CMD)**

  The command component processes commands received from the MVS console operator. For details of these commands, see Chapter 13, “IMS Connect Commands,” on page 137.
- **Datastore communication component (DCC)**

  The datastore communication component processes communication requests between the back-end driver(s) and the front-end driver(s).
- **Environment component (EVC)**

The environment component provides IMS Connect startup and termination services. The EVC loads IMS Connect modules, tables, and required storage areas; loads and calls the user message exits and user initialization exits; and terminates IMS Connect.

- **IMS Connect Base Primitive Environment (IMS Connect BPE)**

  The IMS Connect BPE component provides common system services to all IMS Connect components.
- **IMSplex communications component (ICC)**

  The IMSplex communications component processes communication requests between the front-end TIDC driver and the back-end IPDC driver.
- **IMSplex driver (IPDC)**

  The IMSplex driver, which is a back-end driver, enables IMS Connect to communicate with IMS by using the IMS SCI connection.
- **Local option communication component (LOCC)**

  The local option communication component processes communication requests between the front-end PCDC driver and the back-end drivers, OTDC and IPDC.
- **Local option driver (PCDC)**

  The local option driver, which is a front-end driver, provides the mechanism to communicate with clients by using local option, which is a non-socket communications protocol.
- **OTMA driver (OTDC)**

  The OTMA driver, which is a back-end driver, provides the mechanism to communicate with the IMS datastores by using an XCF connection to IMS OTMA.
- **TCP/IP driver (TIDC)**

  The TCP/IP driver, which is a front-end driver, provides the mechanism to communicate with clients by using a TCP/IP Sockets connection to the clients.

Figure 2 displays the layout of each IMS Connect component.

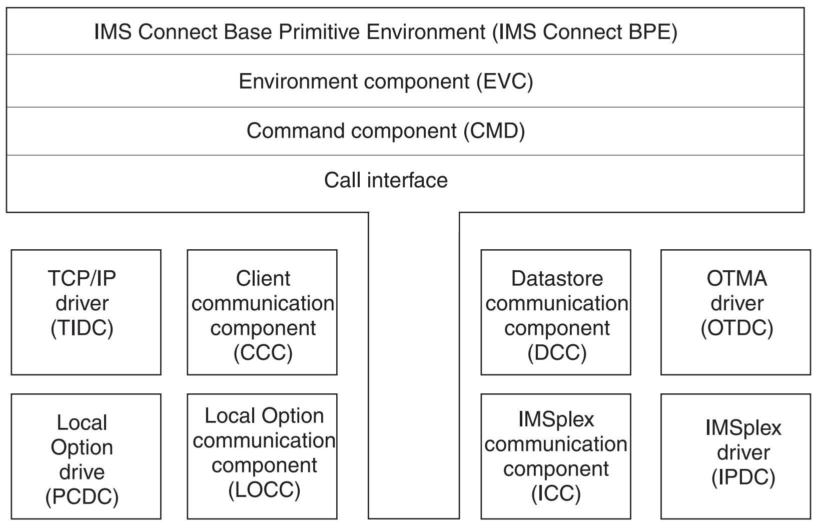

IMS Connect Base Primitive Environment (IMS Connect BPE)
Environment component (EVC)
Command component (CMD)
Call interface
TCP/IP driver (TIDC)
Client communication component (CCC)
Datastore communication component (DCC)
OTMA driver (OTDC)
Local Option drive (PCDC)
Local Option communication component (LOCC)
IMSplex communication component (ICC)
IMSplex driver (IPDC)

*Figure 2. IMS Connect Component Layout*

IMS Connect uses the driver components (TIDC, PCDC, IPDC, and OTDC) to isolate the core components from the communication software. The TCP/IP driver is used to communicate with TCP/IP clients using the TCP/IP communications protocol. The local option driver (PCDC) is used to communicate with the local option clients. The IMS OTMA driver is used to communicate with the datastores (IMSs) using the IMS OTMA communications protocol. The IMSplex driver is used to communicate with IMS Operations Manager (OM) using SCI. Communication between components takes place using the call interface service. The call interface provides the encapsulation and isolation of structures between the components. Each IMS Connect component provides its own set of functions, which it registers with the call interface. When a component requires that a function be performed by another component, the first component calls the call interface using the following parameters:

- Component name to which the request is to be forwarded
- Function the component is to perform
- Parameters required for the function

The call interface uses a function work element (FWE) to carry information between components.

The IMS Connect Base Primitive Environment (IMS Connect BPE) is a common system service base upon which IMS Connect is built. IMS Connect initializes the IMS Connect BPE in the IMS Connect address space. The IMS Connect BPE provides IMS Connect with these services:

- Environment
- Storage
- Serialization
- Tracing

# Chapter 2. IMS Connect Prerequisites

This chapter lists the software prerequisites for IMS Connect. IMS Connect requires the following software:

- z/OS 1.4, or later versions, releases, and modification levels.
- IMS Version 7 or later, with the required maintenance APARs applied for each version. See Chapter 3, "IMS Connect and IMS Coexistence," on page 9 for more information about IMS and IMS Connect coexistence.
- TCP/IP Version 1.4 or later.
- IMS 8 APAR PQ87087 and IMS 9 APAR PQ87088.
- Resource Access Control Facility (RACF) Version 1.9.2 or later, or equivalent product.

  **Note:** This document uses the term *RACF* when referring to RACF or an equivalent product.
- If you plan to use the local option for client communications, there are additional software requirements. See *IMS Connector for Java User's Guide*, which is available at www.ibm.com/ims.
- If you plan to use IMS Connect IMSplex support, IMS Version 8.1 or later is required.
- If you plan to use the IMS Connect IMSplex support, DB2 UDB Version 8.1 with Fixpack 1 (which contains the IMS Control Center) is required.
- If you plan to use SSL (Secure Sockets Layer), z/OS Version 1.4 System SSL, a sub-component of z/OS Cryptographic System Services, is required. For information about z/OS encryption support available with the z/OS Cryptographic System Services SSL module, see *z/OS System Secure Sockets Layer Programming* SC24-5901-02.
- If you plan to use z/OS Version 1.4 or higher, IMS Connect must be a superuser.

# Chapter 3. IMS Connect and IMS Coexistence

IMS Connect supports IMS Version 7 and IMS Version 8 according to the following guidelines:

- For IMS Connect to communicate with IMS Version 7, the following APARs must be installed:
  - PQ33929
  - PQ33996
  - PQ34229
  - PQ34542
  - PQ37322
- For IMS Connect to communicate with IMS Version 8, the following IMS APARs must be installed:
  - PQ87087
  - PQ59431
  - PQ63231
- For IMS Version 9, IMS Connect is an integrated function, which offers a functional replacement for the IMS Connect tool (program number 5655-K52).

# Chapter 4. IMS Connect Definition and Tailoring

This chapter describes the tasks of defining and tailoring IMS Connect. It provides detailed information about configuring and customizing the IMS Connect environment, as well as guidelines and procedures for using and invoking IMS Connect.

The IMS Connect package provides the following components that enable TCP/IP clients to exchange messages with IMS for transaction processing and exchange IMS commands with IMS Operations Manager for command processing. The two components also enable z/OS WebSphere clients to exchange messages with IMS:

**IMS Connect**
A z/OS application program that provides the following services:
- Communication to TCP/IP clients using TCP/IP connections
- Communication to IMS using XCF connections to OTMA
- Communication to z/OS WebSphere clients using Program Call Interface
- Communication to IMS using IMS Structure Call Interface (SCI) connections to OM

**IMS Connect BPE**
A system-service component that supports IMS Connect.

In addition to these two components (which include the eleven core components that constitute the IMS Connect application program), the following are included in the IMS Connect package:
- Sample user message exits: HWSSMPL0, HWSSMPL1, HWSJAVA0, HWSUINIT and HWSYDRU0.
- IMSplex message exits: HWSCSLO0 and HWSCSLO1. These Control Center user message exits are provided.
- The files HWSIMSCB, HWSIMSEA, HWSEXPRM, HWSOMPFX, HWSXIB, and HWSXIBDS, which are required in order to use the sample IMS Connect exits.
- Several files are included for the IMS Connect Extensions (event recording) support. See Appendix G, "HWSTECL0 User Exit," on page 267 for a list of the files.

**In this chapter**:
- "Defining the IMS Connect Environment"
- "Setting IMS Connect Allocations" on page 28
- "Invoking IMS Connect" on page 28
- "Customizing IMS Connect" on page 29
- "JCL to Print IMS Connect RECORDER Output" on page 37

## Defining the IMS Connect Environment

This section describes how to prepare the environment for IMS Connect. To use this information, you need a working knowledge of IMS transaction processing, RACF, IMS OTMA, and TCP/IP.

As the following IMS Connect startup JCL statements illustrate, both IMS Connect and IMS Connect BPE have configuration members:

```
//HWS       PROC  RGN=4096K,SOUT=A,
//             BPECFG=BPECFGHT,
//             HWSCFG=HWSCFG00
//*
//*********************************************************************
//*  BRING UP AN IMS CONNECT                                          *
//*********************************************************************
//STEP1    EXEC PGM=HWSHWS00,REGION=&RGN,TIME=1440,
//             PARM='BPECFG=&BPECFG,HWSCFG=&HWSCFG'
//STEPLIB  DD   DSN=SHWSRESL,DISP=SHR
//         DD   DSN=SDFSRESL,DISP=SHR
//         DD   DSN=CEE.SCEERUN,UNIT=SYSDA,DISP=SHR
//         DD   DSN=SYS1.CSSLIB,UNIT=SYSDA,DISP=SHR
//         DD   DSN=GSK.SGSKLOAD,UNIT=SYSDA,DISP=SHR
//PROCLIB  DD   DSN=USER.PROCLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
//HWSRCORD DD   DSN=HWSRCDR,DISP=SHR
```

**Note:** The SDFSRESL library, which is the IMS Version 8.1 or later execution library, is required by IMS Connect when IMSplex support is used. If IMSplex support is not used, then the SDFSRESL library is not required. IMS Connect requires the CEE.SCEERUN, SYS1.CSSLIB, and GSK.SGSKLOAD libraries (which are the C execution and z/OS system SSL libraries) only when SSL support is used.

## Configuring IMS Connect

IMS Connect supports communication between one or more TCP/IP clients and IMS systems. IMS Connect uses TCP/IP for communication with clients and IMS OTMA for communication with IMS. It also provides a mechanism to start or stop TCP/IP clients or datastores through the use of commands.

You can configure multiple IMSs on multiple MVS systems within a single sysplex and distribute the client request to the IMSs (datastores).

To configure IMS Connect, perform the following actions:

1. Authorize the Application Program Family (APF).
2. Update the Program Properties Table (PPT) in MVS. Updating the PPT allows IMS Connect to run in authorized supervisor state and in key 7.
3. Create an IMS Connect configuration member to hold the configuration statements that IMS Connect uses during initialization.
4. Define IMS Connect security.

### Authorizing IMS Connect to the APF

SHWSRESL, the resident library (RESLIB) in which the IMS Connect modules reside, must be authorized to the APF. Create and run a JCL job that authorizes SHWSRESL to the APF.

### Updating the MVS PPT

Because IMS Connect is executed in supervisor state and key 7, add an entry for it in the MVS Program Properties Table (PPT) as follows:

1. Edit the SCHEDxx member of the SYS1.PARMLIB data set.
2. Add the required entries to the MVS PPT:

   For TCP/IP communications only, add the following entry in the MVS PPT:

   ```
   PPT PGMNAME(HWSHWS00)     /* PROGRAM NAME = HWSHWS00           */
            CANCEL           /* PROGRAM CAN BE CANCELED           */
            KEY(7)           /* PROTECT KEY ASSIGNED IS 7         */
   ```

```
         SWAP             /* PROGRAM IS SWAPPABLE              */
         NOPRIV           /* PROGRAM IS NOT PRIVILEGED         */
         DSI              /* REQUIRES DATA SET INTEGRITY       */
         PASS             /* CANNOT BYPASS PASSWORD PROTECTION */
         SYST             /* PROGRAM IS A SYSTEM TASK          */
         AFF(NONE)        /* NO CPU AFFINITY                   */
         NOPREF           /* NO PREFERRED STORAGE FRAMES       */
```

If you are using local option for client communications, either by itself or with TCP/IP communications, add the following entry in the MVS PPT:

```
PPT PGMNAME(HWSHWS00)     /* PROGRAM NAME = HWSHWS00           */
         CANCEL           /* PROGRAM CAN BE CANCELED           */
         KEY(7)           /* PROTECT KEY ASSIGNED IS 7         */
         NOSWAP           /* PROGRAM IS NOT SWAPPABLE          */
         NOPRIV           /* PROGRAM IS NOT PRIVILEGED         */
         DSI              /* REQUIRES DATA SET INTEGRITY       */
         PASS             /* CANNOT BYPASS PASSWORD PROTECTION */
         SYST             /* PROGRAM IS A SYSTEM TASK          */
         AFF(NONE)        /* NO CPU AFFINITY                   */
         NOPREF           /* NO PREFERRED STORAGE FRAMES       */
```

**Note:** The only difference between the two PPT entries is the use of SWAP or NOSWAP.

3. To make the changes effective, do either of the following:
   - Re-IPL your MVS system.
   - Issue the MVS `SET SCH=` command.

## Creating the IMS Connect Configuration Member

Specify the environment for IMS Connect as a member in your PROCLIB data set. IMS Connect uses the information it retrieves from the member to establish communication with IMS and TCP/IP. You can define several configuration members in the PDS to select from during IMS Connect startup. Specify the member name to use in the HWSCFG= parameter of the IMS Connect startup JCL (see the previous IMS Connect startup JCL example on 11).

***IMS Connect Configuration Statement Parameters:*** Specify the values for some of the parameters that define the way in which IMS Connect is to communicate with TCP/IP and IMS OTMA in the IMS Connect configuration member. The IMS Connect configuration member contains the following configuration statements: HWS, TCPIP, DATASTORE, and IMSPLEX. Because the configuration member must be in a data set whose format is fixed block, 80-byte record length, a statement must be carried over to as many subsequent lines as required if the configuration statement is longer than 80 characters. Configuration statements should have no imbedded spaces or continuation characters at the ends of lines that must be continued to the next line. The following is the format of the configuration statements.

**HWS** Specify only one IMS Connect.

The HWS statement includes only three keyword parameters, which are as follows:

**ID=** The IMS Connect name, which:
- Consists of alphanumeric character data
- Begins with an alphabetic character
- Has a length between 1 and 8 characters

**RACF=**
At IMS Connect startup time, determines whether or not the password and user ID (provided by either the client application or a

user exit routine) are passed to RACF for authentication. This setting can also be changed using the IMS Connect `SETRACF` command. Set it to yes or no as follows:

- Y
- N (this is the default)

**RRS=** Specifies whether RRS communication is to be enabled or disabled. Set RRS to yes or no as follows:

- Y
- N (this is the default)

**XIBAREA=**
Specifies the number of fullwords allocated for the XIB user area. Both the user initialization exit routine and the user message exit routines can access and modify the XIB user area. The default value is 20; the maximum value is 500. If you do not specify a value for this parameter, or you specify a value outside of the 20 to 500 range, the system uses the default value of 20.

**TCPIP** Specify only one TCPIP.

The TCPIP statement keyword parameters are as follows:

**ECB=** Specifies whether TCP/IP exit or ECB (Event Control Block) processing is to be used. ECB processing enhances IMS Connect performance by increasing throughput.

Set *ECB* to yes or no as follows:

- Y
- N (this is the default)

When ECB= N is specified (or left blank), IMS Connect executes with TCP/IP driving an IMS Connect exit.

When ECB=Y is specified, IMS Connect executes with TCP/IP driving IMS Connect with the posting of an ECB.

**EXIT=** A 1 to 8 alphanumeric character field set to the name of the TCP/IP user exit message that receives control for messages received from and sent to TCP/IP clients. More than one exit can be defined as `EXIT=(EZAEXIT,EZBEXIT,EZCEXIT)` to a maximum of 254 user-defined Message Exits (HWSJAVA0 is the 255th exit). These exits support users other than IMS Connector for Java for OTMA linkage through IMS Connect to IMS. Do not include HWSJAVA0 in the `EXIT=` list. IMS Connect automatically loads the HWSJAVA0 exit, which is shipped with IMS Connect, to enable IMS to support the IMS Connector for Java application. The user message exits for IMSplex support are HWSCSLO0 and HWSCSLO1 and must be specified here to ensure activation of the IMS Control Center.

**Note:** HWSUINIT is an IMS Connect exit but is not a user message exit; therefore, do not add HWSUINIT to the EXIT= parameter. If you add HWSUINIT to the EXIT= parameter, IMS Connect will abend.

**HOSTNAME=**
A 1 to 8 alphanumeric character field set to the TCP/IP JOBNAME.

**IPV6=** At IMS Connect startup time, determines whether or not Internet Protocol Version 6 (IPV6) is enabled. Set this parameter to yes or no as follows:

- Y
- N (this is the default)

When IPV6=Y is specified, IPV6 is used.

When IPV6=N is specified (or left blank), IPV4 is used.

**Note:** If you use IPV6, z/OS Version 1.4 or later is required.

**MAXSOC=**
A decimal field set to the maximum number of sockets per IMS Connect that an IMS Connect can open. This value must be a number between 50 and 65535. The value specified will result in one socket dedicated to a Listen State per port and the remainder available for connections. Therefore, if you specify 80 and have five ports, 75 physical connections can be made. The other five are used for Listen sockets. The default value is 50.

**PORTID=**
A 1- to 8-character decimal field to define TCP/IP ports, or a 5-character field with the value of LOCAL to define the local option connection. For TCP/IP port communications, specify the port number or numbers that will bind to the socket. You can define up to 50 ports. Port numbers must be within the range of 1 to 65535 and must not conflict with other ports selected in the TCP/IP domain.

To enable the local option connection, specify a value of LOCAL. You can enable the local option connection and define up to 50 TCP/IP ports for TCP/IP communications. Some PORTID configuration examples include:

```
PORTID=(9999,8888,7777)
PORTID=(LOCAL)
PORTID=(6666,5555,4444,3333,LOCAL)
```

**RACFID=**
A 1 to 8 alphanumeric character field set to the default RACF ID for exits to pass to OTMA for security checking if the RACF ID has not explicitly been set in the incoming message or by the user exit.

**SSLENVAR=**
The member name of the SSL initialization file.

**SSLPORT=**
A 1 to 8 numeric character decimal field to define Secure Socket Layer (SSL) ports. For SSL port communication, specify the port number that will bind to the socket. You can define up to 50 ports, which must be numbered within the range of 1 to 65535. These ports must not conflict with any other ports selected in the TCP/IP domain or those selected under the PORTID parameter as basic TCP/IP ports. An example of an SSLPORT configuration is:

```
SSLPORT=(9998,8887,7776)
```

**TIMEOUT=**
A decimal integer field to disconnect the client. The timeout interval is in hundredths of seconds. The maximum value of timeout is

2147483647 (X'7FFFFFFF') and the default is 0 (which means no timeout). The range is from 0 to 2147483647.

IMS Connect uses the timeout value to determine the amount of time to wait for a response from IMS that is being sent to the client. This timeout value is used to prevent the client from appearing to be "hung." A hang condition occurs when the IMS host application is not responding, because either:

- The IMS program for this transaction code is stopped
- The dependent region that would run the transaction is not active
- The IMS host application is looping

The client sets a second timeout value in the IRM (IMS Request Message) header field IRM_TIMER for use with a READ to OTMA following a RESUME TPIPE command (see "Resume Tpipe/Receive Protocol for Asynchronous Output" on page 108 for more information), and the ACK following the READ(s) for a RESUME TPIPE.

This timeout value is also used to disconnect a client and not send data following a client socket connection. If the timeout value is set to 10 seconds, and the client application performs a socket connection, then the client application has 10 seconds in which to send the transaction code and data. If the socket connection is made and the client application delays for more than 10 seconds, the socket connection terminates. This timeout value on an IMS Connect read of the client only applies to the wait time between the socket connection and the first input from the client application. The timeout function is not activated between reads but only between the connection and the first IMS Connect read of the client application input.

**DATASTORE**

To access IMS OTMA, specify each datastore with which the IMS Connect communicates through IMS OTMA.

The DATASTORE statement keyword parameters are as follows:

**APPL=**

A 1 to 8 alphanumeric character field set to the TCP/IP APPL name defined to RACF in the PTKTDATA statement. This parm is optional and will default to blanks. If you are using PassTicket and user message exits, HWSIMSO0, HWSIMSO1, or both, you must specify the APPL on the DATASTORE statement.

**DRU=** A 1 to 8 alphanumeric character field. The DRU keyword enables you to specify your own OTMA destination resolution user exit name that is to be passed to OTMA. The DRU exit is required to support asynchronous output to IMS Connect clients. The default is DFSYDRU0, but you can write your own exit. See the DRU exit information in the IMS *Open Transaction Manager Access Guide* for more information about OTMA DRU exits.

**GROUP=**

The XCF group name for the IMS OTMA. IMS Connect uses this value to join the appropriate XCF group(s). Because IMS Connect and IMS must be in the same XCF group in order to communicate, this group name must match the XCF group name that you define to IMS (*GRNAME*) in the IMS startup JCL (for example,

″OTMA=Y,***GRNAME***=&***GROUP***,USERVAR=&MEMBER″,...). Each IMS Connect can join any number of groups.

**ID=** The datastore name, which:

- Consists of alphanumeric character data
- Begins with an alphabetic character
- Has a length between 1 and 8 bytes

This ID must match the datastore ID that is supplied by the client. For IMS Connector for Java clients, this ID must match the name that is specified in the IMS Interaction Spec for IMS Connector for Java. For non-IMS Connector for Java clients, the ID must match the datastore ID that is placed in the IMS Request Message (IRM) that is sent to IMS Connect (see “How IMS Connect Communicates with a TCP/IP Client” on page 39).

**Note:** This ID name cannot be the same name as the *tmember* name on the IMSPLEX statement.

**MEMBER=**

The XCF member name that identifies IMS Connect in the XCF group specified by the GROUP parameter. This name is the XCF name that IMS uses to communicate with IMS Connect in that XCF group. This XCF member name for IMS Connect must be unique in the datastore definitions for all datastores that are members of the same XCF group.

**TMEMBER=**

The XCF member name for IMS that IMS Connect uses in order to communicate with an IMS in its XCF group. This target member name must match the member name IMS uses when it joins the XCF group. The XCF member name for IMS is specified in the IMS startup JCL (for example, ″...,OTMA=Y,GRNAME=&GROUP, ***OTMANM***=&***TMEMBER***,...″). Each datastore definition within an IMS Connect configuration member must contain a unique tmember name.

**IMSPLEX**

To access IMS Operations Manager (OM), specify each IMSPLEX that IMS Connect communicates with through the IMS Structure Call Interface (SCI).

The IMSPLEX statement keyword parameters are as follows:

**MEMBER=**

This name is passed to the SCI as the name of the IMS Connect that is communicating with the IMS OM through the SCI.

**TMEMBER=**

This name is the name of the SCI to which IMS Connect communicates. The *tmember* name:

- consists of alphanumeric character data
- begins with an alphabetic character
- has a length between 1 and 5 bytes
- must be the name specified in the SCI initialization proclib member -- IMSPLEX(NAME=*name*)

**Note:** The *tmember* name cannot be the same name as the ID name on the DATASTORE statement.

See Figure 3 for an example of a simple system configuration.

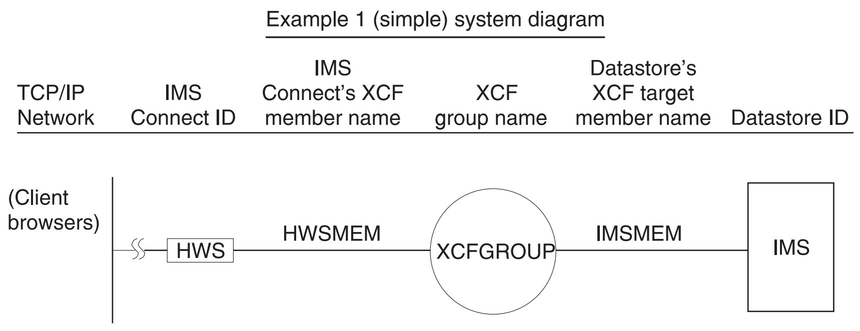


*Figure 3. Simple System Configuration*

- In the following IMS Connect configuration member, the IMS Connect ID is defined as `HWS`. This IMS Connect is configured to include the ports defined for TCP/IP communications and the IMS OTMA group and member names for communication with IMS.
- The TCP/IP configuration defines the HOSTNAME as `MVSTCPIP`, the RACFID as `RACFID`, the PORTID as `9999`, and the EXIT as `HWSSMPL0`.
- The datastore configuration defines the ID as `IMS`, the GROUP as `XCFGROUP`, the MEMBER as `HWSMEM`, and the TMEMBER as `IMSMEM`.

```
****************************************************
* IMS Connect EXAMPLE 1 CONFIGURATION FILE
****************************************************
HWS (ID=HWS,RACF=N,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888,
EXIT=(HWSSMPL0))
DATASTORE  (ID=IMS,GROUP=XCFGROUP,MEMBER=HWSMEM,TMEMBER=IMSMEM,DRU=HWSYDRU0)
```

**Important:** In all of the example configurations shown, you can continue parameters beyond an 80–column line by using any combination of the following techniques:

- Inserting a comma followed by three blanks, then continuing the parameter on the next line. An example of this technique is shown above in the `IMS Connect Example 1 Configuration File`.
- Using all 80 columns of a line, then continuing in the next statement. You do not need to use a continuation indicator (such as an ″x″ in column 72).

See Figure 4 on page 19 for an example of a more complex system configuration.

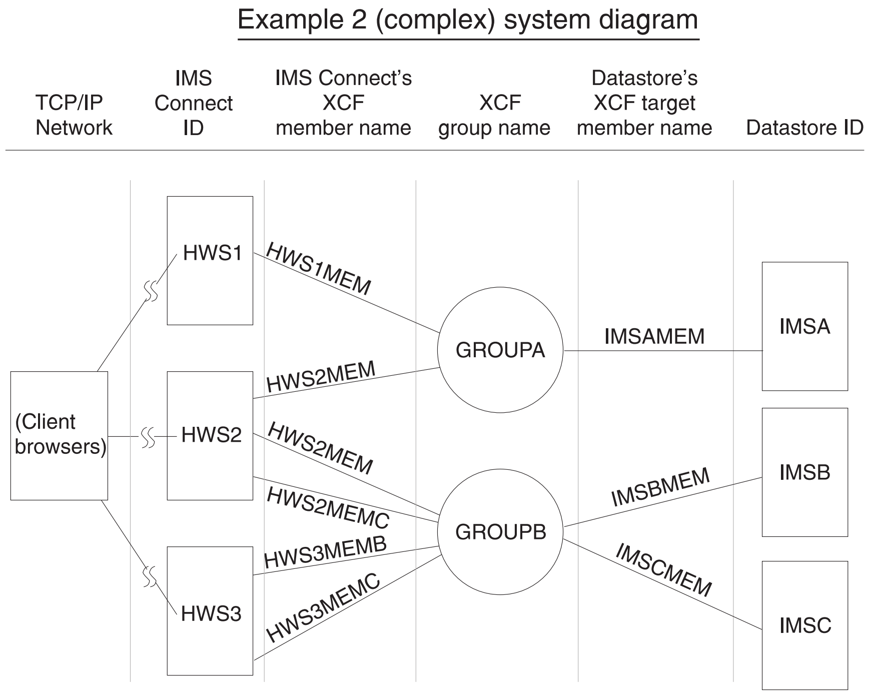


*Figure 4. Complex System Configuration*

- In this example, three IMS Connects are configured. Each IMS Connect has its own configuration member.
- Each IMS Connect uses a different port number for TCP/IP communications and can belong to multiple XCF groups.
- One or more IMSs can belong to each XCF group.
- When defining multiple datastores that belong to the same XCF group in a single IMS Connect configuration member, the XCF member name for that IMS Connect must be unique in each DATASTORE statement. However, if the datastores are members of different XCF groups, the XCF member names can be the same for different datastores within a single IMS Connect configuration member.

For example, observe that the XCF member name for IMS Connect in the IMSA and IMSB DATASTORE statements in the HWS2 configuration member in the configuration example, HWSMEM2, is the same for both DATASTORE statements. The IMSA and IMSB datastores are members of different XCF groups—GROUPA and GROUPB, respectively—so the XCF member names **can** be identical. Note that these member names could have been made unique, for example, HWS2MEMA and HWS2MEMB, but it is not necessary to do so. However, the XCF member names for IMS Connect in the IMSB and IMSC DATASTORE statements in the HWS2 configuration member are different because the IMSB and IMSC datastores are members of the same XCF group, GROUPB.

```
****************************************************
* HWS EXAMPLE 2 CONFIGURATION MEMBER FOR HWS1
****************************************************
HWS (ID=HWS1,RACF=N,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888,EXIT=(HWSSMPL0))
DATASTORE      (ID=IMSA,GROUP=GROUPA,MEMBER=HWS1MEM,TMEMBER=IMSAMEM,DRU=HWSYDRU0)
****************************************************

****************************************************
* HWS EXAMPLE 2 CONFIGURATION MEMBER FOR HWS2
****************************************************
```

```
HWS (ID=HWS2,RACF=N,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9998),MAXSOC=2000,TIMEOUT=8888,EXIT=(HWSSMPL0))
DATASTORE      (ID=IMSA,GROUP=GROUPA,MEMBER=HWS2MEM,TMEMBER=IMSAMEM,DRU=HWSYDRU0)
DATASTORE      (ID=IMSB,GROUP=GROUPB,MEMBER=HWS2MEM,TMEMBER=IMSBMEM,DRU=HWSYDRU0)
DATASTORE      (ID=IMSC,GROUP=GROUPB,MEMBER=HWS2MEMC,TMEMBER=IMSCMEM,DRU=HWSYDRU0)
***************************************************

***************************************************
* HWS EXAMPLE 2 CONFIGURATION MEMBER FOR HWS3
***************************************************
HWS (ID=HWS3,RACF=Y,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9997),MAXSOC=2000,TIMEOUT=8888,
EXIT=(HWSSMPL0))
DATASTORE      (ID=IMSB,GROUP=GROUPB,MEMBER=HWS3MEMB,TMEMBER=IMSBMEM,DRU=HWSYDRU0)
DATASTORE      (ID=IMSC,GROUP=GROUPB,MEMBER=HWS3MEMC,TMEMBER=IMSCMEM,DRU=HWSYDRU0)
***************************************************

***************************************************
* HWS EXAMPLE OF INCLUDING THE SUPPORT FOR CONTROL CENTER
***************************************************
HWS (ID=HWS4,RACF=Y,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999,LOCAL),MAXSOC=2000,TIMEOUT=8800,
EXIT=(HWSCSLO0,HWSCSLO1,HWSSMPL1))
IMSPLEX (MEMBER=IMSPLEX1,TMEMBER=PLEX1)
***************************************************

***************************************************
* HWS EXAMPLE OF INCLUDING THE SUPPORT FOR CONTROL CENTER AND IPV6
***************************************************
HWS (ID=HWS4,RACF=Y,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8800,
EXIT=(HWSCSLO0,HWSCSLO1,HWSSMPL1),IPV6=Y)
IMSPLEX (MEMBER=IMSPLEX1,TMEMBER=PLEX1)
***************************************************

***************************************************
* HWS EXAMPLE OF INCLUDING THE APPL NAME FOR PASSTICKET SUPPORT
***************************************************
HWS (ID=HWS5,RACF=Y,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8800,
EXIT=(HWSSMPL0)
DATASTORE      (ID=IMS,GROUP=XCFGROUP,MEMBER=HWSMEM,TMEMBER=IMSMEM,DRU=HWSYDRU0,APPL=APPLID1)
***************************************************

***************************************************
* HWS EXAMPLE OF INCLUDING THE SUPPORT FOR SSL
***************************************************
HWS (ID=HWSG7,RACF=N,XIBAREA=20))
TCPIP (HOSTNAME=TCIPI,PORTID=(9998),SSLPORT=(9999),SSLENVAR=SSLENVAR,EXIT=(HWSSMPLO))
DATASTORE (ID=SOCKEYE,MEMBER=COHO,TMEMBER=CHINOOK,GROUP=SALMON)
***************************************************
```

## Enabling Support for Internet Protocol Version 6

Internet Protocol Version 6 (IPV6) is the next generation of the Internet Protocol designed to replace the current Internet Protocol Version 4 (IPV4). The features of IPV6 include:

- Dramatically larger address spaces
- Global unique and hierarchical addressing, based on prefixes rather than on address classes to keep routing tables small and backbone routing efficient
- Multicasting instead of broadcasting
- A class of service to distinguish between different types of traffic
- A built-in mechanism for autoconfiguration of network interfaces
- Built in authentication and encryption
- Mobile IP support
- Encapsulation of itself and other protocols
- Transition methods to migrate from IPV4
- Compatibility methods to coexist and communication with IPV4

**Related Reading:** For more information about IPV6, see *IPv6 Network and Application Design Guide*.

To enable IPV6 support for IMS Connect, do the following:

- Ensure that IMS Connect is running on z/OS V1R4.
- Customize the BPXPRMxx member, as follows:

  ```
  FILESYSTYPE Type(INET) Entrypoint(EZBPFINI)
  NETWORK DOMAINNAME(AF_INET)
  DOMAINNUMBER(2)
    MAXSOCKETS(2000)
    TYPE(INET)

  NETWORK DOMAINNAME(AF_INET6)
  DOMAINNUMBER(19)
    MAXSOCKETS(3000)
    TYPE(INET)
  ```

  Then recycle the TCP/IP stack. For more information about customizing this member, see *z/OS UNIX System Services Planning*.
- Customize the IMS Connect configuration member with the `IPv6` parameter, as described in "IMS Connect Configuration Statement Parameters" on page 13.
- For each of the READ subroutines in the list below that you use, determine whether the EXPREA_IPV6 bit is turned on in the EXPREA_FLAG2 field of the READ subroutine. If it is turned on, IPV6 is enabled. Then map EXPREA_SOCKET6 to the AF_INET6 socket address structure. See "READ Subroutine" on page 63 for more information.
  - HWSJAVA0
  - HWSSMPL0
  - HWSSMPL1

The IP address format when IPV6 is enabled is described in "VIEWHWS" on page 143.

## Defining IMS Connect Security

You can start IMS Connect as a job or as a procedure.

If the datastore (which is IMS) is RACF protected, you have to start IMS Connect as a job with the JOB card specifying a valid *USERID* in order to make the connection from IMS Connect to IMS, or you can use the RACF started procedure table. The *USERID*=&*userid* parameter specified in the JOB card of the IMS Connect job JCL is used as the security vehicle to ensure IMS Connect access to IMS. &USERID must have READ access to *IMSXCF.group.member*. IMS OTMA provides security for the IMS XCF connection by defining and permitting *IMSXCF.group.member* in the RACF *FACILITY* class. For details, see the section dealing with security for OTMA in the *IMS Open Transaction Manager Access Guide*.

**Requirement:** To configure security for the local option using RACF, you must add HWS.ICON_NAME as the SAF facility class name (whether you configured security with the IMS Connect configuration member or SETRACF command). ICON_NAME is how IMS Connect is defined in the ID parameter of the HWS statement in the IMS Connect configuration member. The resource that must access IMS Connect is the Websphere Application Server (WAS), and UPDATE authority is required to update the RACF profile.

## Configuring the IMS Connect Base Primitive Environment (BPE)

The IMS Connect address space is built on top of the IMS Connect BPE. Generally, you do not need to work with the IMS Connect BPE. However, your IBM service representative could request that you change the default settings for certain IMS Connect BPE functions such as storage management, internal tracing, dispatching, and other system-service functions.

This section describes how to define the configuration data set member, and includes some examples.

### Changing the IMS Connect BPE Configuration Parameter PROCLIB Member

The IMS Connect BPE configuration parameter PROCLIB member defines IMS Connect BPE execution environment settings for the IMS Connect address space. You specify the PROCLIB member name by coding `BPECFG=`*member_name* on the EXEC PARM= statement in the IMS Connect address space startup JCL, as shown in the following example:

```
EXEC HWSHWS00,PARM='BPECFG=BPECFGHW'
```

You can use the IMS Connect BPE configuration parameter PROCLIB member to specify the following items:

- The language used for IMS Connect BPE and IMS Connect messages
- The trace level settings for IMS Connect BPE and IMS Connect internal trace tables

These are the keywords that are available for the BPE configuration parameter PROCLIB member:

- LANG=
- TRCLEV=

**Recommendation:** Avoid coding statements in the IMS Connect BPE configuration member that specify definitions for the same resources more than one time. For example, multiple TRCLEV statements for the same trace table type, or multiple EXITMBR statements for the same IMS Connect. BPE uses the **last** statement it encounters in the member. Any values that are specified on earlier duplicate statements are ignored. A message BPE0017I is issued for each duplicate found.

***LANG:***

```
►►──LANG=ENU──────────────────────────────────────────────────────────►◄
```

The LANG parameter specifies the language used for IMS Connect BPE and IMS Connect messages. ENU is for US English, which is currently the only supported language.

***TRCLEV:***

```
►►──TRCLEV=──(type,level,component─┬─────────────────────┬─)──────────►◄
                                   └─,PAGES=──num_pages──┘
```

The TRCLEV parameter specifies the trace level for a trace table, and optionally the number of pages of storage allocated for the trace table. TRCLEV= controls the level of tracing (the amount of detail traced) for each specified trace table type.

BPE-managed trace tables are areas in storage where IMS Connect BPE, and IMS Connect, can trace diagnostic information about events going on within the address space. Each trace table has a trace table type associated with it. A trace table's type refers to the kind of events that are traced into that table. For example, the BPE DISP trace table contains entries related to events in the BPE dispatcher.

IMS Connect BPE-managed trace tables are internal in-core tables only. Trace records are not written to any external data sets. Some trace table types are defined and owned by IMS Connect BPE itself. These are known as system trace tables. IMS Connect also defines its own trace tables. These are known as component trace tables or user-product trace tables.

*type*
Specifies the type of trace table.

You can code the following values for IMS Connect BPE-defined trace tables:

*****
Specify as `TRCLEV=(*,`*`level`*`,BPE)`.

Specifying a type of ***** enables you to set the default trace level (and optionally, the default number of pages per trace table) for **all** IMS Connect BPE-defined trace table types. If you use the ***** type, make sure it is the first TRCLEV statement for IMS Connect BPE-defined trace table types in your PROCLIB member. You can then code additional TRCLEV statements for specific IMS Connect BPE types to selectively override the defaults.

**Recommendation:** Code a TRCLEV statement with a type of ***** for IMS Connect BPE traces, specifying a level of at least LOW as your first TRCLEV statement for IMS Connect BPE-defined trace table types. Using this coding ensures that at least some tracing is done for all BPE trace tables. It also ensures that any new trace table types that are added in the future will be turned on in your system, even if you have not modified your IMS Connect BPE configuration parameter PROCLIB member to explicitly add a TRCLEV statement.

**AWE**
Specify as `TRCLEV=(AWE,`*`level`*`,BPE)`.

The asynchronous work element (AWE) services trace table traces AWE server creation and deletion and AWE processing requests. The default number of pages for this table is 2.

**CBS**
Specify as `TRCLEV=(CBS,`*`level`*`,BPE)`.

The control block services trace table traces requests for control block storage. The default number of pages for this table is 4.

**CMD**
Specify as `TRCLEV=(CMD,`*`level`*`,BPE)`.

The command trace table traces the first 48 characters of each command processed by IMS Connect BPE. The default number of pages for this table is 2.

**Recommendation:** The CMD trace table performance impact is very low. To ensure that a command history is kept for diagnostics, specify a LEVEL of at least LOW.

**DISP**
Specify as `TRCLEV=(DISP,`*`level`*`,BPE)`.

The dispatcher trace table traces BPE dispatcher activity. The default number of pages for this table is 4.

**ERR**
Specify as `TRCLEV=(ERR,`*`level`*`,BPE,PAGES=`*`num_pages`*`)`.

The error trace table traces error events within an IMS Connect BPE address space. The default number of pages for this table is 2.

**Restriction:** You cannot set the level for the ERR trace table. BPE forces the level to HIGH to ensure that error diagnostics are captured. Any level that you specify for the ERR trace table is ignored. You can, however, specify the number of pages for the ERR trace table on the TRCLEV statement.

**LATC**
Specify as `TRCLEV=(LATC,`*`level`*`,BPE)`.

The latch trace table traces IMS Connect BPE latch management (serialization) activity. The default number of pages for this table is 4.

**SSRV**
Specify as `TRCLEV=(SSRV,`*`level`*`,BPE)`.

The system services trace table traces IMS Connect BPE system service calls. The default number of pages for this table is 2.

**STG**
Specify as `TRCLEV=(STG,`*`level`*`,BPE)`.

The storage service trace table traces storage service requests. The default number of pages for this table is 4.

**USRX**
Specify as `TRCLEV=(USRX,`*`level`*`,BPE)`.

The user exit routine trace table traces activity related to exit routines (for example, loads, calls, or abends). The default number of pages for this table is 2.

You can code the following values for IMS Connect-defined trace tables:

***** Specify as `TRCLEV=(*,`*`level`*`,HWS)`.

Specifying a type of ***** enables you to set the default trace level (and optionally, the default number of pages per trace table) for **all** IMS Connect-defined trace table types. If you use the ***** type, make sure it is the first TRCLEV statement for IMS Connect-defined trace table types in your PROCLIB member. You can then code additional TRCLEV statements for specific IMS Connect types to selectively override the defaults.

| **Recommendation:** Code a TRCLEV statement with a type of ***** for IMS Connect traces, specifying a level of HIGH as your first TRCLEV statement for IMS Connect-defined trace table types. Using this coding ensures that at least some tracing is done for all IMS Connect trace tables. It also ensures that any new trace table types that are added in the future will be turned on in your system, even if you have not modified your IMS Connect BPE configuration parameter PROCLIB member to explicitly add a TRCLEV statement.

**CMDT**
Specify as `TRCLEV=(CMDT,`*`level`*`,HWS)`.

The command trace table traces IMS Connect command activity. The default number of pages for this table is 2.

**ENVT**
Specify as `TRCLEV=(ENVT,`*`level`*`,HWS)`.

The interface trace table traces activity in the interface between an IMS Connect and its client. The default number of pages for this table is 2.

**HWSI**
Specify as `TRCLEV=(HWSI,`*`level`*`,HWS)`.

The IMS Connect to OTMA driver trace table traces communication activity between IMS Connect and OTMA drivers. The default number of pages for this table is 2.

**HWSN**
Specifies as `TRCLEV=(HWSN,`*`level`*`,HWS)`

The IMS Connect to local option driver trace table traces communication activity and event between local option driver and IMS Connect. The default number of pages for this table is 2.

**HWSW**
Specify as `TRCLEV=(HWSW,`*`level`*`,HWS)`.

The IMS Connect to TCP/IP driver trace table traces communication activity and events between TCP/IP drivers and IMS Connect. The default number of pages for this table is 2.

**OTMA**
Specify as `TRCLEV=(OTMA,`*`level`*`,HWS)`.

The OTMA communication driver trace table traces internal communication protocol activity (XCF calls). The default number of pages for this table is 2.

**PCDR**
Specifies as `TRCLEV=(PCDR,`*`level`*`,HWS)`

The local option driver trace table traces local option communication protocol activity. The default number of pages for this table is 2.

**TCPI**
Specify as `TRCLEV=(TCPI,`*`level`*`,HWS)`.

The TCP/IP communication driver trace table traces communication protocol activity (TCP/IP calls). The default number of pages for this table is 2.

**OMDR**
Specify as `TRCLEV=(OMDR,`*`level`*`,HWS)`.

The IMSplex communication driver trace table traces communication protocol activity (SCI calls). The default number of pages for this table is 2.

**HWSO**
Specify as `TRCLEV=(HWSO,`*`level`*`,HWS)`.

The IMSplex driver trace table traces communication activity and events between the IMSplex driver and IMS Connect. The default number of pages for this table is 2.

**RRSI**
Specify as `TRCLEV=(RRSI,`*`level`*`,HWS)`.

The Two Phase Commit trace table traces communication activity and events between IMS Connect and RRS. The default number of pages for this table is 2.

*component*
Specifies the IMS Connect component that defines the trace table type. Possible values are:

**BPE**
Indicates that the table is an IMS Connect BPE-defined (system) trace table type.

**HWS**
Indicates that the table is an IMS Connect-defined trace table type.

*level*
Controls how much tracing is done in a specified trace table. Each trace entry that is made has a level associated with the entry. Each trace level has a level setting that is controlled by the *level* value which you specify on the TRCLEV statement.

| IMS Connect only specifies HIGH on its trace entries, so you must set the level
| *value* to HIGH to receive any traces.

A high setting of the *level* parameter results in more trace entries written to the table. More trace entries can provide additional diagnostic information for solving a problem; however, the trace table tends to wrap more frequently, and higher settings can cause additional CPU usage. Also, trace information is not as detailed at higher settings so the captured information may not be sufficient to solve a problem.

Choose one of the following values for the *level* parameter:

**NONE**
No tracing.

**Note:** Do not specify NONE because no tracing, not even tracing for error conditions, is done for the specified table. The *level* can be changed from NONE to one of the other values by issuing the BPE command UPD.

**ERROR**
Only trace entries for error conditions are made. ERROR is the default.

**HIGH**
High-volume tracing (all component events).

***ims_component***
Specifies the IMS component that defines the trace table.

**PAGES=***num_pages*
An optional parameter that can be used to specify the number of 4 KB pages to be allocated for the trace table type.

The maximum number of pages for any trace table is 32767. If you specify a number greater than this, IMS Connect BPE uses 32767 as the value for the PAGES= parameter. If IMS Connect BPE is unable to get the amount of storage you requested for a trace table, it will try to get a smaller number of pages to enable some tracing to still be done. You can see the actual number of pages BPE obtained for each trace table by issuing the DISPLAY TRACETABLE command.

If you do not use this parameter, then the trace table has the default number of pages, as specified under the description each trace table type.

***Sample IMS Connect BPE Configuration File:*** A sample IMS Connect BPE configuration data set is shown in Figure 5.

```
*********************************************************************
* CONFIGURATION FILE FOR IMS CONNECT BPE                            *
*********************************************************************

LANG=ENU                             /* LANGUAGE FOR MESSAGES     */
                                     /* (ENU = U.S. ENGLISH)      */

#
# DEFINITIONS FOR IMS CONNECT BPE SYSTEM TRACES
#

TRCLEV=(*,LOW,BPE)                   /* DEFAULT TRACES TO LOW     */
TRCLEV=(AWE,HIGH,BPE)                /* AWE SERVER TRACE ON HIGH  */
TRCLEV=(CBS,MEDIUM,BPE)              /* CTRL BLK SRVCS TRC ON MED */
TRCLEV=(DISP,HIGH,BPE,PAGES=12)      /* DISPATCHER TRACE ON HIGH  */
                                     /* WITH 12 PAGES (48K BYTES) */
#
# DEFINITIONS FOR IMS CONNECT TRACES
#

TRCLEV=(*,HIGH,HWS)             /* DEFAULT ALL IMS CONNECT TRACES TO HIGH        */
TRCLEV=(HWSI,HIGH,HWS)        /* BUT RUN IMS CONNECT TO IMS OTMA TRACE...      */
TRCLEV=(HWSW,HIGH,HWS)      /* AND SERVER TO IMS CONNECT TRACE AT MEDIUM  */
```

*Figure 5. Example of a Configuration File for IMS Connect BPE*

## Formatting Incore Trace Tables

IMS Connect trace tables are “incore” tables, which can be formatted from a dump of an IMS Connect address space by using the IMS Connect dump formatter.

The traces are formatted by the standard IMS Connect BPE formatting services. You must link-edit HWSFTRC0 with an alias of HWSFTvrm (where v is the version level, r is the release level, and m is the modification level); for example, IMS Connect 2.2.0 would have the alias of HWSFT220. This link-edit is provided in HWSJCLIN and is link-edited into your own defined RESLIB. Either the IMS Connect RESLIB must be included in the IMS Connect BPE procedure for formatting the dump or you will have to link-edit the IMS Connect trace formatter module (HWSFTRC0) into the IMS Connect BPE procedure RESLIB (IMS RESLIB). Following is the include, alias, and name statement:

```
INCLUDE LOAD(HWSFTRC0)          HWS FORMATTED TRACE
ALIAS HWSFT110                  HWS 1.1.0 FORMATTED TRACE ALIAS
NAME HWSFTRC0(R)
```

The link-edit step must use the parms as defined in HWSJCLIN for step 1 as defined in the following example for IMS Connect:

```
//       PARM=('SIZE=880K,64K)',RENT,REFR,
//       NCAL,LET,XREF,LIST)
```

## Setting IMS Connect Allocations

Refer to the *Program Directory for IBM IMS Connect for OS/390* for the recommended allocations for the required IMS Connect libraries.

To allocate the HWSRCDR data set from TSO, use these settings:

```
                                  Data Set Information
Command ===>

Data Set Name . . . .: HWSRCDR

General Data                         Current Allocation
 Volume serial. . . .: USER01         Allocated cylinders: 1
 Device type  . . . .: 3390           Allocated extents. : 1
 Organization . . . .: PS
 Record format. . . .: FB
 Record length. . . .: 1440
 Block size . . . . .: 14400         Current Utilization
 1st extent cylinders: 1              Used cylinders . . : 1
 Secondary cylinders : 5              Used extents . . . : 1

 Creation date  . . .: 1998/10/01
 Expiration date  . .: ***None***
```

## Invoking IMS Connect

You invoke IMS Connect using either an MVS procedure or an MVS job. If you start multiple instances of IMS Connect with the same configuration, a connection outage can occur.

**Recommendation:** To avoid starting the same IMS Connect address space more than once, start the IMS Connect by running an MVS job with a unique MVS initiator class assigned to it, rather than starting the connection as a procedure. The following is an example of such a job.

```
//HWS01    JOB   MSGLEVEL=1,TIME=1440,CLASS=Y,USERID=&USERID
//*********************************************************************
//*  BRING UP IMS CONNECT USING A JOB                                  *
//*********************************************************************
//HWS01    EXEC HWS,SOUT=A
```

The following example shows the JCL statements required to define the MVS environment for IMS Connect.

```
//HWS       PROC  RGN=4096K,SOUT=A,
//             BPECFG=BPECFGHT,
//             HWSCFG=HWSCFG00
//*
//*********************************************************************
//*  BRING UP AN IMS CONNECT                                           *
//*********************************************************************
//STEP1    EXEC PGM=HWSHWS00,REGION=&RGN,TIME=1440,
//             PARM='BPECFG=&BPECFG,HWSCFG=&HWSCFG'
//STEPLIB  DD   DSN=SHWSRESL,DISP=SHR
//         DD   DSN=SDFSRESL,DISP=SHR
//         DD   DSN=CEE.SCEERUN,UNIT=SYSDA,DISP=SHR
//         DD   DSN=SYS1.CSSLIB,UNIT=SYSDA,DISP=SHR
//         DD   DSN=GSK.SGSKLOAD,UNIT=SYSDA,DISP=SHR
//PROCLIB  DD   DSN=USER.PROCLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
//HWSRCORD DD   DSN=HWSRCDR,DISP=SHR
```

**Note:** The SDFSRESL library, which is the IMS Version 8.1 or later execution library, is required by IMS Connect when IMSplex support is used. If IMSplex support is not used, then the SDFSRESL library is not required. IMS Connect requires the CEE.SCEERUN, SYS1.CSSLIB, and GSK.SGSKLOAD libraries (which are the C execution and z/OS system SSL libraries) only when SSL support is used.

Use the following parameters to define the values in the JCL:

**RGN=** Specifies the size of the MVS address space to be allocated for the IMS Connect control program (HWSHWS00).

**SOUT=**
Specifies the class assigned to SYSOUT DD statements.

**BPECFG=**
Specifies the name of a member in the PROCLIB data set that contains the IMS Connect BPE specifications.

**HWSCFG=**
Specifies the name of a member in the PROCLIB data set that contains the IMS Connect configuration information.

## Customizing IMS Connect

You can customize various aspects of IMS Connect to fit your specific business needs by modifying any of the user message exits that IMS Connect provides. These exits are described in the following table.

*Table 2. IMS Connect Exits and Descriptions*

| Exit Name | Type | Associated Macro Files | Purpose and Description |
|---|---|---|---|
| HWSCSLO0[b] HWSCSLO1[b] | User message exits | N/A | User message exits that are required to support the IMS Control Center and are used in conjunction with the Operations Manager support. If you want to connect to the IMS Control Center, these exits must be specified on the EXIT= parameter of the TCP/IP statement in the IMS Connect configuration file. |
| HWSIMSO0[ad] | User message exit | N/A | Replaces the EZAIMSO0 exit previously provided by TCP/IP. |
| HWSIMSO1[ad] | User message exit | N/A | Replaces the HWSIMS00 exit. Passes a fullword length field preceding the message. |
| HWSJAVA0 [c] | User message exit for IMS Connector for Java clients only | HWSIMSCB HWSIMSEA HWSEXPRM HWSOMPFX | Enables IMS Connector for Java users to edit messages and perform security checking. |

*Table 2. IMS Connect Exits and Descriptions (continued)*

| Exit Name | Type | Associated Macro Files | Purpose and Description |
|---|---|---|---|
| HWSSMPL0[ae] | User message exit for non-IMS Connector for Java clients only | HWSIMSCB HWSIMSEA HWSEXPRM HWSOMPFX | Enables you to use your own message formats. HWSSMPL0 returns the MOD name to the client for message formatting if the IMS transaction supplies the name. You can also use your own formats to pass the client's authentication and have this, or another user exit, verify client authentication. |
| HWSSMPL1[a] | User message exit for non-IMS Connector for Java clients only | HWSIMSCB HWSIMSEA HWSEXPRM HWSOMPFX | Enables you to use your own message formats. HWSSMPL1 returns the MOD name to the client for message formatting if the IMS transaction supplies the name. HWSSMPL1 also passes a fullword length field preceding the message. You can also use your own formats to pass the client's authentication and have this, or another user exit, verify client authentication. |
| HWSTECL0 | User message exit | See Appendix G, "HWSTECL0 User Exit," on page 267 for a list of associated macro files. | Enables you to customize IMS Connect to support event recording. Stores all trace and event notifications through a recording routine to be used by any event recording function. For more information about customizing this message exit, see Appendix G, "HWSTECL0 User Exit," on page 267. |
| HWSYDRU0 [c] | Sample OTMA DRU exit | N/A | A DRU exit is required to support IMS Connect's asynchronous output features. See the *IMS Open Transaction Manager Access Guide* for information about writing a DRU exit. |
| HWSUINIT [c] | User initialization exit | HWSXIB HWSXIBDS | Enables you to perform your own processing during IMS Connect initialization and termination. The user message exits receive control during each incoming and outgoing message, but HWSUINIT receives control only at initialization and termination time. |

*Table 2. IMS Connect Exits and Descriptions (continued)*

| Exit Name | Type | Associated Macro Files | Purpose and Description |
|---|---|---|---|

[a]These exits must be specified on the EXIT= parameter of the TCP/IP statement in the IMS Connect configuration file. If you use your own user message exit, your exit must also be specified on the EXIT= parameter of the TCP/IP statement in the IMS Connect configuration file.

[b] If you want to connect to the IMS Control Center, these exits must be specified on the EXIT= parameter of the TCP/IP statement in the IMS Connect configuration file.

[c] Do not define these exits in the IMS Connect configuration file, specifically on the EXIT= parameter of the TCP/IP statement. HWSJAVA0 is dynamically loaded by IMS Connect; HWSYDRU0 is loaded by OTMA during IMS Connect initialization; HWSUINIT is dynamically loaded by IMS Connect during IMS Connect initialization.

[d]IMS Connect Version 9 is the last release to support HWSIMSO0 and HWSIMS01. These two user message exits will not be available with any future release of IMS Connect. It is recommended that you migrate to HWSSMPL1.

[e]It is recommended that you migrate from HWSSMPL0 to HWSSMPL1 because new function will no longer be added to HWSSMPL0.

The HWSIMSO0, HWSIMS01, HWSCSLO0, and HWSCSLO1 user message exits, as well as the IMS Connect components, are installed on your system. To customize any of the other exits, modify the exit and then install it into your IMS Connect resource library (SHWSRESL).

**Requirement:** You must install the following two exits into your IMS Connect resource library, regardless of whether you intend to customize them, because IMS Connect automatically loads these exits when it executes:

- HWSJAVA0
- HWSUINIT

You must compile and link-edit these exits before you execute IMS Connect, or else IMS Connect will not run. If you do not need to customize either of these two exits, you do not need to do anything else with them.

**Related Reading:**

- "Installing HWSJAVA0, HWSUINIT, HWSYDRU0, HWSSMPL0, and HWSSMPL1" on page 32 provides information and instructions on installing the IMS Connect exits into the IMS Connect resource library.
- "Modifying HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0" on page 33 and "Modifying HWSIMSO0 and HWSIMSO1" on page 35 describe how to modify the IMS Connect exits.
- Appendix D, "IMS Connect JCL," on page 259 provides sample JCL examples to assist you when link-editing and compiling the HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0 exits.
- "Installing HWSCSLO0 and HWSCSLO1" on page 36 provides information on installing the IMS Connect exit.
- Appendix G, "HWSTECL0 User Exit," on page 267 provides information on customizing the IMS Connect exit.

## Installing HWSJAVA0, HWSUINIT, HWSYDRU0, HWSSMPL0, and HWSSMPL1

The exits HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0 are installed into AHWSSRC (the source library) during the IMS Connect installation process. HWSSMPL0, HWSSMPL1, HWSJAVA0, HWSUINIT, and HWSYDRU0 are **not** placed in the load library, and they are **not** link-edited into the IMS Connect resource library (SHWSRESL) during the installation process. This is to ensure that subsequent IMS Connect installations and SMP/E maintenance do not destroy your copies of these exits in either the load library or in your IMS Connect resource library.

**Requirement:** You must install the following two exits into your IMS Connect resource library (SHWSRESL), regardless of whether you intend to use them, because IMS Connect automatically loads them:

- HWSJAVA0
- HWSUINIT

If these two exits are not present, IMS Connect will not run. If you do not need to customize either of these two exits, you do not need to do anything else with them.

You need to install the HWSSMPL0, HWSSMPL1, and HWSDRU0 exits only if you want to use the features that they support. HWSSMPL0 will no longer be enhanced after the IMS Connect Version 9 release. It is recommended that you migrate from HWSSMPL0 to HWSSMPL1 to obtain future function support.

**Requirement:** Compile and link-edit the HWSYDRU0 exit into your IMS resource library (SDFSRESL), *not* the IMS Connect resource library (SHWSRESL). Otherwise, OTMA will not be able to use the HWSYDRU0 exit.

All five of these exits can be used as shipped or can be modified (customized). See “Modifying HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0” on page 33 for more information about modifying these five exits.

To install an exit into the IMS Connect resource library, you compile and link-edit the exit into the IMS Connect resource library (in other words, into SHWSRESL). The required macros for each exit are shipped in the SHWSMAC library.

The following table describes the link-editing requirements for installing each of the four exits.

*Table 3. Link-Editing Requirements*

| Exit Name | Installation Required? | Link-editing Requirements |
|---|---|---|
| HWSJAVA0 | Yes | Link-edit this exit using its given name. |
| HWSUINIT | Yes | Link-edit this exit using its given name. |

Table 3. Link-Editing Requirements (continued)

| Exit Name | Installation Required? | Link-editing Requirements |
|---|---|---|
| HWSYDRU0 | No | You can link-edit this exit using its given name or you can supply your own name. You specify the name used for the exit on the `DATASTORE` statement for `DRU=` in the IMS Connect configuration file.<br>**Note:** This exit is not required unless you plan to support asynchronous output with IMS Connect. If so, an OTMA DRU exit (either HWSYDRU0 or DFSYDRU0 or your own DRU exit) must exist in your IMS system. See *IMS Version 9: Open Transaction Manager Access Guide and Reference* for more information. |
| HWSSMPL0 | No | You can link-edit this exit using its given name or you can supply your own name. You specify the name used for the exit on the `TCPIP` statement for `EXIT=` in the IMS Connect configuration file. **Important:** This exit is shipped with all asynchronous output support options as able to be activated based on the IRM input. If you want to support only the ″noauto″ asynchronous output message management function, then you do not need to modify this exit. |
| HWSSMPL1 | No | You can either link-edit this exit using its given name, or supply your own name. Specify the name used for the exit on the TCPIP statement for EXIT= in the IMS Connect configuration file. **Important:** HWSSMPL1 is shipped with all asynchronous output support options capable of activation based on the IRM input. If you want to support only the “noauto” asynchronous output message management function, do not modify this exit. |

## Modifying HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0

**Recommendation:** Before you modify an exit, make a copy of that exit and rename the copy. Having this copy will enable you to control the modifications to the exit.

The following instructions describe how to customize, modify and re-install the HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0 exits.

1. Make your changes to the source code provided in the AHWSSRC source library (see the *Program Directory for IBM IMS Connect for OS/390* for more information about the AHWSSRC source library).
2. Assemble the exit. The exit and its associated macro files are members of the partitioned data set into which you receive the AHWSSRC data set, as described in the IMS Connect installation procedures in the *Program Directory for IBM IMS Connect for OS/390*. See Table 2 on page 29 for a list of the macro files that are associated with each exit.
3. Link-edit the output from the assembled job to create a load module named HWSXXXXX. XXXXX stands for the name of the exit that you are link-editing.

   **Requirements**:

   - Link-edit the HWSJAVA0, HWSUINIT, HWSSMPL0, and HWSSMPL1 exits into the IMS Connect resource library SHWSRESL.

- Link-edit the HWSYDRU0 exit into the *IMS* resource library SDFSREL. Doing this enables OTMA to use the exit.

4. Link-edit the resulting load module into the appropriate resource library.
   - If you are customizing HWSJAVA0, HWSUINIT, HWSSMPL0, or HWSSMPL1, link-edit the load module into your IMS Connect resource library (SHWSRESL). IMS Connect loads the module from its resource library during initialization.
   - If you are customizing HWSYDRU0, link-edit the load module into the IMS resource library SDFSRESL. IMS loads the exit HWSYDRU0 when IMS Connect initializes.
5. ***For HWSSMPL0 and HWSSMPL1***: You also need to modify the IMS Connect configuration file so that it includes the HWSSMPL0 and/or HWSSMPL1 user exits in the TCPIP statement, as follows:
   `TCPIP=(...,EXIT=(HWSSMPL0,HWSSMPL1),..)`. Then restart IMS Connect.

**Related Reading**:
- The user message exit structures are described in “User Exit Message Description and Structures” on page 69.
- HWSYDRU0, the sample destination resolution (DRU) exit, is described in Chapter 6, “IMS Connect DRU Exit for Asynchronous Output Support,” on page 85.
- HWSUINIT, the sample user initialization exit, is described in Chapter 7, “IMS Connect User Initialization Exit Support,” on page 87.
- Appendix D, “IMS Connect JCL,” on page 259 provides sample JCL examples to help you link-edit and compile the HWSJAVA0, HWSUINIT, HWSSMPL0, HWSSMPL1, and HWSYDRU0 exits.

## Modifying User Message Exits to Provide Trusted User Support

Trusted user support allows you to define and identify a client input message as a trusted IMS Connect user so that the IMS Connect RACF call to IMS can be bypassed. IMS is not involved in identifying trusted users.

To define a client input message as a trusted user, select a field in the IRM header that will identify the trusted user to a user message exit. For example, you can specify one of the PORTID, CLIENTID, USERID, TRANSACTION CODE fields, or user data to identify a trusted user. The user message exit evaluates the client input message based on the trusted user identifier and determines whether or not the input message is from a trusted user. If the input is from a trusted user, the user message exit requests IMS Connect to bypass the IMS Connect RACF call to IMS if RACF=Y is specified for IMS Connect.

To provide trusted user support, you must define and provide most of the logic in the client code and in the user message exits (HWSSMPL0 and HWSSMPL1) or in your own user message exit.

Sample logic (which is commented out) is provided in both HWSSMPL0 and HWSSMPL1 and can be found by looking for the following comment lines:

```
************************************************
**************TRUSTED USER SUPPORT**************
************************************************
```

You must define one or more fields (user-defined length) in the IRM user portion to be set by the client code and analyzed by the user message exit. For example, you may decide to add three one-byte fields in the IRM and set different values in each field. When the message is passed to the user message exit, the exit interrogates those three fields to determine if the connection should be treated as a trusted user.

If the connection is treated as a trusted user, the exit will return to IMS Connect in the OTMA User Data Header, field OMUSR_FLAG2 set to OMUSR_TRSTUSR, to signal to IMS Connect to bypass issuing the RACF authentication call. You may also base the trusted user on other values in the IRM, such as ClientID, UserID, Port number, IP address, or any other data that you wish to use.

IMS Connect does not define which flag bytes in the IRM to set or what settings to use. You must define the IRM bytes and byte settings so that the definitions are unique to your system.

Trusted user is only supported through user-written user message exits and through the IMS Connect-supplied user message exits HWSSMPL0 and HWSSMPL1. Because HWSIMSO0 and HWSIMSO1 are not shipped as source code, they do not support Trusted user. If you are currently using HWSIMSO0, HWSIMSO1, or both and want to use trusted user support, you must change and use HWSSMPL0, HWSSMPL1, or both, and provide the changes described above.

The user message exit, HWSCSLO0, and the IMS Control Center do not support the trusted user function. IMS Connector for Java also does not support the trusted user function. However, you can modify the HWSJAVA0 user message exit and use other criteria to determine if the client is a trusted user.

You can modify the HWSJAVA0 exit based on the existing data in the OTMA headers such as OMUSR_DESTID (DataStore), OMUSR_ORIGIN (ClientID), OMUSR_PORTID (PortId), OMUSR_PASSTICKET (Password), or other message values to determine if this input is from a trusted user client that does not need authentication. If the HWSJAVA0 exit determines that the message is from a trusted user client, you can set OMUSR_FLAG2 to OMUSR_TRSTUSR. By changing the OMUSR_FLAG2 value to OMUSR_TRSTUSR, IMS Connect bypasses the RACF call if RACF=Y was specified in the IMS Connect configuration file (HWSCFG nn) or if the SETRACF ON command was issued.

## Modifying HWSSMPL0 and HWSSMPL1 for PassTicket

When you implement PassTicket support and send an APPLname in the IRM field, (IRM_APPL_NM) you must ensure that the logic in the user message exit is modified to check for a new minimum length in the llll field. The llll field must be modified from a minimum length of 88 to minimum length of 96. The IRM length must also be modified from 80 to 88. Another option is to comment out the length comparison in the user message exit.

## Modifying HWSIMSO0 and HWSIMSO1

The HWSIMSO0 and HWSIMSO1 user message exits replace the EZAIMSO0 exit that was previously provided by TCP/IP. You must use one of these exits to replace the TCP/IP EZAIMSO0 exit. You can, however, either customize the linkedit of either of these exits to include a security exit (IMSLSECX), or provide an installation-specific name instead of HWSIMSO0 or HWSIMSO1 in the linkedit step.

IMS Connect Version 9 is the final release of these two user message exits. HWSIMSO0 and HWSIMSO1 will not be available in any future release of IMS Connect. It is recommended that you migrate to HWSSMPL1 to support future enhancements to IMS Connect.

**Requirements**:

- If you want to use your own security checking routine, it must be called IMSLSECX.
- The JCL that is provided in HWSJCLIN link-edits HWSIMSO0 with the name *HWSIMSO0* and HWSIMSO1 with the name *HWSIMSO1*. However, you can change the name to be any name you want to use. If you change the *HWSIMSO0* or *HWSIMSO1* name in the JCL to a different name, you must make the same name change in the IMS Connect configuration file. The *HWSIMSO0* and *HWSIMSO1* names are used in the `EXIT` parameter on the `TCPIP` statement in the IMS Connect configuration file. See "IMS Connect Configuration Statement Parameters" on page 13 for more information about modifying the IMS Connect configuration file.

The HWSIMSO0 and HWSIMSO1 user message exits are installed as part of HWSxxxxx module installation, using step 6 of HWSJCLIN. (HWSJCLIN is a sample link-edit job contained in SHWSMAC.)

The following lines are the INCLUDE statements for HWSIMSO0 and HWSIMSO1 from step 6 of HWSJCLIN:

```
//SYSLIN DD *
INCLUDE LOAD(HWSIMSO0)
ENTRY HWSIMSO0
MODE RMODE(24),AMODE(31)
NAME HWSIMSO0(R)

INCLUDE LOAD(HWSIMSO1)
ENTRY HWSIMSO1
MODE RMODE(24),AMODE(31)
NAME HWSIMSO1(R)
```

**Note:** The dependency on the TCP/IP library to detect the inclusion of the TCP/IP translate tables is no longer required.

If you want to use security checking (either the IMSLSECX file that comes with TCP/IP or your own IMSLSECX file), you must add the following INCLUDE statement.

```
  INCLUDE USERLOAD(IMSLSECX)
```

HWSIMSO0 and HWSIMSO1 do not support the trusted user function.

## Installing HWSCSLO0 and HWSCSLO1

The HWSCSLO0 and HWSCSLO1 user message exits are provided as OCO code and is present in the Load library that is delivered with IMS Connect. The HWSCSLO0 and HWSCSLO1 user message exits are only valid with support of the IMS Control Center support.

**Requirements**:

- HWSCSLO0 and HWSCSLO1 exit names must be added to the TCPIP statement EXIT= parameter if the IMS Control Center support is made available. If the IMS Control Center is not supported in your installation, do not add this exit to the TCPIP statement EXIT= parameter.
- HWSCSLO0 and HWSCSLO1 are included in the install process of IMS Connect.

## JCL to Print IMS Connect RECORDER Output

IMS Connect provides a line trace function to capture data that is received from and sent to a client. The line trace contains a copy of the first 670 bytes of the data as it is passed to the user message exit and upon return from the user message exit. Line traces are intended for use in problem resolution.

Use the RECORDER command to activate and terminate the line trace function. The following sample JCL illustrates how to print the line trace data set:

```
//IDCAMS   JOB  JOB 1,IDCAMS,MSGLEVEL=1,CLASS=K,TIME=1440
//SELECT   EXEC PGM=IDCAMS
//DD1      DD   DSNAME=HWSRCDR,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  PRINT    INFILE(DD1)
```

# Chapter 5. IMS Connect User Message Exit Support

IMS Connect communicates with clients through TCP/IP sessions (including SSL sessions) and Program Call interface for local support using the IRM message header that is defined in the HWSIMSCB macro. IMS Connect communicates with OTMA through an XCF session using the OTMA message headers. Clients that use TCP/IP socket calls as their communication vehicle can design a user message exit routine that runs with IMS Connect to convert messages between formats as follows:

- Convert the client message format to OTMA message format.
- Convert the IMS response, in OTMA message format, to client message format.

These conversions enable the client to retrieve IMS data through a TCP/IP connection. IMS Connect automatically sends and receives messages when they are formatted correctly.

**Related Reading**:

- Chapter 6, "IMS Connect DRU Exit for Asynchronous Output Support," on page 85 describes HWSYDRU0, which is a sample OTMA Destination Resolution (DRU) exit that can be used to support asynchronous output that is generated by an IMS application.
- Chapter 7, "IMS Connect User Initialization Exit Support," on page 87 describes HWSUINIT, which is a user initialization exit routine that receives control at IMS Connect initialization and termination time.

**Attention**: Do not issue any MVS calls in the user message exit that result in an `MVS WAIT`. If you modify the user message exit and add code that results in an `MVS WAIT`, all work on the TCP/IP PORT will halt until the WAIT has been posted. The user message exits cannot be modified to free any storage passed to the exit, and IMS Connect will not free any storage obtained by the user message exit when the exit returns to IMS Connect. All storage obtained by IMS Connect must be released by IMS Connect and cannot be freed by the User Message Exit without causing failures.

**In this chapter**:

## How IMS Connect Communicates with a TCP/IP Client

IMS Connect expects all client messages that it receives to start with a four byte total length field, followed by a common 28 (decimal) byte message IRM prefix. At your option, you can add data following the common 28 (decimal) byte IRM prefix if you are providing your own user message exit or are modifying HWSSMPL0 or HWSSMPL1. Because HWSIMSO0 and HWSIMSO1 are not provided as source, you cannot change the IRM for these exits. The IRM also cannot be modified for HWSJAVA0 user message exit.

If you add or delete items that follow the common IRM section (first 28 bytes), you must also adjust the user message exits that you use, or provide your own user message exits.

**Note:** However, the user message exits, HWSCSLO0 and HWSCSLO1, do not have any message definitions because HWSCSLO0 and HWSCSLO1 cannot be modified or replaced.

Table 4 shows the fixed format preceding the input message sent to IMS Connect from clients. It includes the message field, the field length, and a brief explanation of the message.

*Table 4. Fixed Format*

| **Field** | **Length** | **Meaning** |
|---|---|---|
| Message field. | | |
| llll | 4 bytes | Length of the total message. The total message length includes the length of the IRM (variable, depending on your requirements), the llll field (four bytes), the length of the message, and the end of message indicator X'0004000' (four bytes). The minimum value is X'58'. The maximum value is X'00989680' (10,000,000 bytes) for IMS Connector for Java and any positive value for non-IMS Connector for Java clients. This field is read as a binary number. |
| The following fields are for the 28 (decimal) byte IRM prefix. | | |
| IRM_LEN | 2 bytes | Length of the IRM structure. The minimum size of the IRM for user written exits is X'24' or binary ’00100100’. HWSIMSO0 and HWSSMPL0 have a minimum IRM length of X'50' or binary ’01010000’. |
| IRM_ZZ | 2 bytes | Reserved. Initialize to binary zeros. |
| IRM_ID | 8 bytes | Character string. Specifies the identifier of the user exit that is to be driven after the complete message has been received. For details, see “How IMS Connect Communicates with User Message Exits” on page 60. The IMS Connect-supplied user message exits reserve and use these IDs: • `*IRMREQ*`-- for HWSIMSO0 • `*IRMRE1*`-- for HWSIMSO1 • `*SAMPLE*`-- for HWSSMPL0 • `*SAMPL1*`-- for HWSSMPL1 • `*HWSJAV*`-- for HWSJAVA0 • `*HWSCSL*`-- for HWSCSLO0 |
| Reserved | 4 bytes | Reserved for future use. Initialize to binary zeros. |

*Table 4. Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_F5 | 1 byte | Input message type.<br>**X'80'** OTMA headers built by client.<br>**X'40'** Translation done by client.<br>**X'00'** No option flow of messages (see meaning for X'04'). This is the default if no value is specified.<br>**X'01'** Single message. Only one message returned on receive following resume tpipe.<br>**X'02'** Auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a large value. Use only if the client is a dedicated output client.<br>**X'04'** No auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a small value. Use only if the client is a dedicated output client. This value is similar to Auto, except that the IRM_TIMER will cause the last receive to terminate. |
| IRM_TIMER | 1 byte | Time delay that IMS Connect will wait for IMS to return data to IMS Connect which, in turn, will be sent to the client. The following functions support the IRM_TIMER settings:<br>• TCP/IP SEND of a RESUME TPIPE<br>• TCP/IP SEND of an ACK or NAK<br>• TCP/IP SEND of data<br>• PC SEND of a RESUME TPIPE<br>• PC SEND of an ACK or NAK<br>• PC SEND of data<br>**Note:** See “IRM_TIMER Usage” on page 52. |

*Table 4. Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_SOCT | 1 byte | Socket connection type. The client can set this value as follows: • X'00' - transaction socket. The socket connection lasts across a single transaction. • X'10' - persistent socket. The socket connection lasts across multiple transactions. • X'40' - non-persistent socket. The socket connection lasts for a single exchange consisting of one input and one output. **Recommendation**: Do not use this socket type if you plan on implementing conversational transactions, because multiple connects and disconnects will occur. |
| IRM_ES | 1 byte | Unicode encoding schema. Initialize to binary zeros. • X'01' UTF8 encoding schema. • X'02' UCS2 encoding schema. • X'02' UTF16 encoding schema. |
| IRM_CLIENTID | 8 bytes | A string of 1 to 8 uppercase alphanumeric (A through Z, 0 to 9) or special (@, #, $) characters, left justified, and padded with blanks. IRM_CLIENTID specifies the name of the client ID that is used by IMS Connect. If this string is not supplied by the client, then the user exit must generate it. The client ID is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_CLID. |

The llll field tells IMS Connect how long the message is, and the IRM provides additional information, such as the specific user exit to which the data is to be passed.

IMS Connect supports client applications written for IMS TOC 2.1.3 without any application modifications to the current message structure.

The base structure for non-IMS Connector for Java clients is shown in Table 5. The base structure contains the four byte llll total message length field, the 28-byte message IRM prefix, and the user-defined structure.

*Table 5. Base Structure for Non-IMS Connector for Java Clients*

| Field | Length | Meaning |
|---|---|---|
| Message field. | | |
| llll | 4 bytes | Length of the total message. The total message length includes the length of the IRM (variable, depending on your requirements), the llll field (four bytes), the length of the message, and the end of message indicator X'0004000' (four bytes). The value must be between X'58' and X'7FFFFFFF'. This field is read as a binary number. |
| The following fields are for the 28-byte IRM prefix. | | |

*Table 5. Base Structure for Non-IMS Connector for Java Clients (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_LEN | 2 bytes | Length of the IRM structure. The minimum size of the IRM for user written exits is X'24' or binary '00100100'. HWSIMSO0 and HWSSMPL0 have a minimum IRM length of X'50' or binary '01010000'. |
| IRM_ZZ | 2 bytes | Reserved. Initialize to binary zeros. |
| IRM_ID | 8 bytes | Character string. Specifies the identifier of the user exit that is to be driven after the complete message has been received. For details, see "How IMS Connect Communicates with User Message Exits" on page 60. The IMS Connect-supplied user message exits reserve and use these IDs: <br>• `*IRMREQ*`-- for HWSIMSO0 <br>• `*IRMRE1*`-- for HWSIMSO1 <br>• `*SAMPLE*`-- for HWSSMPL0 <br>• `*SAMPL1*`-- for HWSSMPL1 <br>• `*HWSJAV*`-- for HWSJAVA0 <br>• `*HWSCSL*`-- for HWSCSLO0 |
| Reserved | 4 bytes | Reserved for future use. Initialize to binary zeros. |
| IRM_F5 | 1 byte | Input message type. <br>**X'80'** OTMA headers built by client. <br>**X'40'** Translation done by client. <br>**X'00'** No option flow of messages (see meaning for X'04'). This is the default if no value is specified. <br>**X'01'** Single message. Only one message returned on receive following resume tpipe. <br>**X'02'** Auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a large value. Use only if the client is a dedicated output client. <br>**X'04'** No auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a small value. Use only if the client is a dedicated output client. This value is similar to Auto, except that the IRM_TIMER will cause the last receive to terminate. |

*Table 5. Base Structure for Non-IMS Connector for Java Clients (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_TIMER | 1 byte | Time delay that IMS Connect will wait for IMS to return data to IMS Connect which, in turn, will be sent to the client. The following functions support the IRM_TIMER settings:<br>• TCP/IP SEND of a RESUME TPIPE<br>• TCP/IP SEND of an ACK or NAK<br>• TCP/IP SEND of data<br>• PC SEND of a RESUME TPIPE<br>• PC SEND of an ACK or NAK<br>• PC SEND of data<br><br>**Note:** See “IRM_TIMER Usage” on page 52. |
| IRM_SOCT | 1 byte | Socket connection type. The client can set this value as follows:<br>• X'00' - transaction socket. The socket connection lasts across a single transaction.<br>• X'10' - persistent socket. The socket connection lasts across multiple transactions.<br>• X'40' - non-persistent socket. The socket connection lasts for a single exchange consisting of one input and one output. **Recommendation**: Do not use this socket type if you plan on implementing conversational transactions, because multiple connects and disconnects will occur. |
| IRM_ES | 1 byte | Unicode encoding schema:<br>• X'01' UTF8<br>• X'02' UCS2<br>• X'02' UTF16<br><br>Initialize to binary zeros. |
| IRM_CLIENTID | 8 bytes | A string of 1 to 8 uppercase alphanumeric (A through Z, 0 to 9) or special (@, #, $) characters, left justified, and padded with blanks. IRM_CLIENTID specifies the name of the client ID that is used by IMS Connect. If this string is not supplied by the client, then the user exit must generate it.<br><br>The client ID is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_CLID. |

In Table 6, following the client ID (IRM_CLIENTID) of the common portion of the prefix is the user portion (required by IMS Connect):

*Table 6. User Portion*

| Field | Length | Meaning |
|---|---|---|
| Datastore ID | 8 bytes | The Datastore ID passed by the client can be changed or supplied by the exit. The Datastore ID must be returned to IMS Connect in the OTMA user data header field OMUSR_DESTID. |

The following items should be considered for your installation:

- RACF values
  - User ID
    8 bytes
  - Group Name
    8 bytes
  - Pass ticket
    8 bytes
  - APPL Name
    8 bytes
  - Transaction code
    8 bytes
  - TRUSTED USER Byte(s)
    n byte(s)
- LTERM override name
  8 bytes
- MFS MID name
  8 bytes
- MFS MOD name
  8 bytes
- Asynchronous output timer
  1 byte
- Other required data for user-written exit
- Flag bytes for:
  - MFS MOD name to be returned
  - Commit mode
  - Sync level
  - ACK, NAK, and deallocate
  - Security scope
  - Socket type

Table 7 shows an example of a fixed IRM format preceding the input message. This format is used by the HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 exits. Following the IRM structure is the IMS data that is composed of LLZZDATA where LL= the total length of this segment (includes LL=length, ZZ=binary zeros, and DATA=IMS trancode followed by transaction data).

The last 11 rows of Table 7, as well as the paragraph immediately following the table, contain Product-Sensitive Programming Interface and Associated Guidance Information.

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format*

| Field | Length | Meaning |
|---|---|---|
| Message field. | | |

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| llll | 4 bytes | Length of the total message. The total message length includes the length of the IRM (variable, depending on your requirements), the HDR_LLLL field (four bytes), the length of the message, and the end of message indicator X'0004000' (four bytes). The value is between X'58' and X'7FFFFFFF'. This field is read as a binary number. |
| The following fields are for the 28 (decimal) byte IRM prefix. | | |
| IRM_LEN | 2 bytes | Length of the IRM structure. The minimum size of the IRM for user written exits is X'24' or binary ’00100100’. HWSIMSO0 and HWSSMPL0 have a minimum IRM length of X'50' or binary ’01010000’. |
| IRM_RSV | 2 bytes | Reserved. Set to binary ’00000000’. |
| IRM_ID | 8 bytes | Character string. It specifies the identifier of the user exit that is to be driven after the complete message has been received. For details, see “How IMS Connect Communicates with User Message Exits” on page 60. The IMS Connect-supplied user message exits reserve and use these IDs: • `*IRMREQ*`-- for HWSIMSO0 • `*IRMRE1*`-- for HWSIMSO1 • `*SAMPLE*`-- for HWSSMPL0 • `*SAMPL1*`-- for HWSSMPL1 • `*HWSJAV*`-- for HWSJAVA0 • `*HWSCSL*`-- for HWSCSLO0 |
| Reserved | 4 bytes | Reserved for future use. Set to binary ’00000000’. |

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format (continued)*

| Field | Length | Meaning | |
|---|---|---|---|
| IRM_F5 | 1 byte | Input message type. | |
| | | **X'80'** | OTMA headers built by client. |
| | | **X'40'** | Translation done by client. |
| | | **X'00'** | No option flow of messages (see meaning for X'04'). This is the default if no value is specified. |
| | | **X'01'** | Single message. Only one message returned on receive following resume tpipe. |
| | | **X'02'** | Auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a large value. Use only if the client is a dedicated output client. |
| | | **X'04'** | No auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a small value. Use only if the client is a dedicated output client. This value is similar to Auto, except that the IRM_TIMER will cause the last receive to terminate. |
| IRM_TIMER | 1 byte | Time delay that IMS Connect will wait for IMS to return data to IMS Connect which, in turn, will be sent to the client. The following functions support the IRM_TIMER settings: • TCP/IP SEND of a RESUME TPIPE • TCP/IP SEND of an ACK or NAK • TCP/IP SEND of data • PC SEND of a RESUME TPIPE • PC SEND of an ACK or NAK • PC SEND of data **Note:** See "IRM_TIMER Usage" on page 52. | |

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_SOCT | 1 byte | Socket connection type. The client can set this value as follows:<br>• X'00' - transaction socket. The socket connection lasts across a single transaction.<br>• X'10' - persistent socket. The socket connection lasts across multiple transactions.<br>• X'40' - non-persistent socket. The socket connection lasts for a single exchange consisting of one input and one output. **Recommendation**: Do not use this socket type if you plan on implementing conversational transactions, because multiple connects and disconnects will occur. |
| IRM_ES | 1 byte | Unicode encoding schema:<br>• X'01' UTF8<br>• X'02' UCS2<br>• X'02' UTF16<br><br>Set to binary zeros. |
| IRM_CLIENTID | 8 bytes | A string of 1 to 8 uppercase alphanumeric (A through Z, 0 to 9) or special (@, #, $) characters, left justified, and padded with blanks. It specifies the name of the client ID that is used by IMS Connect. If this string is not supplied from the client, then the user exit must generate it.<br><br>The client ID is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_CLID. |
| The following fields are for the user-defined portion of the IRM header. | | |
| IRM_F1 | 1 byte | This value is used to specify if the MFS mod name is to be returned.<br>• X'00' - user requests no MFS mod name to be returned.<br>• X'80' - user requests MFS mod name to be returned.<br><br>If this value is not supplied by the client, the user exit must use a default value.<br><br>The MFS mod name flag is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_UFLAG1. |
| IRM_F2 | 1 byte | It specifies the commit mode.<br>• X'40' - commit mode ’0’.<br>• X'20' - commit mode ’1’.<br><br>If this value is not supplied from the client, the user exit must use a default value.<br><br>The commit mode flag is returned to IMS Connect from the exit in the OTMA header field, OMHDRSYN. |

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_F3 | 1 byte | It specifies the sync level.<br>• X'00'- sync level is 'NONE'.<br>• X'01' - sync level is 'CONFIRM'.<br>• X'02' - sync level is SYNCPT<br>• X'04' - purge not deliverable<br><br>For Commit Mode 0, the sync level must be set to confirm. If the synch level is not supplied from the client, the user exit must use a default value.<br><br>The sync level flag is returned to IMS Connect from the exit in the OTMA header field, OMHDRSLV.<br><br>The purge not deliverable flag is returned to IMS Connect from the user message exit in the OTMA header field, OMHDRCFL, with the setting of OMHDRPND X'10'. |

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_F4 | 1 byte | Character value. It specifies if the client is sending:<br>• A = ACK - Positive acknowledgment<br>• C = CANCEL TIMER - Cancel the timer<br>• N = NAK - Negative acknowledgment<br>• D = DEALLOCATE - Deallocate connection<br>• R = RESUME - Resume tpipe<br>• S = SENDONLY - Send only<br>• blank (binary ’01000000’) = no request for acknowledgment or deallocation<br><br>The value is sent to IMS Connect to be forwarded to IMS. When the value is received and passed to the user exit, the exit builds the appropriate OTMA structure and returns it to IMS Connect.<br><br>The ACK / CANCEL TIMER / NAK / DEALLOCATE / RESUME (A/C/N/D/R) must be sent to IMS Connect with no data element.<br>• ACK/NAK is used in response to a message sent to the client where SYNC level = CONFIRM.<br>• CANCEL TIMER is used to request to cancel the timer associated with the wait for data from IMS.<br>• DEALLOCATE is used to terminate a conversation rather than complete the conversation.<br>• RESUME TPIPE is used to request asynchronous output data from IMS. The RESUME TPIPE must execute on a transaction socket as commit mode zero.<br><br>SENDONLY is used to send the transaction code and data for a non-response transaction to IMS Connect, and to request that no DFS2082 message be returned to the client if the host application terminates without issuing an ISRT to the IO PCB. The SENDONLY must execute as commit mode zero.<br><br>A blank X'40' is used to send the transaction code and data, or data only, for response mode transactions and conversational mode. |
| IRM_TRNCOD | 8 bytes | Character string. It specifies the IMS transaction code. |
| IRM_IMSDESTID | 8 bytes | Character string. It specifies the Datastore name (IMS destination ID). This field must be specified by the client. The Datastore name is returned to IMS Connect from the exit in the OTMA header field, OMUSR_DESTID. |

*Table 7. HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 Message Fixed Format (continued)*

| Field | Length | Meaning |
|---|---|---|
| IRM_LTERM | 8 bytes | Character string. It specifies the IMS LTERM override. This field can be set to a valid name or to blanks.<br><br>The LTERM override name is returned to IMS Connect from the exit in the OTMA header field, OMHDRLM.<br><br>For IMS host applications, the value for this field is set by the user message exit, which either moves this value to the OTMA field OMHDRLTM or sets OMHDRLTM with a predetermined value. If you have specified an LTERM override value, OTMA places that value in the IOPCB LTERM field. If you do not specify an LTERM override value, OTMA instead places the IMS Connect-defined TPIPE name in the IOPCB LTERM field. The TPIPE name is set to the CLIENT ID if the commit mode is zero; it is set to the PORT ID if the commit mode is one.<br><br>If you use the LTERM value in the IOPCB to make logic decisions, be aware of the naming conventions of the IOPCB LTERM name. |
| IRM_RACF_USERID | 8 bytes | Character string. It specifies the RACF user ID. The client must provide it if RACF is to be used.<br><br>The RACF user ID name is returned to IMS Connect from the exit in the OTMA header field, OMSECUID. |
| IRM_RACF_GRNAME | 8 bytes | Character string. It specifies the RACF group name. The client must provide it if RACF is to be used.<br><br>The RACF group name is returned to IMS Connect from the exit in the OTMA header field, OMSECGRP. |
| IRM_RACF_PW | 8 bytes | Character string. It specifies the RACF PassTicket or PASSWORD. The client must provide it if RACF is to be used.<br><br>The PassTicket or PASSWORD value is returned to IMS Connect from the user message exit, in the OTMA header field, OMUSR_PASSTICK. |
| IRM_APPL_NM | 8 bytes | Character string. It specifies the RACF APPL name, that was defined to RACF on the PTKTDATA definition. (This is not supported for HWSIMSO0 or HWSIMSO1.) |

For the complete non-IMS Connector for Java message structure used by the HWSIMSO0, HWSIMSO1, HWSSMPL0, and HWSSMPL1 exits, see the table under “Non-IMS Connector for Java Message Structure - Type 2” on page 75.

## IRM_TIMER Usage

Set the IRM_TIMER value to an appropriate wait time for IMS to return data to IMS Connect. Each client SEND, within a transaction, can have a different IRM_TIMER value.

The settings for the IRM_TIMER is enforced as described in the following list:

1. If the IRM_TIMER is set at X'00', the following default values are used:
   - The default for all RESUME_TPIPE is two seconds.
   - The default for all RESUME_TPIPE non-single ACK is .25 seconds.
   - The TIMER setting in the configuration file for all others.
2. X'FF' and X'01' - X'9E' are used only when requested.
3. X'E9' (char Z) NO_WAIT means do not wait for any IMS output. NO_WAIT is not valid on some Client SENDs. Because IMS Connect does not wait for output from IMS, on a transaction socket connection, IMS Connect disconnects the socket; and on a persistent socket connection, IMS Connect requests the next input from the client rather than disconnect the socket. If NO_WAIT is used, it is enforced as follows:
   - There is a two second delay for:
     – RESUME_TPIPE request
     – conversational trancode
     – conversational data
     – ACK or NAK associated with a conversational transaction
     – non-conversational trancode
   - A .25 second delay for each of the following is used:
     – an ACK or NAK associated with a non-conversational transaction commit mode one confirm
     – an ACK or NAK associated with a RESUME_TPIPE with Asynch output options AUTO or NOAUTO
     – an ACK or NAK associated with non-conversational transaction commit mode zero confirm
   - NO_WAIT can be used for the following:
     – a SENDONLY
     – an ACK or NAK associated with RESUME_TPIPE with Asynch output option SINGLE

Misuse of X'E9' may result in one of the following problems:

1. The socket disconnects.
2. An output message to the client on a transaction socket is lost.
3. A hang condition occurs between the client and IMS Connect or IMS Connect and OTMA. For example, the client can be in a READ state waiting for output from IMS Connect while IMS Connect is in a READ state waiting for input from the client and OTMA is in READ state waiting for acknowledgement.
4. The deallocate commit or deallocate abort notification for Commit Mode 1 SynchLevel=Confirm is lost.
5. Other unpredictable conditions occur.

To determine the appropriate wait time for IMS to return data to IMS Connect, consider the following guidelines:

- For a client SEND of trancode and data or data only, the IRM_TIMER value should be set to reflect the amount of time IMS Connect should wait for the output from IMS. It is recommended that X'E9' not be used.
- If the client application knows that the last message received is the last output message to the client for the transaction, then it is recommended that the IRM_TIMER be set to X'01' (.01 of a second) for a client SEND of ACK or NAK. The IRM_TIMER of X'01' is the smallest value that can be set for non-RESUME TPIPE ACKs. However, if the ACK is associated with an output from a RESUME TPIPE, then an IRM_TIMER value of X'E9' (character Z) is **not** recommended.
- For a client SEND of RESUME TPIPE, the timer value can be set as follows:

  **AUTO option**
  X'FF' for dedicated output device, or

  any X'00' to X'9E' values for non-dedicated output device

  **NOAUTO option**
  any value other than X'FF' or X'E9'

  **SINGLE or SINGLE with WAIT option**
  any value other than X'FF' or X'E9'

Table 8 lists the IRM_TIMER values and their corresponding time in hundredths of a second.

*Table 8. IRM_TIMER Values in Hundredths of a Second*

| Value | Time |
|---|---|
| X'01' | .01 of a second |
| X'02' | .02 of a second |
| X'03' | .03 of a second |
| X'04' | .04 of a second |
| X'05' | .05 of a second |
| X'06' | .06 of a second |
| X'07' | .07 of a second |
| X'08' | .08 of a second |
| X'09' | .09 of a second |
| X'0A' | .10 of a second |
| X'0B' | .11 of a second |
| X'0C' | .12 of a second |
| X'0D' | .13 of a second |
| X'0E' | .14 of a second |
| X'0F' | .15 of a second |
| X'10' | .16 of a second |
| X'11' | .17 of a second |
| X'12' | .18 of a second |
| X'13' | .19 of a second |
| X'14' | .20 of a second |
| X'15' | .21 of a second |
| X'16' | .22 of a second |

Table 8. IRM_TIMER Values in Hundredths of a Second (continued)

| Value | Time |
|---|---|
| X'17' | .23 of a second |
| X'18' | .24 of a second |
| X'19' | .25 of a second |
| X'1A' | .30 of a second |
| X'1B' | .35 of a second |
| X'1C' | .40 of a second |
| X'1D' | .45 of a second |
| X'1E' | .50 of a second |
| X'1F' | .55 of a second |
| X'20' | .60 of a second |
| X'21' | .65 of a second |
| X'22' | .70 of a second |
| X'23' | .75 of a second |
| X'24' | .80 of a second |
| X'25' | .85 of a second |
| X'26' | .90 of a second |
| X'27' | .95 of a second |

Table 9 lists the IRM_TIMER values and their corresponding time in seconds.

Table 9. IRM_TIMER Time Values in Seconds

| Value | Time |
|---|---|
| X'28' | 1 second |
| X'29' | 2 seconds |
| X'2A' | 3 seconds |
| X'2B' | 4 seconds |
| X'2C' | 5 seconds |
| X'2D' | 6 seconds |
| X'2E' | 7 seconds |
| X'2F' | 8 seconds |
| X'30' | 9 seconds |
| X'31' | 10 seconds |
| X'32' | 11 seconds |
| X'33' | 12 seconds |
| X'34' | 13 seconds |
| X'35' | 14 seconds |
| X'36' | 15 seconds |
| X'37' | 16 seconds |
| X'38' | 17 seconds |
| X'39' | 18 seconds |

*Table 9. IRM_TIMER Time Values in Seconds (continued)*

| Value | Time |
|---|---|
| X'3A' | 19 seconds |
| X'3B' | 20 seconds |
| X'3C' | 21 seconds |
| X'3D' | 22 seconds |
| X'3E' | 23 seconds |
| X'3F' | 24 seconds |
| X'40' | 25 seconds |
| X'41' | 26 seconds |
| X'42' | 27 seconds |
| X'43' | 28 seconds |
| X'44' | 29 seconds |
| X'45' | 30 seconds |
| X'46' | 31 seconds |
| X'47' | 32 seconds |
| X'48' | 33 seconds |
| X'49' | 34 seconds |
| X'4A' | 35 seconds |
| X'4B' | 36 seconds |
| X'4C' | 37 seconds |
| X'4D' | 38 seconds |
| X'4E' | 39 seconds |
| X'4F' | 40 seconds |
| X'50' | 41 seconds |
| X'51' | 42 seconds |
| X'52' | 43 seconds |
| X'53' | 44 seconds |
| X'54' | 45 seconds |
| X'55' | 46 seconds |
| X'56' | 47 seconds |
| X'57' | 48 seconds |
| X'58' | 49 seconds |
| X'59' | 50 seconds |
| X'5A' | 51 seconds |
| X'5B' | 52 seconds |
| X'5C' | 53 seconds |
| X'5D' | 54 seconds |
| X'5E' | 55 seconds |
| X'5F' | 56 seconds |
| X'60' | 57 seconds |
| X'61' | 58 seconds |

Table 9. IRM_TIMER Time Values in Seconds (continued)

| Value | Time |
|---|---|
| X'62' | 59 seconds |
| X'63' | 60 seconds |

Table 10 lists the IRM_TIMER values and their corresponding time in minutes.

Table 10. IRM_TIMER Time Values in Minutes

| Value | Time |
|---|---|
| X'63' | 1 minute |
| X'64' | 2 minutes |
| X'65' | 3 minutes |
| X'66' | 4 minutes |
| X'67' | 5 minutes |
| X'68' | 6 minutes |
| X'69' | 7 minutes |
| X'6A' | 8 minutes |
| X'6B' | 9 minutes |
| X'6C' | 10 minutes |
| X'6D' | 11 minutes |
| X'6E' | 12 minutes |
| X'6F' | 13 minutes |
| X'70' | 14 minutes |
| X'71' | 15 minutes |
| X'72' | 16 minutes |
| X'73' | 17 minutes |
| X'74' | 18 minutes |
| X'75' | 19 minutes |
| X'76' | 20 minutes |
| X'77' | 21 minutes |
| X'78' | 22 minutes |
| X'79' | 23 minutes |
| X'7A' | 24 minutes |
| X'7B' | 25 minutes |
| X'7C' | 26 minutes |
| X'7D' | 27 minutes |
| X'7E' | 28 minutes |
| X'7F' | 29 minutes |
| X'80' | 30 minutes |
| X'81' | 31 minutes |
| X'82' | 32 minutes |
| X'83' | 33 minutes |

Table 10. IRM_TIMER Time Values in Minutes (continued)

| Value | Time |
|---|---|
| X'84' | 34 minutes |
| X'85' | 35 minutes |
| X'86' | 36 minutes |
| X'87' | 37 minutes |
| X'88' | 38 minutes |
| X'89' | 39 minutes |
| X'8A' | 40 minutes |
| X'8B' | 41 minutes |
| X'8C' | 42 minutes |
| X'8D' | 43 minutes |
| X'8E' | 44 minutes |
| X'8F' | 45 minutes |
| X'90' | 46 minutes |
| X'91' | 47 minutes |
| X'92' | 48 minutes |
| X'93' | 49 minutes |
| X'94' | 50 minutes |
| X'95' | 51 minutes |
| X'96' | 52 minutes |
| X'97' | 53 minutes |
| X'98' | 54 minutes |
| X'99' | 55 minutes |
| X'9A' | 56 minutes |
| X'9B' | 57 minutes |
| X'9C' | 58 minutes |
| X'9D' | 59 minutes |
| X'9E' | 60 minutes |

Table 11 lists additional values that can be used for IRM_TIMER and provides a brief explanation of the values.

Table 11. Additional IRM_TIMER Time Values

| Value | Description |
|---|---|
| X'00' | Use default value: <br>• for RESUME TPIPE and associated ACK, the default is .25 seconds <br>• for all other SENDs, the default is the configuration file TIMEOUT value |
| X'E9' C'Z' | No timer (no wait occurs). |
| X'FF' | Wait indefinitely. This setting is intended to support the auto option of the asynchronous output function. |

## Output from Client Exit

Table 12 shows the structure (one occurrence per message) of the message returned by the non-IMS Connector for Java client exit. The table lists the field name, the length of the field, and a brief explanation of the field.

*Table 12. Structure 1*

| Field | Length | Meaning |
|---|---|---|
| BPE header | 64 bytes | Defined in Table 14 on page 59. |
| OTMA structure | Total length of OTMA header | For more information about the HWSOMPFX macro (full OTMA structure) see Appendix B, “OTMA Headers,” on page 235. |
| LLZZTRANCODEDATA | n bytes | • LL - length of segment<br>• ZZ - set to binary zeros<br>• TRANCODE - IMS 1-8 byte transaction code<br>• DATA - user data |
| LLZZDATA | n bytes | • LL - length of segment<br>• ZZ - set to binary zeros<br>• DATA - user data |
| The LLZZDATA is repeated to a maximum of 32 KB overall length. If there is more data, then the structures continues as shown in Table 13. Only the first segment will contain the IMS transaction code and the following segment will contain the segment data necessary for the transaction to process. | | |
| LL | 2 bytes | LL - set to binary zeros to denote the end of this structure. The LL field is not part of the segment length. |

## Other IMS Connect Structures

Table 13 shows other structures that are repeated until all data has been mapped to be returned to IMS Connect. The table lists the field name, the length of the field, and a brief explanation of the field.

*Table 13. Structure 2*

| Field | Length | Meaning |
|---|---|---|
| BPE header | 64 bytes | Defined in Table 14 on page 59. |
| OTMA structure | 32 bytes | For more information about the HWSOMPFX macro (control OTMA structure only), see Appendix B, “OTMA Headers,” on page 235. |
| LLZZDATA | n bytes | • LL - length of segment<br>• ZZ - set to binary zeros<br>• DATA - user data |

*Table 13. Structure 2 (continued)*

| Field | Length | Meaning |
|---|---|---|
| The LLZZDATA is repeated to a maximum of 32 KB overall length. If there is more data, then the structures continues. | | |
| LL | 2 bytes | LL - set to binary zeros to denote the end of this structure. |

Table 14 lists the fields in the BPE header layout. It also contains Product-Sensitive Programming Interface and Associated Guidance Information.

*Table 14. BPE Header Layout*

| Field | Length | Meaning |
|---|---|---|
| llll | 4 bytes | The length of the total structure and it is set for the first BPE header only. This field is managed by IMS Connect and must not be altered by the exit. |
| CHAIN PTR | 4 bytes | The chain pointer to the next BPE header within this message. The last BPE header in the message must have binary zeros as a chain pointer value to denote the end of the BPE headers within the message.<br><br>These chain pointers are set by the non-IMS Connector for Java user exit. |
| STORAGE TYPE | 8 bytes | This field is managed by IMS Connect and should not be modified by the user exit. |
| TYPE ACCESS | 4 bytes | This field is managed by IMS Connect and should not be modified by the user exit. |
| SUBPOOL | 1 byte | This field is managed by IMS Connect and should not be modified by the user exit. |
| Reserved | 43 bytes | This field is managed by IMS Connect and should not be modified by the user exit. |

## How IMS Connect Communicates with an SSL Client

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protect the privacy and integrity of data that is transferred through a network. SSL rests on top of TCP/IP to provide a mechanism for secure sockets. SSL uses a combination of public and private keys and symmetric key encryption to authorize clients and servers to one another. Once an SSL connection is established between a client and server, data communications between client and server are transparent to the encryption and integrity is added by the SSL protocol.

Because the SSL interface directly overlays the TCP/IP layer, it uses the same message formats as the TCP/IP message formats sent to IMS Connect. See “How IMS Connect Communicates with a TCP/IP Client” on page 39 for information about the message formats.

## How IMS Connect Communicates with User Message Exits

When the IMS Connect starts, it loads user message exits one at a time and calls each user message exit INIT subroutine.

**Example**: USREXIT1, USREXIT2, and USREXIT3 are defined in the HWSCFG parameter of the IMS Connect startup JCL as follows:

```
TCPIP=(HOSTNAME=...,EXIT=(USREXIT1,USREXIT2,USREXIT3),...)
```

IMS Connect loads USREXIT1 first and calls the USREXIT1 INIT subroutine. After successfully loading USREXIT1, IMS Connect loads USREXIT2 and calls the USREXIT2 INIT subroutine, and then repeats this process for USREXIT3. Any unsuccessful loading or INIT failure prevents IMS Connect from connecting with TCP/IP.

**Important:** If you define a user exit name in the IMS Connect configuration member, but that user exit cannot be loaded during IMS Connect startup, the job abends with Abend 806, RC=4.

In order to provide full user exit support in the IMS Connect environment, every user exit routine must include the subroutines INIT, READ, XMIT, TERM, and EXER. IMS Connect only supports assembler language exits.

When a user exit takes control, it saves the contents of the registers and restores them when returning to the caller. IMS Connect provides a 1 KB buffer in the parmlist to be used for this purpose.

### Register Contents on Subroutine Entry

Table 15 provides a brief description of the contents of the register on a subroutine entry.

*Table 15. Register Contents on Subroutine Entry*

| Register | Contents |
|---|---|
| 1 | Pointer to a parmlist that is defined in the HWSEXPRM macro. |
| 14 | Return address of IMS Connect. |
| 15 | Entry point address to the user exit routine. The entry point name and load module name for a user exit routine must be the same as the name used for the user exit routine in HWSCFG. |

### Register Contents on Subroutine Exit

Table 16 provides a brief description of the contents of the register on a subroutine exit.

*Table 16. Register Contents on Subroutine Exit*

| Register | Contents |
|---|---|
| 1 | Pointer to a parmlist that is defined in the HWSEXPRM macro. |

## INIT Subroutine

After a user exit has been successfully loaded, the INIT subroutine for that user exit is called and a parmlist is passed to that user exit.

### Contents of Parmlist Pointed to by Register 1 at INIT Subroutine Entry

Table 17 lists the contents of the parmlist that is passed to the user exit at entry.

*Table 17. Contents of Parmlist Pointed to by Register 1 at INIT Subroutine Entry*

| Field | Length | Meaning |
|---|---|---|
| EXPRM_FUNCTION | 4 bytes | Character string of value INIT. Specifies that the function to be performed is: `Initialize user exit`. |
| EXPRM_TOKEN | 4 bytes | Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and for local variables. |
| EXPRM_XIB | 4 bytes | Address of XIB (exit interface block). |

The user exit finishes all its initialization processes here. It returns two MSGID identifiers for the messages that it is to handle, as well as the increase to the output buffer size for its READ, XMIT, and EXER subroutines. The user exit returns the *increase* in buffer size, but not the actual buffer size. The only reason to return anything other than 0 is to allow the exit to add data to the data portion of the message. The storage required for the BPE headers and OTMA headers is computed by IMS Connect. Typically, one of the MSGIDs is used by ASCII clients and the other by EBCDIC clients. IMS Connect computes the actual size of the output buffer, and it allocates the buffer size before it passes control to the user exit for READ, XMIT and EXER. The two identifiers can take any value, in EBCDIC or ASCII, other than the three reserved MSGIDs (see ″Important,″ which follows), provided that the values are both unique among user exits called by a given IMS Connect. Blanks and binary 0 are significant. The IMS Connect saves these identifiers to identify the owner of the incoming request messages. Any conflict in the identifiers must be resolved before a TCP/IP connection can be made.

**Important**: The following MSGIDs are reserved:

- `*IRMREQ*`-- Supports existing IMS TCP/IP messages that use the HWSIMSO0 user exit
- `*IRMRE1*`-- Supports existing IMS TCP/IP messages that use the HWSIMSO1 user exit
- `*HWSJAV*`-- Supports IMS Connector for Java clients
- `*HWSCSL*`-- Supports the IMSplex connection that uses the HWSCSLO0 user message exit
- `*HWSCS1*`-- Supports the IMSplex connection that uses the HWSCSLO1 user message exit
- `*SAMPLE*`-- Supports non-IMS Connector for Java clients that use the HWSSMPL0 user exit
- `*SAMPL1*`-- Supports non-IMS Connector for Java clients that use the HWSSMPL1 user exit

If duplicate MSGID identifiers exist, one of the user exits that uses the conflicting identifier must either be dropped or be rewritten with a unique identifier. A system administrator should coordinate the assignment of MSGIDs.

## Contents of Parmlist Pointed to by Register 1 at INIT Subroutine Exit

Table 18 lists the contents of the parmlist that is pointed to by Register 1 and then passed to the user exit during exit.

*Table 18. Contents of Parmlist Pointed to by Register 1 at INIT Subroutine Exit*

| Field | Length | Meaning |
|---|---|---|
| Reserved | 68 bytes | Reserved space. |
| EXPINI_RETCODE | 4 bytes | Binary. Specifies the return code, which can be one of the following: • 0=INIT function was successful. • 4=INIT function was not successful. |
| EXPINI_RSNCODE | 4 bytes | Binary. Specifies the reason code. |
| EXPINI_STRING1 | 8 bytes | Character string. Specifies the first MSGID that clients can use to identify this user exit. This MSGID could be used for ASCII clients. |
| EXPINI_STRING2 | 8 bytes | Character string. Specifies the second MSGID that clients can use to identify this user exit. This MSGID could be used for EBCDIC clients. |
| EXPINI_BUFINC | 4 bytes | Binary. Specifies the increase size to the output buffer needed to allow the exit to denote that data will be moved from the exit input buffer to the output buffer to add data to the message if required. Field EXPINI_BUFINC is an increased size for input and output messages above what is needed for the BPE and OTMA headers. If, for example, you want to have the exit add data to the message either on input or output, then there will be increase in buffer size. |

If the INIT subroutine fails to complete the initialization function successfully, the IMS Connect does not connect with TCP/IP. A system programmer can start the connection after the problem has been fixed by issuing the `OPENPORT` command. When all user exits have been loaded and initialized, the IMS Connect is ready to receive messages from TCP/IP application programs. The IMS Connect uses the TCP/IP Socket API to receive stream data across the net. The completion of a message is determined by its `MSGLength` value returned by TCP/IP to IMS Connect. The IMS Connect receives data up to the value specified in `MSGLength` and uses `MSGID` to determine which user exit receives control for processing the request message.

## READ Subroutine

After a complete request message that originated at a TCP/IP client has been received, control is passed to the READ subroutine in the user exit whose `MSGID` matches the `MSGID` of that request message and a parmlist is passed to that user exit.

### Contents of Parmlist Pointed to by Register 1 at READ Subroutine Entry

Table 19 lists the contents of the parmlists which are pointed to by Register 1 during the READ subroutine entry.

*Table 19. Contents of Parmlist Pointed to by Register 1 at READ Subroutine Entry*

| **Field** | **Length** | **Meaning** |
|---|---|---|
| EXPRM_FUNCTION | 4 bytes | Character string of value READ. Specifies that the function to be performed is: `Read client data and convert it to OTMA format`. |
| EXPRM_TOKEN | 4 bytes | Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables. |
| EXPRM_XIB | 4 bytes | Address of XIB (exit interface block). |
| EXPREA_INBUF | 4 bytes | Address of the input buffer. |
| EXPREA_IBUFSIZE | 4 bytes | Binary. Specifies the size of the input buffer. |
| EXPREA_OUTBUF | 4 bytes | Address of the output buffer. |
| EXPREA_OBUFSIZE | 4 bytes | Binary. Specifies the size of the output buffer. |
| EXPREA_FLAG1 | 1 byte | Data string flag:<br>• X'80' - Input data contains a MSGID matching `EXPINI_STRING1`.<br>• X'40' - Input data contains a MSGID matching `EXPINI_STRING2`. |
| EXPREA_FLAG2 | 1 byte | Data flag:<br>• X'01' - Data moved by exit from INBUF to OUTBUF.<br>• X'02' - If this EXPREA_IPV6 bit is turned on, IPV6 is enabled. Map EXPREA_SOCKET6 to AF-INET6 socket address structure. |
| Reserved | 2 bytes | Reserved space. |
| EXPREA_RACFID | 8 bytes | Character string. Specifies the default user ID for RACF. |

*Table 19. Contents of Parmlist Pointed to by Register 1 at READ Subroutine Entry (continued)*

| Field | Length | Meaning |
|---|---|---|
| The following 28 bytes have two definitions: one definition is for a 4 byte IPV4 address (EXPREA_NAMEID) and another definition is for a 16 byte IPV6 address (EXPREA_SOCKET6). | | |
| **4 byte IPV4 address:** | | |
| EXPREA_NAMEID | 0 bytes | Pointer referenced to the next 16 bytes. |
| EXPREA_FAMILY | 2 bytes | Binary. Specifies the client family type. |
| EXPREA_PORT | 2 bytes | Binary. Specifies the client port number. |
| EXPREA_ADDRESS | 4 bytes | Client's IP address. |
| EXPREA_RESERVE | 8 bytes | Reserved space. |
| | 12 bytes | Reserved. |
| **16 byte IPV6 address:** | | |
| EXPREA_SOCKET6 | 0 bytes | Map to the AF_INET6 socket address structure (if the EXPREA_IPV6 bit of EXPREA_FLAG2 is turned on). |
| EXPREA_6LEN | 1 byte | Address of socket length. |
| EXPREA_6FAMILY | 1 byte | Address of family. |
| EXPREA_6PORT | 2 bytes | Port number used by the application. |
| EXPREA_6FLOW | 4 bytes | Flow information. |
| EXPREA_6ADDR | 16 bytes | INET address (NETID). |
| EXPREA_6SCOPE | 4 bytes | Scope ID. |

`EXPREA_IBUFSIZE` and `EXPREA_OBUFSIZE` are the sizes of the input buffer and output buffer, respectively. These sizes are not related to the actual length of the input data and output data. The input buffer contains an exact copy of the data that was received from the client. The user exit might need to perform an ASCII-to-EBCDIC conversion on the data so that the data can be properly interpreted by the IMS application. The user exit can use `EXPREA_FLAG1` to determine where the data originated and whether additional processing is required by the exit.

IMS Connect also supplies the default RACF user ID and the client's TCP/IP connection information to the user exit. At this point, the user exit might edit or filter its client's input data, then translate that data to OTMA message segments and place them in the output buffer. The user exit also must specify the length of the output data in `EXPREA_DATALEN`.

## Contents of Parmlist Pointed to by Register 1 at READ Subroutine Exit

Table 20 on page 65 lists the contents of the parm list that are pointed to by Register 1 during the subroutine exit.

*Table 20. Contents of Parmlist Pointed to by Register 1 at READ Subroutine Exit*

| Field | Length | Meaning |
|---|---|---|
| Reserved | 68 bytes | Reserved space. |
| EXPREA_RETCODE | 4 bytes | Binary. Specifies the return code, which can be one of the following values: • 0=READ function was successful. Process the data. • 4=READ function was not successful. Send the data in `EXPREA_OUTBUF` back to client. • 8=READ function was not successful. Just clean up. |
| EXPREA_RSNCODE | 4 bytes | Binary. Specifies the reason code. |
| EXPREA_DATALEN | 4 bytes | Binary. Specifies the size of data in the `EXPREA_OUTBUF` to be returned to IMS Connect. This field is only meaningful when `EXPREA_RETCODE` = 0 or 4. |
| EXPREA_UFLAG1 | 1 byte | User flag: • X'80' - Client requests IMS MOD name be returned |
| Reserved | 3 bytes | Reserved space. |
| EXPREA_CLID | 8 bytes | Character string. It specifies the client ID name passed by the client or generated by the exit for non-IMS Connector for Java clients only. |

The output buffer contains data when the return code is 0 or 4. When the return code is 4, the data in the output buffer is sent back to the user exit’s client, and then the connection is closed and cleaned up. When the return code is 0, the IMS Connect prepares to present the data to a datastore. `EXPREA_UFLAG1` is also saved by the IMS Connect. This flag is set by the user exit during READ subroutine processing and is used for recording user selected characteristics of the request message. This flag is passed back to the user exit in the input parmlist pointed to by Register 1 on the next subroutine call, which is either an XMIT or an EXER subroutine call. You define the value of `EXPREA_UFLAG1` in the user exit code. IMS Connect uses this value to provide a communication vehicle between the READ and XMIT or EXER subroutines on a per request/response message basis. The XMIT and EXER subroutines can thus format the message in a better manner.

If IMS Connect detects an error in the output data that would prevent it from properly presenting the data to the datastore (for example, the output data is not formatted properly to conform to the IMS OTMA protocol), the EXER subroutine is called where the error can be dealt with appropriately. IMS Connect then waits until it receives the response message from IMS OTMA. After receiving a response, it calls the XMIT subroutine of the appropriate user exit (based on the MSGID in the response) and passes it an exact copy of the response data that it received from IMS OTMA.

## XMIT Subroutine

After a complete response message has been received from the datastore, control is passed to the XMIT subroutine in the user exit whose MSGID matches the MSGID of the response message (which in turn matches the MSGID of the original request message) and a parmlist is passed to that user exit.

### Contents of Parmlist Pointed to by Register 1 at XMIT Subroutine Entry

Table 21 lists the contents of the parmlist that are pointed to by Register 1 during the XMIT subroutine entry and passed to the user exit.

*Table 21. Contents of Parmlist Pointed to by Register 1 at XMIT Subroutine Entry*

| Field | Length | Meaning |
|---|---|---|
| EXPRM_FUNCTION | 4 bytes | Character string of value XMIT. Specifies that the function to be performed is: `Read OTMA data and convert it to client format`. |
| EXPRM_TOKEN | 4 bytes | Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables. |
| EXPRM_XIB | 4 bytes | Address of XIB (exit interface block). |
| EXPXMT_INBUF | 4 bytes | Address of the input buffer. |
| EXPXMT_IBUFSIZE | 4 bytes | Binary. Specifies the size of the input buffer. |
| EXPXMT_OUTBUF | 4 bytes | Address of the output buffer. |
| EXPXMT_OBUFSIZE | 4 bytes | Binary. Specifies the size of the output buffer. |
| EXPXMT_FLAG1 | 1 byte | Data string flag: • X'80' - Input data contains a MSGID matching `EXPINI_STRING1`. • X'40' - Input data contains a MSGID matching `EXPINI_STRING2`. |
| EXPXMT_UFLAG1 | 1 byte | User flag. X’xx’ - User-defined value. The value was set in READ subroutine. |
| Reserved | 2 bytes | Reserved space. |

`EXPXMT_IBUFSIZE` and `EXPXMT_OBUFSIZE` are the sizes of the input buffer and output buffer, respectively. These sizes are not related to the actual length of the input data and output data. The input buffer contains an exact copy of the OTMA message segments that were received from the datastore. The user exit might need to perform an EBCDIC-to-ASCII conversion on the data so that the data can be properly interpreted by the client application. The user exit translates OTMA message segments to its client’s data format, places the data in the output buffer, and specifies the length of the output data in `EXPXMT_DATALEN`. The user exit might also edit or filter the output data at this point.

### Contents of Parmlist Pointed to by Register 1 at XMIT Subroutine Exit

Table 22 lists the contents of the parmlist that are pointed to by Register 1 during the XMIT subroutine exit.

*Table 22. Contents of Parmlist Pointed to by Register 1 at XMIT Subroutine Exit*

| Field | Length | Meaning |
|---|---|---|
| Reserved | 68 bytes | Reserved space. |
| EXPXMT_RETCODE | 4 bytes | Binary. Specifies the return code, which can be one of the following values:<br>• 0=XMIT function was successful. Process the data.<br>• 8=XMIT function was not successful. Just clean up. |
| EXPXMP_RSNCODE | 4 bytes | Binary. Specifies the reason code. |
| EXPXMT_DATALEN | 4 bytes | Binary. Specifies the size of data in the `EXPXMT_OUTBUF` to be returned to IMS Connect. This field is only meaningful when `EXPXMT_RETCODE` = 0. |

When the return code is 0, the data in the output buffer is sent back to the originator of the client request message. If the return code is not 0, the connection is dropped. If the user exit sets a non-zero return code value, the connection closes without sending a response back to the originator of the client request message.

## TERM Subroutine

When IMS Connect is shutting down, control is passed, in turn, to the TERM subroutine in each user exit that is currently active, and a parmlist is passed to that user exit.

### Contents of Parmlist Pointed to by Register 1 at TERM Subroutine Entry

Table 23 lists the contents of the parmlist that are pointed to by Register 1 during TERM Subroutine entry and passed to the user exit.

*Table 23. Contents of Parmlist Pointed to by Register 1 at TERM Subroutine Entry*

| Field | Length | Meaning |
|---|---|---|
| EXPRM_FUNCTION | 4 bytes | Character string of value TERM. Specifies that the function to be performed is: `Clean up in preparation for IMS Connect shutdown`. |
| EXPRM_TOKEN | 4 bytes | Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables. |
| EXPRM_XIB | 4 bytes | Address of XIB (exit interface block). |

The user exit finishes all its termination processes here.

### Contents of Parmlist Pointed to by Register 1 at TERM Subroutine Exit

Table 24 lists the contents of the parmlist that are pointed to by Register 1 during the TERM subroutine exit.

*Table 24. Contents of Parmlist Pointed to by Register 1 at TERM Subroutine Exit*

| Field | Length | Meaning |
|---|---|---|
| Reserved | 68 bytes | Reserved space. |
| EXPTRM_RETCODE | 4 bytes | Binary. Specifies the return code, which can be one of the following values: • 0=TERM function was successful. • 4=TERM function was not successful. |
| EXPTRM_RSNCODE | 4 bytes | Binary. Specifies the reason code. The reason codes are set by the exits (HWSIMSO0, HWSIMSO1, HWSSMPL0, HWSSMPL1, and HWSJAVA0). |

IMS Connect shutdown proceeds independently of the return code value. The return code merely indicates the completeness of the user exit cleanup.

## EXER Subroutine

When IMS Connect detects an error in the output buffer after execution of the previous READ subroutine completes, control is passed to the EXER subroutine in the same user exit where the READ subroutine executed and a parmlist is passed to that user exit.

### Contents of Parmlist Pointed to by Register 1 at EXER Subroutine Entry

Table 25 lists the contents of the parmlist that are pointed to by Register 1 during EXER subroutine entry and passed to the user exit.

*Table 25. Contents of Parmlist Pointed to by Register 1 at EXER Subroutine Entry*

| Field | Length | Meaning |
|---|---|---|
| EXPRM_FUNCTION | 4 bytes | Character string of value EXER. Specifies that the function to be performed is: `Process error found in output buffer after previous READ subroutine processing completed`. |
| EXPRM_TOKEN | 4 bytes | Address of a 1 KB buffer for user exit use. The user exit can use this storage for a save area and local variables. |
| EXPRM_XIB | 4 bytes | Address of XIB (exit interface block). |
| EXPXER_OUTBUF | 4 bytes | Address of the output buffer. |
| EXPXER_OBUFSIZE | 4 bytes | Binary. Specifies the size of the output buffer. |
| EXPXER_FLAG1 | 1 byte | Data string flag, which can be one of the following values: • X'80' - Input data contains a MSGID matching `EXPINI_STRING1`. • X'40' - Input data contains a MSGID matching `EXPINI_STRING2`. |
| EXPXER_UFLAG1 | 1 byte | User flag. X'xx' - User-defined value. The value was set in READ subroutine. |

Table 25. Contents of Parmlist Pointed to by Register 1 at EXER Subroutine Entry (continued)

| Field | Length | Meaning |
|---|---|---|
| Reserved | 2 bytes | Reserved space. |
| EXPXER_CODE | 4 bytes | Binary. Specifies the failure code.<br>• 4=Error in the output buffer from the previous READ function. |
| EXPXER_REASON | 4 bytes | Binary. Specifies the failure reason, which can be one of the following:<br>• 20=Segment length error<br>• 24=Missing first in chain flag<br>• 28=Missing last in chain flag<br>• 32=Sequence number error |

The user exit could have experienced difficulties in forming OTMA message segment format and should notify its client of this situation (for example, through an error message). The user exit can use `EXPXER_FLAG1` to determine where the request message from the client originated and whether to compose an ASCII or EBCDIC data stream for sending back to the originating client.

### Contents of Parmlist Pointed to by Register 1 at EXER Subroutine Exit

Table 26 lists the contents of the parmlist that are pointed to by Register 1 during the EXER subroutine exit.

Table 26. Contents of Parmlist Pointed to by Register 1 at EXER Subroutine Exit

| Field | Length | Meaning |
|---|---|---|
| Reserved | 68 bytes | Reserved space. |
| EXPXER_RETCODE | 4 bytes | Binary. Specifies the return code, which can be one of the following values:<br>• 4=Send the data in `EXPXER_OUTBUF` back to client.<br>• 8=Just clean up. |
| EXPXER_RSNCODE | 4 bytes | Binary. Specifies the reason code. |
| EXPXER_DATALEN | 4 bytes | Binary. Specifies the size of data in the `EXPXER_OUTBUF` to be returned to clients. This field is only meaningful when `EXPER_RETCODE`=4. |

When the return code is 4, IMS Connect sends the data in the output buffer back to the client. If the user exit sets the return code value to 8, the connection closes without a response.

## User Exit Message Description and Structures

IMS Connect allows up to 254 user exits to be defined in the configuration file (see the example 18). There are two input message structures supported by IMS Connect and two message structures supported on return from a user exit.

## Input Messages from Client

Table 27 shows the structure for input messages received from the client by IMS Connect. The table provides information about the input message structure type, if the OTMA header is present or not, if the exit data is translated by the client code, the exit type flag, and the supporting message type.

*Table 27. Input Message Structure*

| Input message structure type | OTMA header present | Exit data translated by client code | Exit type flag (IRMHDR_FLG5) | Supporting message type |
|---|---|---|---|---|
| 1 | Y | Y | 11000000 | HWSJAVA0 |
| 1 | Y | N | 10000000 | HWSSMPL0 and HWSSMPL1 modified not to build OTMA headers when the client/server builds OTMA headers |
| 2 | N | Y | 01000000 | HWSSMPL0 and HWSSMPL1 modified not to translate data |
| 2 | N | N | 00000000 | HWSIMSO0 HWSIMSO1 HWSSMPL0 HWSSMPL1 HWSCSLO0 |

Table 28 shows the structure for input messages returned by the exit based on the input structure received by the exit. The table provides information about the input message structure type, the exit output message structure type, the exit type flag, and the supporting message type.

*Table 28. Input Message Structure Returned by the Exit*

| Input message structure type | Exit output message structure type | Exit type flag (IRMHDR_FLG5) | Supporting message type |
|---|---|---|---|
| 1 | 1 | 11000000 | HWSJAVA0 |
| 1 | 1 | 10000000 | HWSSMPL0 and HWSSMPL1 modified not to build OTMA headers when the client/server builds OTMA headers |
| 2 | 3 | 01000000 | HWSSMPL0 and HWSSMPL1 modified not to translate data |
| 2 | 3 | 00000000 | HWSIMSO0 HWSIMSO1 HWSSMPL0 HWSSMPL1 |

## Output Message to Client

The output message from IMS is passed to the user exit that was called from the client. The user exit normally removes the OTMA headers for output if the exit added the OTMA headers for input. The user exit normally translates the data from EBCDIC to ASCII if it did the translation for input. And the reverse is true if these things were not done for input.

The OTMA header can consist of up to four sections and application data. If the exit is to remove the OTMA header (not present on input), there must be a check for each section. The four sections include:

- Control (always present in the OTMA structure)
- Header (might or might not exist in the OTMA structure)
- Security (might or might not exist in the OTMA structure)
- User (might or might not exist in the OTMA structure)

**Output message from IMS to IMS Connect**
All output messages received by IMS Connect from IMS consist of the same structure, the OTMA header followed by LLZZ DATA. If the message contains multiple segments, then the OTMA header and LLZZ DATA are repeated for the number of segments in the message.

**Output message from IMS returned by the exit back to IMS Connect**
The message returned to IMS Connect from the exit consists of one of two structures:

- Messages with OTMA structures imbedded in the message
- Message with no OTMA structures imbedded in the message

## IMS Connect User Message Exit (HWSIMSO0 and HWSIMSO1)

This user message exit is shipped with IMS Connect and link-edited into the IMS Connect RESLIB. You must use the IMS Connect user message exits instead of the one shipped by TCP/IP. The installation must place the IMS Connect RESLIB that contains the IMS Connect supplied exit (HWSIMSO0 and HWSIMSO1) in front of the TCP/IP RESLIB. The HWSIMSO0 and HWSIMSO1 exits are shipped as object code only (OCO). See the user exits HWSIMSO0 and HWSIMSO1. You can modify it as described in “Modifying HWSIMSO0 and HWSIMSO1” on page 35.

**Note**: IMS Connect Version 9 is the final release of these two user message exits. HWSIMSO0 and HWSIMSO1 will not be available in any future IMS Connect release.

The installation can also change the name of this exit to ensure that this exit is called rather than the one shipped with TCP/IP; the new exit name must be specified in the EXIT=( ) parm of the IMS configuration file definition.

The COMMIT mode is set to “1,” and the SYNC level is set to “NONE.” These values can be overridden by supplying the COMMIT mode and/or sync level in the message prefix received from the client (see client message formats in HDR_FLG2 and HDR_FLG3 shown in Table 7 on page 45).

The IMS Connect HWSIMSO0 and HWSIMSO1 exits translate ASCII to EBCDIC and build the required message structure containing the OTMA headers for messages received *from* the client. This exit performs the translation from EBCDIC to ASCII and removes the OTMA headers for messages being transmitted *to* the client.

When you install the supplied sample user exits HWSSMPL0 and HWSSMPL1, you can modify either exit and link-edit it out as HWSIMSO0 or HWSIMSO1 to replace the copy supplied by IMS Connect, if you want to change any of the options (for example, translation, OTMA build, commit mode, sync level) in HWSIMSO0 or HWSIMSO1.

The user exits supplied by IMS Connect, HWSIMSO0 and HWSIMSO1, call the user-provided security exit, IMSLSECX, and pass a parameter list in register 1 to the security exit if it is defined in the exit. For the security parameter list structure, see “Security Exit” on page 74. For information about the security actions that HWSIMSO0 takes, see Appendix C, “HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 Security Actions,” on page 251.

**Input message structure passed to HWSIMSO0 and HWSIMSO1 exits**
The message structure (type 2) is defined in “Non-IMS Connector for Java Message Structure - Type 2” on page 75.

**Input message structure returned from HWSIMSO0 and HWSIMSO1 exits**
The message structure (type 3) is defined in “Non-IMS Connector for Java Message Structure - Type 3” on page 77.

**Output message passed to HWSIMSO0 and HWSIMSO1 exits**
The message structure is defined in “Non-IMS Connector for Java Message Structure” on page 80.

**Output message returned from HWSIMSO0 and HWSIMSO1 exits**
The message structure is defined in “Non-IMS Connector for Java Message Structure” on page 81.

## Sample User Message Exits (HWSSMPL0 and HWSSMPL1)

These sample user message exits perform the same functions as the IMS Connect HWSIMSO0 and HWSIMSO1 exits, which cannot be modified. However, you can modify the source code for the HWSSMPL0 or HWSSMPL1 exit, which are supplied with the IMS Connect installation. If you do not need to modify the user message exits HWSIMSO0 or HWSIMSO1, you can use either HWSSMPL0, HWSSMPL1, HWSIMSO0, or HWSIMSO1.

The COMMIT mode is set to “1,” and the SYNC level is set to “NONE.”These values can be overridden by supplying the COMMIT mode and/or sync level in the message prefix received from the client (see client message formats in IRM_FLG2 and IRM_FLG3 shown in Table 7 on page 45). Or, you can change the exit to set the COMMIT mode and SYNC level to the desired values.

The IMS Connect HWSSMPL0 and HWSSMPL1 exits translate ASCII to EBCDIC and build the required message structure containing the OTMA headers for messages received *from* the client. This exit performs the translation from EBCDIC to ASCII and removes the OTMA headers for messages being transmitted *to* the client.

These user exits call the user-provided security exit if one is defined to this exit and passes a parameter list in register 1. For the security parameter list structure, see “Security Exit” on page 74. For information about the security actions that HWSSMPL0 takes, see Appendix C, “HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 Security Actions,” on page 251.

**Input message structure passed to HWSSMPL0 or HWSSMPL1 exits**
The message structure is defined in “Non-IMS Connector for Java Message Structure - Type 2” on page 75.

**Input message structure returned from HWSSMPL0 or HWSSMPL1 exits**
The message structure is defined in “Non-IMS Connector for Java Message Structure - Type 3” on page 77.

**Output message passed to HWSSMPL0 or HWSSMPL1 exits**
The message structure is defined in “Non-IMS Connector for Java Message Structure” on page 80.

**Output message returned from HWSSMPL0 or HWSSMPL1 exits**
The message structure is defined in “Non-IMS Connector for Java Message Structure” on page 81.

## IMS Connector for Java User Message Exit (HWSJAVA0)

This IMS Connector for Java client exit is shipped with IMS Connect, and link-edited into the installation RESLIB. This exit does not perform a translation or build to the OTMA headers. Both the translation and insertion or deletion of the OTMA header is done by the IMS Connector for Java client server. HWSJAVA0 is supplied as source code and can be modified.

The COMMIT mode is set to “1,” and the SYNC level is set to “NONE.”

This user exit calls the user-provided security exit if one is defined to this exit and passes a parm list in register 1. For the security parm list structure, see “Security Exit” on page 74.

**Input message structure passed to HWSJAVA0 exit**
The message structure is defined in “IMS Connector for Java Message Structure - Type 1” on page 75.

**Input message structure returned from HWSJAVA0 exit**
The message structure is defined in “IMS Connector for Java Message Structure - Type 1” on page 77.

**Output message passed to HWSJAVA0 exit**
The message structure is defined in “IMS Connector for Java message structure” on page 79.

**Output messaged returned from HWSJAVA0 exit**
The message structure is defined in “IMS Connector for Java message structure” on page 79.

## IMS Connect IMSplex Message Exits (HWSCSLO0 and HWSCSLO1)

HWSCSLO0 and HWSCSLO1 IMSplex message exits are shipped with IMS Connect and link-edited into the IMS Connect RESLIB. You must use these exits for the IMSplex support which supports the IMS Control Center. The source code for HWSCLSO0 and HWSCSLO1 are not shipped and cannot be modified or replaced.

The COMMIT mode is set to 1, and the SYNC level is set to NONE. These values can be overridden by supplying either the COMMIT mode, the sync level, or the COMMIT mode and sync level in the message prefix received from the IMS Control Center client (see client message formats in IRM_F2 and IRM_F3 shown in Table 7 on page 45).

The IMS Connect HWSCSLO0 and HWSCSLO1 exits translate ASCII to EBCDIC and build the required message structure containing the required internal headers for messages received from the client. HWSCSLO0 exit performs the translation from EBCDIC to ASCII and removes the internal headers for messages being transmitted to the client. HWSCSLO1 exit does not perform any translation on the output data, but it does remove the internal headers for messages being transmitted to the client.

**Input message structure passed to HWSCSLO0 exit**
The input message structure is defined in "Non-IMS Connector for Java Message Structure - Type 2" on page 75.

**Output message returned from the HWSCSLO0 exit**
The output message is defined in "Non-IMS Connector for Java Message Structure" on page 81.

## Security Exit

You must provide a security exit (or use the TCP/IP exit, IMSLSECX) if any security checking is to be done by the message exit. Due to the many options available for security, and the fact that most installations have their own specific security method, no sample security exit is provided. The call to RACF is performed by IMS Connect if RACF parameters are provided in the OTMA header when the message exit returns the message.

The name of the security exit called by HWSSMPL0, HWSSMPL1, HWSIMSO0, HWSCSLO0, or HWSIMSO1 is *IMSLSECX*. You can change the name of the security exit called by HWSSMPL0 or HWSSMPL1, and supply and define it in the HWSSMPL0 and HWSSMPL1 message exit, by changing the **EXTRN IMSLSECX** to a name of your choice. If you require a different security exit in HWSSMPL0 or HWSSMPL1, you must provide the new security exit name. You must also provide the name of the security exit called by HWSJAVA0 and define it in the HWSJAVA0 message exit.

**Parameter list for user security exit:**

Following is the list and order of parameters being passed to the security exit, IMSLSECX. The order of the parameters is *fixed* for the exits supplied by IMS Connect: HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1.

- Address of fullword client's IP address
- Address of halfword client's port
- Address of 8-char string IMS transaction
- Address of halfword data type (data type setting: 0=ASCII, 1=EBCDIC)
- Address of fullword length of user data
- Address of user-supplied data
- Address of fullword set by security exit
- Address of fullword set by security exit
- Address of RACF user ID

  If blanks are returned (in the field pointed to) from the security exit, then the RACF fields in the OTMA security header are not set.

  The address points to a field containing blanks.
- Address of RACF group ID

  The address points to a field containing blanks.

## Message Structures

The following section describes the message structures for IMS Connector for Java and non-IMS Connector for Java messages.

## Input Message From Client and Passed to Exit

Input messages from the client consist of IMS Connector for Java and non-IMS Connector for Java message structures.

This section contains Product-Sensitive Programming Interface and Associated Guidance Information.

***IMS Connector for Java Message Structure - Type 1:*** Table 29 shows the input message format supported by IMS Connect from an IMS Connector for Java client.

*Table 29. Supported Message Format from an IMS Connector for Java Client*

| Field | Length | Meaning |
|---|---|---|
| llll | 4 bytes | Length of entire message, including llll field. |
| The IRM fields follow. | | |
| IRM_LL | 2 bytes | Length of IMS Connector for Java interface header, including LLZZ field. |
| IRM_RSV | 2 bytes | Reserved (set to binary zeros). |
| IRM_ID | 8 bytes | Char value of *HWSJAV*. |
| Reserved | 4 bytes | Reserved (set to binary zeros). |
| IRM_F5 | 1 byte | Binary value for input message type and resume type processing. |
| IRM_TIMER | 1 byte | Receive after ACK/RESUME TPIPE wait time. |
| IRM_SOCT | 1 byte | Receive after ACK/RESUME TPIPE wait time. |
| IRM_ES | 1 byte | Unicode encoding schema. |
| IRM_CLIENTID | 8 bytes | Char value of a unique client ID. |
| OTMA HDRs | 466 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description. |
| LL | 2 bytes | Length of data segment. |
| zz | 2 bytes | Reserved (set to binary zeros). |
| DATA | n bytes | User data with the tran code first. |
| OTMA CTL HDR | 20 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description. |
| LL | 2 bytes | Length of 2nd data segment. |
| zz | 2 bytes | Reserved (set to binary zeros). |
| DATA | n bytes | User data 2nd data segment (no tran code). |
| ... | | |
| OTMA CTL HDR | 20 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description. |
| LL | 2 bytes | Length of this and last data segment. |
| zz | 2 bytes | Reserved (set to binary zeros). |
| DATA | n bytes | User data with this data segment (no tran code). |

***Non-IMS Connector for Java Message Structure - Type 2:*** Table 30 on page 76 shows the input message format supported by IMS Connect from a non-IMS Connector for Java client.

*Table 30. Supported Message Format for Non-IMS Connector for Java Clients*

| Field | Length | Meaning |
|---|---|---|
| llll | 4 bytes | Length of entire message, including llll field |
| The IRM fields follow. | | |
| IRM_LEN | 2 bytes | Length of TCP/IP interface header |
| IRM_RSV | 2 bytes | Reserved (set to binary zeros) |
| IRM_ID | 8 bytes | Char value of *IRMREQ* (for HWSIMSO0) Char value of *IRMRE1* (for HWSIMSO1) Char value of *SAMPLE* (for HWSSMPL0) Char value of *SAMPL1* (for HWSSMPL1) |
| IRM_RES | 4 bytes | Reserved for future use. |
| IRM_F5 | 1 byte | Binary value for input message type and resume type processing |
| IRM_TIMER | 1 byte | Receive after ACK/RESUME TPIPE wait time |
| IRM_SOCT | 1 byte | Socket type |
| IRM_ES | 1 byte | Unicode encoding schema |
| IRM_CLIENTID | 8 bytes | Char value of a unique client ID |
| The following definition is for use with the HWSIMSO0 and HWSSMPL0 exits. The user installation can provide its own exit, and structure the following items as required by the user exit. The following items should be considered. This example lists only some of the items you can use. You might want to include fields that are used only by the user exit or other items that can be passed in the OTMA headers, such as the MID name. | | |
| IRM_F1 | 1 byte | Binary MFS and Unicode flag |
| IRM_F2 | 1 byte | Binary COMMIT MODE flag |
| IRM_F3 | 1 byte | Binary SYNC LEVEL flag |
| IRM_F4 | 1 byte | Char value conversation byte |
| IRM_TRNCOD | 8 bytes | Char value for user transaction code |
| IRM_IMSDESTID | 8 bytes | Char value for Datastore ID |
| IRM_LTERM | 8 bytes | Char value for LTERM override name |
| IRM_RACF_USERID | 8 bytes | Char value for RACF user ID |
| IRM_RACF_GRNAME | 8 bytes | Char value for RACF group name |
| IRM_RACF_PW | 8 bytes | RACF PassTicket/password |
| The following is the data structure for all non-IMS Connector for Java clients. An IMS command input can only contain a single LL ZZ DATA followed by EOM. | | |
| LL | 2 bytes | Length of data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data with the tran code first |
| LL | 2 bytes | Length of 2nd data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data 2nd data segment (no tran code) |
| ... | | |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data segment (no tran code) |

Table 30. Supported Message Format for Non-IMS Connector for Java Clients (continued)

| Field | Length | Meaning |
|---|---|---|
| LL | 2 bytes | End of message (set to binary 0000 0000 0000 0100) |
| zz | 2 bytes | Reserved (set to binary zeros) |

## Input Message Returned From Message Exit

Input messages from the message exit consist of IMS Connector for Java and non-IMS Connector for Java message structures.

***IMS Connector for Java Message Structure - Type 1:*** The IMS Connector for Java exit output message format that is supported by IMS Connect is the same message format of the input message. See "IMS Connector for Java Message Structure - Type 1" on page 75 for the message format.

The total length of the message can be 10,000,000 bytes. The length of each segment (from the BPE header to the next BPE header) within the message can be a maximum of 32 KB, excluding the BPE and OTMA headers.

***Non-IMS Connector for Java Message Structure - Type 3:*** Table 31 shows the output message format supported by IMS Connect from the supplied HWSIMSO0 and HWSSMPL0 exits (non-IMS Connector for Java client exits). The table provides information about the field, length, and meaning. Variable length OTMA headers are supported, and therefore, the OTMA header length can be other than 466 bytes. The following example contains 466 bytes as used by the supplied exits.

Table 31. Supported Output Message Format for HWSIMSO0 and HWSSMPL0 Exits

| Field | Length | Meaning |
|---|---|---|
| BPE HEADER | 64 bytes | See BPE header definition that follows this table |
| OTMA HDRs | 466 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of first data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data with the tran code first |
| Repeat of ll,zz,DATA | | |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data (no tran code) |
| yy | 2 bytes | Binary value of zero |
| BPE HEADER | 64 bytes | See BPE header definition that follows this table |
| OTMA CTL HDR | 32 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of this data segment |

*Table 31. Supported Output Message Format for HWSIMSO0 and HWSSMPL0 Exits (continued)*

| Field | Length | Meaning |
|---|---|---|
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data segment (no tran code) |
| Repeat of LL,zz,DATA | | |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data (no tran code) |
| ... | | |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data (no tran code) |
| yy | 2 bytes | Binary value of zero |
| BPE HEADER | 64 bytes | See BPE header definition that follows this table |
| OTMA CTL HDR | 32 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data (no tran) |
| ... | | |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data (no tran code) |
| yy | 2 bytes | Binary value of zero |

**Restriction**: The length of data from one BPE header to the next BPE header cannot exceed 32K, excluding the BPE header and the OTMA header.

**BPE header format:**Table 32 on page 79 describes length and meaning of the fields in the BPE header format.

**Restriction**: Only the chain pointer field is modified by the message exit to chain the BPE headers together with the last BPE chain pointer set to binary zeros. The other fields in the BPE header ***MUST NOT BE MODIFIED BY THE EXIT***.

**Important:** The following table contains Product-Sensitive Programming Interface and Associated Guidance Information.

Table 32. BPE Header Format

| Field | Length | Meaning |
|---|---|---|
| llll | 4 bytes | Length of field of entire buffer |
| CHAIN PTR | 4 bytes | Chain pointer to next BPE header |
| STORAGE TYPE | 8 bytes | Storage type |
| TYPE ACCESS | 4 bytes | Type access |
| SUBPOOL | 1 byte | Subpool |
| RESV | 43 bytes | Reserved |

## Output Message From IMS Connect to Client

Output messages from IMS Connect to the client consist of the IMS Connector for Java and non-IMS Connector for Java message structures.

***IMS Connector for Java message structure:*** Table 33 shows the message format from IMS Connect to user message exit, HWSJAVA0, and the message format returned to IMS Connect from the user message exit, HWSJAVA0. The messages passed to HWSJAVA0 and returned by HWSJAVA0 are the same format. The table provides information about the length and meaning of the fields in the output message.

**Important:** The following table contains Product-Sensitive Programming Interface and Associated Guidance Information.

Table 33. Output Message Format from IMS Connect to the Client

| Field | Length | Meaning |
|---|---|---|
| llll | 4 bytes | Total message length |
| Id | 8 bytes | *HWSJAV* |
| OTMA HDRs | 466 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| OTMA CTL HDR | 20 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| ... | | |
| OTMA CTL HDR | 20 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of this segment |
| zz | 2 bytes | Reserved (set to binary zeros) |

*Table 33. Output Message Format from IMS Connect to the Client (continued)*

| Field | Length | Meaning |
|---|---|---|
| DATA | n bytes | User data |
| OTMA CTL HDR | 32 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| ... | | |
| LL | 2 bytes | Length of this data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |

***Non-IMS Connector for Java Message Structure:*** Table 34 shows the message format from IMS Connect to the exit for the client.

*Table 34. Output Message Format from IMS Connect to the Exit*

| Field | Length | Meaning |
|---|---|---|
| OTMA HDRs | Length of total OTMA headers. | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| OTMA CTL HDR | 20 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of this segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| ... | | |
| OTMA CTL HDR | 20 bytes | See OTMA DSECT (HWSOMPFX) in GENLIB for description |
| LL | 2 bytes | Length of this segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |

## Output Message From Message Exit

Output messages from the message exit consist of the non-IMS Connector for Java message structures.

***Non-IMS Connector for Java Message Structure:*** The non-IMS Connector for Java message structure can consist of one or more TCP/IP message structures. These TCP/IP message structures are described in this section.

**RMM - Request Mod Message**
Returned as the first structure of an output message if the MFS mod name is requested and the data output is present. (This does not apply to IMS command output.)

Table 35 shows the output message format of the Request Mod Message built by the user message exits, HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1. The table includes the field name, field length, and field meaning.

*Table 35. Request Mod Message Output Message Format*

| Field | Length | Meaning |
|---|---|---|
| LL | 2 bytes | Length of RMM message |
| zz | 2 bytes | Reserved (set to binary zeros) |
| ID | 8 bytes | Char value of *REQMOD* |
| MOD | 8 bytes | Char value of the requested MFS MOD name |

**CSM - Complete Status Message**
Returned as the last structure of an output message if the input message is processed successfully. (This does not apply to IMS command output.)

Table 36 shows the output message format of the Complete Status Message. The table includes the field name, field length, and field meaning.

*Table 36. Complete Status Message Output Message Format*

| Field | Length | Meaning |
|---|---|---|
| CSM_LEN | 2 bytes | Length of CSM message |
| CSM_FLG1 | 1 byte | FLAG BYTE ONE X'80' asynchronous message queued in IMS. X'40' conversational output message. X'20' ACK/NAK required. |
| Reserved | 1 byte | Reserved (set to binary zeros) |
| CSM_ID | 8 bytes | Char value of *CSMOKY* |

**RSM - Request Status Message**
Returned as the only structure of an output message if IMS Connect or the message exit determined an error occurred. (This is valid for IMS command output.)

Table 37 shows the output message format of the Request Status Message for an error condition. The table includes the field name, field length, field meaning.

*Table 37. Request Status Message Output Message Format*

| Field | Length | Meaning |
|---|---|---|
| RSM_LEN | 2 bytes | Length of RSM message |

Table 37. Request Status Message Output Message Format (continued)

| Field | Length | Meaning |
|---|---|---|
| RSM_FLG1 | 1 byte | FLAG BYTE ONE X'80' asynchronous message queued in IMS. X'40' conversational output message. X'20' ACK/NAK required. |
| Reserved | 1 byte | Reserved (set to binary zeros) |
| RSM_ID | 8 bytes | Char value of *REQSTS* |
| RSM_RETCOD | 4 bytes | Return code |
| RSM_RSNCOD | 4 bytes | Reason code |

The output message from the message exit that is sent to non-IMS Connector for Java clients is in one of the following formats:

- MFS MOD name request, data, and CSM is being sent. (This does not apply to IMS command output.) Table 38 shows one of the formats of output message that is sent to non-IMS Connector for Java clients. The table includes the field names, field lengths, and field meaning.

Table 38. Output Message Format Containing RMM, DATA, and CSM

| Field | Length | Meaning |
|---|---|---|
| RMM header (optional) | 20 bytes | Request Mod message, contains mod name if requested |
| LL | 2 bytes | Length of data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| LL | 2 bytes | Length of 2nd data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data 2nd data segment |
| ... | | |
| LL | 2 bytes | Length of nth segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data nth data segment |
| CSM | 12 bytes | Complete status message |

- MFS MOD name is not requested and only data and CSM is being sent. (This does not apply to IMS command output.) Table 39 shows the other format of output message that is sent to non-IMS Connector for Java clients. The table includes the field names, field lengths, and field meaning.

Table 39. Output Message Format Containing Output Data and CSM Only

| Field | Length | Meaning |
|---|---|---|
| LL | 2 bytes | Length of data segment |

*Table 39. Output Message Format Containing Output Data and CSM Only (continued)*

| Field | Length | Meaning |
|---|---|---|
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data |
| LL | 2 bytes | Length of 2nd data segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data 2nd data segment |
| ... | | |
| LL | 2 bytes | Length of nth segment |
| zz | 2 bytes | Reserved (set to binary zeros) |
| DATA | n bytes | User data nth data segment |
| CSM | 12 bytes | Complete status message |

The output message sent to an IMS Control Center client is shown in Table 40.

**IMS Command Output**
Returned as the only data response structure of an output command response.

*Table 40. Output Message Format Sent to the IMS Control Center*

| Field | Length | Meaning |
|---|---|---|
| LLLL | 4 bytes | Length of output |
| DATA | n bytes | IMS command output |

## Macros

IMS Connect supports six macros: HWSEXPRM, HWSOMPFX, HWSIMSCB, HWSIMSEA, HWSXIB, and HWSXIBDS.

**HWSEXPRM**
This macro provides the mapping for the parameter list that is passed to the user exit on each subroutine call. A copy of this macro is in AHWSSRC. To see the structure, assemble the macro.

**HWSOMPFX**
This macro maps the OTMA message prefix format to the output buffer that the user exit returns on each READ subroutine call and the input buffer that is passed to the user exit on each XMIT subroutine call. A copy of this macro is in AHWSSRC. To see the structure, assemble the macro.

**HWSIMSCB**
This macro maps the IMS request messages and BPE header formats used by HWSSMPL0. A copy of this macro is in AHWSSRC. To see the structure, assemble the macro.

**HWSIMSEA**
This macro maps the storage area used by HWSSMPL0 and HWSSMPL1. A copy of this macro is in AHWSSRC. To see the structure, assemble the macro.

**HWSXIB**
This macro maps the exit interface block used by HWSUINIT. A copy of this macro is in AHWSSRC. To see the structure, assemble the macro.

**HWSXIBDS**
This macro maps the entry in the exit interface block datastore list (used by HWSUINIT). The list contains the datastore name, the datastore status, and a user field. A copy of this macro is in AHWSSRC. To see the structure, assemble the macro.

# Chapter 6. IMS Connect DRU Exit for Asynchronous Output Support

An OTMA DRU (destination resolution) exit is required to support asynchronous output that is generated by an IMS application that does an insert (ISRT) to an alternate PCB (program communication block). IMS Connect provides a sample OTMA DRU exit named HWSYDRU0. You can either modify the HWSYDRU0 exit to work with your installation, or provide your own DRU exit.

**In this chapter**:

**Related Reading:** For more information on the OTMA DRU exit, see the *IMS Customization Guide*.

## How IMS Connect Communicates with the DRU Exit

OTMA allows transaction pipe names (TPIPEs) to be the same as an IMS LTERM name. In IMS Connect, the LTERM name is analogous to the unique CLIENTID name. To clarify whether a destination is for IMS Connect (via OTMA), IMS provides OTMA exit routines that can specify where IMS should look to resolve the destination names. In this case, the IMS needs to look at the IMS Connect CLIENTIDs. The DRU exit cannot change the actual destination name. Determining the destination for an OTMA (IMS Connect client) message requires two phases.

1. The prerouting exit routine (DFSYPRX0) is called to determine the initial destination for the output.

   The exit routine can determine whether the message should be directed to OTMA (IMS Connect clients) or to IMS TM for processing. The exit routine cannot determine the final destination.
2. The DRU exit routine (for example, the IMS Connect supplied exit HWSYDRU0) is called to determine the final destination for the output.

   Each OTMA client can specify a separate DRU exit routine. In other words, each OTMA client can specify a single DRU exit for each copy of IMS Connect that is connected to a given datastore (IMS). This means that one IMS Connect can have the same or a different DRU exit for each of the datastore definitions in the IMS Connect configuration file.

## How to Use the HWSYDRU0 Exit

HWSYDRU0, the IMS Connect supplied OTMA DRU exit, provides only a sample of what the DRU exit can do. You can use this exit only under one of the following conditions:

- The IMS Connect CLIENTIDs are named CLIENT01 through CLIENT09 and they all belong to the same member name.
- The non-IMS Connect CLIENTIDs are as follows:
  - TPIPE001 through TPIPE099 all belong to member MEMBER0
  - TPIPE100 through TPIPE199 all belong to member MEMBER1
  - TPIPE200 through TPIPE299 all belong to member MEMBER2
  - TPIPE300 through TPIPE399 all belong to member MEMBER3
  - TPIPE400 through TPIPE499 all belong to member MEMBER4

- – TPIPE500 through TPIPE599 all belong to member MEMBER5
- – TPIPE600 through TPIPE699 all belong to member MEMBER6
- – TPIPE700 through TPIPE799 all belong to member MEMBER7
- – TPIPE800 through TPIPE899 all belong to member MEMBER8
- – TPIPE900 through TPIPE999 all belong to member MEMBER9

The HWSYDRU0 exit is only an example, and when you use it, the following sequence of events will occur:

1. The prerouting exit (DFSYPRX0) sets up addressability to the parameters that are passed to the HWSYDRU0 exit.
2. The output member name in the output parameter list is set to blanks.
3. HWSYDRU0 determines the action to take based on whether the name in the input destination parameter (that is, the destination where the message is to be sent) is an IMS LTERM or an IMS Connect destination. After HWSYDRU0 makes this determination, it takes a course of action, and sets the contents of register 15 on exit.
4. If an IMS application was initiated by a non-IMS Connect client, then the YDRU exit must build the OTMA user data.
5. If the YDRU exit places the character string, ICONNECT, into the OTMA user data header field, OMUSR_PORTID, (whether built by YDRU or passed to YDRU) then IMS Connect will determine the correct PORTID to be used for the selected output client ID.

Table 41 describes the register settings and the action taken for the specific return code.

*Table 41. Register Settings and HWSYDRU Actions*

| Register Settings | HWSYDRU Actions |
|---|---|
| Register 15 = X'00' | • The input destination name is an IMS Connect client name and the Member name for the destination is the same as the Member name for the origin.<br>• No changes made to the output parameters. |
| Register 15 = X'04' | • LTERM exists in IMS (LEGACY), is not an IMS Connect client.<br>• No changes made to the output parameters. |
| Register 15 = X'08' | • The input destination name is an IMS Connect client name, and the Member name for the destination is a different name from the Member name for the origin.<br>• The output member name in the output parameters is set to the new Member name. |
| Register 15 = X'0C' | The input destination name is not an LTERM for IMS, and IMS Connect does not know the client name. |

# Chapter 7. IMS Connect User Initialization Exit Support

IMS Connect provides a user initialization exit routine, HWSUINIT. This routine enables you to perform customized initialization tasks during IMS Connect startup and/or customized termination tasks during IMS Connect shutdown.

For example, you can modify the HWSUINIT routine to display a specific message when IMS Connect starts up or shuts down.

The HWSUINIT routine contains two user control blocks that enable further customization: XIB and XIBDS. The XIB control block can be used to store any data that you want. The XIBDS control block keeps track of the status of the IMS Connect datastores. All of the IMS Connect user message exits can access both the XIB and XIBDS user control blocks.

For example, you can modify HWSUINIT to load a specific table when IMS Connect starts up; then, store the table address into the XIB control block area. Once the IMS Connect user message exits get control, they access that table and perform their customized processing. When IMS Connect shuts down, you can modify HWSUINIT to unload the updated table.

**Important**: The HWSUINIT user initialization exit routine that comes with IMS Connect does not do any processing. Modify HWSUINIT only if you want to use it.

**Related Reading**: “Macros” on page 83 describes the XIB and XIBDS definitions.

**In this chapter:**

## How IMS Connect Communicates with HWSUINIT

HWSUINIT contains two subroutines: INIT and TERM. When IMS Connect starts, HWSUINIT loads and gives control to the INIT subroutine. When IMS Connect shuts down, HWSUINIT gives control to the TERM subroutine.

HWSUINIT contains two of its own user control blocks: XIB and XIBDS. The HWSXIB and HWSXIBDS DSECTs map the XIB and XIBDS user control blocks. The message exit routines in the INIT, READ, XMIT, TERM, and EXER subroutines can also use the XIB and XIBDS user control blocks. The XIB user control block contains a fixed length header section and a variable length user area.

**Restriction**: You cannot modify the fixed header section. You can only modify the user area.

You specify the size of the XIB control block user area, in full words, with the *xibarea* parameter (in the HWS statement of the IMS Connect configuration file). The default value is 20; the maximum value is 500. If you do not specify a value for the *xibarea* parameter, or you specify a value outside of the 20 to 500 range, IMS Connect uses the default value of 20.

The XIBDS user control block represents an entry in a list of datastores that are defined in the configuration file. The second word in the fixed header area of the

XIB user control block points to the datastore list. The XIBDS user control block is 16 bytes long. Each datastore list entry contains the datastore name, the datastore status (active or inactive), a flag byte, and a 4 byte field that you can use to store any kind of data. The last entry is indicated by a value of X'80' (hexadecimal) in the flag byte. The number of entries in the list is equal to the number of datastores defined in the IMS Connect configuration file.

Because the XIBDS user control block keeps track of *all* IMS Connect datastore statuses, you can enable any user message exit to take action based on the status of one or more of the IMS Connect datastores. For example, before a user message exit passes a client message to an IMS Connect datastore for processing, you could have the user message exit query the XIBDS control block area for the target datastore’s status. If the target datastore is not active, you could enable the user message exit to switch to an active datastore by modifying the datastore name in the message header. Refer to “Macros” on page 83 for the XIBDS definition.

When the HWSUINIT routine takes control, it saves the contents of the registers and restores them when returning to the caller. IMS Connect provides a 1 KB buffer in the parmlist to be used for this purpose.

## Register Contents on HWSUINIT Entry

Table 42 lists the contents of each register on the HWSUINIT entry.

*Table 42. Register Contents on HWSUINIT Entry*

| Register | Contents |
|---|---|
| 1 | Pointer to a parmlist: <br>• +0 — XIB address <br>• +4 — Function to perform (INIT or TERM) <br>• +8 — 1 KB buffer for exit to use |
| 14 | Return address of IMS Connect. |
| 15 | Entry point address to HWSUINIT. |

## Register Contents on HWSUINIT Exit

Table 43 lists the contents of each register on the HWSUINIT exit.

*Table 43. Register Contents on HWSUINIT Exit*

| Register | Contents |
|---|---|
| 0–14 | Restored. |
| 15 | 0 — completed successfully. 1 to 7 — warning, but IMS Connect initialization continues. 8 or higher — force IMS Connect termination. |

# Chapter 8. IMS Connect IMSplex Support

This chapter describes how IMS Connect sends and receives OM commands and response string messages to and from the IMS Control Center client, which is delivered as part of the DB2 Control Center, to an IMS Operations Manager within an IMSplex using IMS SCI. It also provides detailed information about the environment requirements and how to set up IMS Connect to support OM.

**In this chapter:**

## IMSplex Support

The IMS Connect support, called IMSplex, allows the IMS Control Center on the DB2 Control Center client to access OM. The IMSplex support accesses OM through the IMS Structure Call Interface (SCI). The IMSplex statement in the IMS Connect configuration file (HWSCFGxx) defines IMS Connect for IMSplex support. If the IMSplex statement is omitted, then IMSplex support is not available.

IMSplex support, sends IMS command string messages directly for a client (for example, the IMS Control Center supplied TCP/IP client) to a selected OM within an IMSplex. One or more IMSplex can be defined to IMS Connect to receive DB2 Control Center client command messages. SCI is used to communicate between IMS Connect and the IMSplex. To gain access to the selected OM, you can define the same IMS system as both a datastore and as an IMSplex.

The same security method (authentication of the userid, groupid, and password) used for accessing datastores also applies to IMSplex support. An IMS Connect command message exit performs similar functions as an IMS Connect user message exit.

There is a separate and specific message exit (HWSCSLO0) that is defined for IMSplex support. This exit is similar to the user message exits provided by IMS Connect for client access to the datastore. The IMSplex message exit is designed to be used only by the IMS Control Center client and cannot be used by any client that sends messages to a datastore. The difference between the two message exits is that the IMS Connect command message exit processes only the IMS Control Center command string messages.

## IMSplex Support Environment

IMS Connect requires that the following environments be running to communicate with OM:

- IMS 8.1 or later (in the same MVS image or a different MVS image on the same sysplex)
- IMS 8.1 Operations Manager or later (in the same MVS image or a different MVS image on the same sysplex)
- IMS 8.1 Structure Call Interface (SCI) (in the same MVS image or a different MVS image on the same sysplex)

See *IMS Version 8: Common Service Layer Guide and Reference* for bringing up the IMS, SCI, and OM address spaces.

IMS Connect can be brought up before or after IMS, SCI, OM, and RM. During IMS Connect initialization, connection to SCI is made. IMS Connect attempts to connect to SCI for 30 minutes. If SCI connection is not made, then an `OPENIP` command will need to be issued to connect to the SCI after the SCI has been initialized. If the SCI terminates normally or abnormally, IMS Connect will automatically reconnect to the SCI when the SCI is restarted.

## Installing IMSplex Support

IMSplex support requires that the installation process be performed in the following order:

**IMS Connection Configuration File**

1. Add the IMSplex statement.
2. Add the HWSCSLO0 exit to the TCPIP statement EXIT= parameter.

**IMS Connect BPE Configuration File**

1. Add OMDR and HWSO statements, if the IMS Connect trace entries are listed separately.

**IMS Control Center**

1. Identify the IMS Connect HWS ID= value.
2. Identify the IMS Connect IMSPLEX tmember= value.

**IMS Connect Commands Introduced by IMSplex Support**

- `STOPIP`
- `OPENIP`
- `VIEWIP`

**IMS Connect IMSplex Support Requirements**

- IMS Operations Manager (OM)

  OM requires IMS to be running to provide the OM command capability. IMS Connect IMSplex support does not require IMS to be running; however, none of the commands will be processed.

  **Recommendation by IMS**: Follow, in order, the start procedures:

  1. Start SCI
  2. Start OM
  3. Start RM
  4. Start IMS

  You can start IMS Connect before, during, or after the steps listed above. IMS Connect will attempt to connect to SCI for 30 minutes, and if SCI is not brought up or the connection attempt fails, you must issue an `OPENIP` *`imsplex_name`* command.
- IMS Structure Call Interface (SCI)
- TCP/IP
- IMS 8.1 must be installed to use the IMS Connect IMSplex support
- IMS 8.1 (or higher) RESLIB must be added to the STEPLIB

  The IMS RESLIB is required to be able to access the SCI for IMS Connect IMSplex support.

# Chapter 9. IMS Connect Two-Phase Commit Support

IMS Connect supports two phase-commit which allows IMS transactions to participate in two-phase-commit transactions that are coordinated by RRS or an external coordinator (for example, IBM WebSphere Application Server). The external coordinator must use IMS Connector for Java as the resource adapter. Together, IMS Connector for Java and IMS Connect handle the data flow for two-phase-commit processing.

This chapter provides an overview of two-phase commit and some key scenarios that IMS Connect supports.

**In this chapter:**

## Overview of Two-Phase Commit Protocol

Two-phase commit protocol is comprised of a set of actions that ensure a transaction involving multiple databases does not produce unsynchronized updates. Two-phase commit provides a way for a series of database interactions that are grouped into a single transaction to be completed or rolled back as one transaction.

At the beginning of a two-phase commit transaction, an ID is generated and used by an external transaction coordinator or resource manager to monitor and make modifications to the state of the transaction. Each interaction within the two-phase transaction is temporarily executed upon the associated databases. The results of the interactions are then sent back to the application for processing. When the two-phase commit transaction scope is met, a prepare call is sent to each database that was accessed. The prepare call verifies that each database has made the appropriate resources accessible to perform the interactions attempted within the scope of the two-phase commit transaction. Upon receiving verification that each database is ready to complete the interactions, a commit call is then sent to each database.

At any point in time, prior to sending the commit call, the two-phase commit transaction can be rolled back. If the transactions is rolled back, a rollback call is sent to each database involved in the transaction and the temporary changes are removed or discarded. If any database failures occur during the commit phase, the external coordinator or resource manager indicates that a heuristic situation may have been reached. The external coordinator either ignores or forgets the database modifications that have already been committed and attempts to repeat or recover the calls that failed until the calls are either successful or manually removed from the external coordinator's or resource manager's logs.

## Distributed Two-Phase Commit Support

Distributed two-phase commit protocol uses TCP/IP to communicate transactions between various platforms (for example, Windows®, AIX®, Solaris, Linux™). A distributed TCP/IP transaction normally involves the following components:

- An application component
- An application server
- A resource adapter
- A resource manager
- A transaction manager
- An enterprise information system (EIS)

In distributed two-phase commit protocol, a client issues a transaction that is deployed by the application server. The application server acts as an external transaction manager (external coordinator) to manage transactions across one or more resource managers. To access the resource manager of an enterprise information system, the external coordinator must use a resource adapter. IMS Connector for Java (a resource adapter) accesses the resource manager (IMS) through IMS Connect using TCP/IP.

IMS does not support X/Open XA protocol and only supports RRS. To participate in two-phase-commit processing, IMS uses RRS (Resource Recovery Service) on z/OS. As a result, IMS Connect communicates with RRS and passes to RRS the transaction context from IMS Connector for Java. In turn, RRS as the syncpoint coordinator coordinates the changes so that all or no updates are made to IMS. In RRS, the set of changes that are made or not made within the transaction scope is called a unit of recovery (UR).

IMS Connect plays dual roles in two-phase-commit processing. IMS Connect, acts as an extension to RRS (the syncpoint manager) and is considered the server distributed syncpoint resource manager (SDSRM). As the SDSRM, IMS Connect allows RRS to communicate with other syncpoint managers as needed to ensure coordination of the distributed resources the application accesses. IMS Connect also is the communication resource manager (CRM). As the CRM, IMS Connect controls access to distributed resources by allowing an application component to communicate with other application components and resource managers that may be on different systems. Also, as the CRM, IMS Connect® assists in processing a syncpoint event and communicates the events to distributed syncpoint managers.

Distributed two-phase-commit processing can be broken down into two types of transactions: global transaction which uses two-phase commit optimization or global transaction which uses the one-phase commit protocol.

## Global (XA) transaction with TCP/IP

A global (XA) transaction is controlled and coordinated by an external transaction manager (external coordinator) to a resource manager. The transaction normally requires coordination across multiple resource managers that may reside on different platforms.

To access an enterprise information system, the external coordinator sends an XID, which is defined by the X/Open XA standard, to a resource adapter. In addition to the length and FormatID fields, an XID has two other parts: the global transaction identifier (GTRID) and the branch qualifier (BQUAL). Because IMS does not support X/Open XA protocol, IMS Connector for Java uses the LocalTransaction and

XAResources interfaces to participate in transactions coordinated by the external coordinator to communicate with IMS Connect. IMS Connect maintains the XID and associates it with a work context token and an IMS name. IMS Connect then passes the context token to RRS.

IMS Connect sends the transaction output back to IMS Connector for Java which returns the output data to the client. Upon sending the output message to IMS Connector for Java successfully, IMS Connect sends an ACK to IMS to acknowledge the message. After making requests to IMS, the application component indicates to IMS Connector for Java that it is ready to commit the changes. At this point IMS Connector for Java sends a prepare signal to IMS Connect. IMS Connect, in turn, tells RRS to initiate the prepare phase. If the IMS resource manager is prepared to commit, RRS collects the prepare to commit confirmation from the resource manager and sends the results to IMS Connect. IMS Connect will then send a *request to commit* signal to IMS Connector for Java to request committing the changes.

When IMS Connect for Java receives the request to commit signal, it tells the external coordinator that the resources on the IMS system can be committed. The transaction manager determines the overall results. If all the resource managers can commit, the transaction manager hardens the commit decision and will drive IMS Connector for Java to commit the change. IMS Connector for Java sends a commit signal to IMS Connect and IMS Connect tells RRS that the overall decision is to commit all resources. RRS tells IMS to commit the changes. After IMS commits the changes, RRS then returns to IMS Connect with the information that the local resources have been committed. IMS Connect tells RRS to delete its log records.

Figure 6 on page 94 illustrates the flow of a distributed two-phase commit global transaction. The transaction involves two IMSs. IMS Connect, RRS, and IMS must all be on the same MVS image.

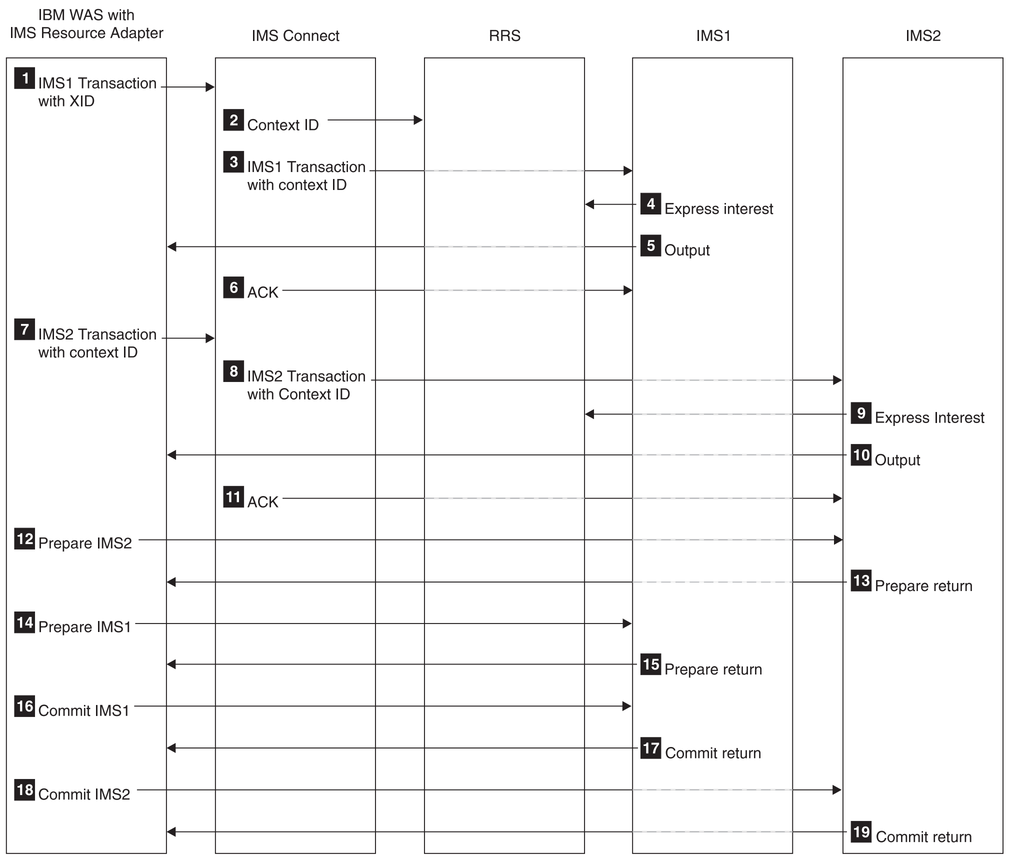


*Figure 6. Distributed Two-Phase Commit Global Transaction Client Flow*

## Global Transaction with One-Phase Commit Optimization

If only one resource manager is registered in a transaction that is making changes to shared resources, the transaction manager can perform one-phase-commit optimization. An external coordinator is not required. The transaction manager can send the phase two *commit request* directly to the resource manager to commit the changes. IMS Connect does not have to go through phase one, *prepare to commit* of the two-phase commit protocol and can go directly to phase two, *commit request*.

Figure 7 on page 95 illustrates the flow for a distributed one-phase commit global transaction. IMS Connect, RRS, and IMS must all be on the same MVS image.

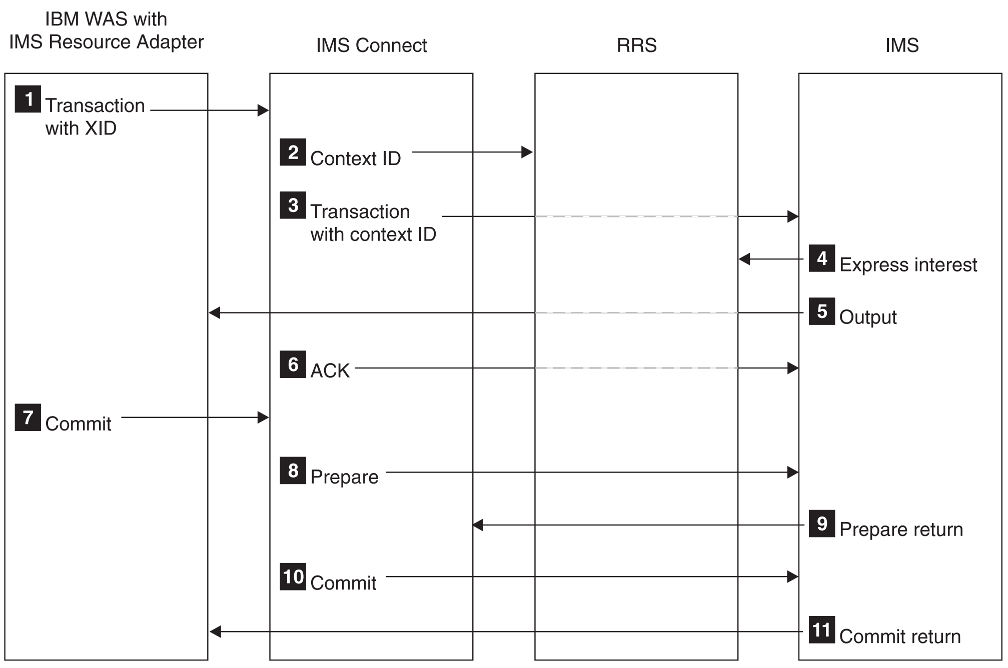


*Figure 7. Distributed One-Phase Commit Optimization Client Flow*

## Local Option Two-Phase Commit Support

IMS Connect also supports two-phase commit through Local Option. To enable two-phase commit flow, two fields in the OTMA prefix must be set correctly before the transaction is sent to IMS Connect. First, the OMHDRSYN flag must be set to X'02' (for example, OMHDRSL2) to indicate SyncLevel Syncpt. Second, the OMHDRCID field must be set to a 16-byte RRS context token. Then the ensuing flow is as follows:

1. When IMS Connect detects that the incoming transaction contains the context token and SyncLevel Syncpt, it calls RRS to switch the context off and pass the transaction to IMS.
2. IMS receives the transaction from IMS Connect and schedules the transaction for processing. Because the context token and SyncLevel Syncpt are set, the transaction output will be delivered first before sync point processing is started. When the client receives the transaction output, it is expected to send an acknowledgement to IMS Connect.
3. When the client decides to start two-phase commit processing after it has sent the positive acknowledgement to IMS Connect, it needs to inform RRS of its decision. When RRS is informed of the decision, IMS will be driven for its own two-phase commit processing. As a result, the transaction will commit or backout based on the final decision from RRS.

Figure 8 on page 96 illustrates the two-phase commit flow using the local option provided by IMS Connect. The IBM WebSphere Application Server for z/OS, IMS Connect, RRS, and IMS must all be on the same MVS image.

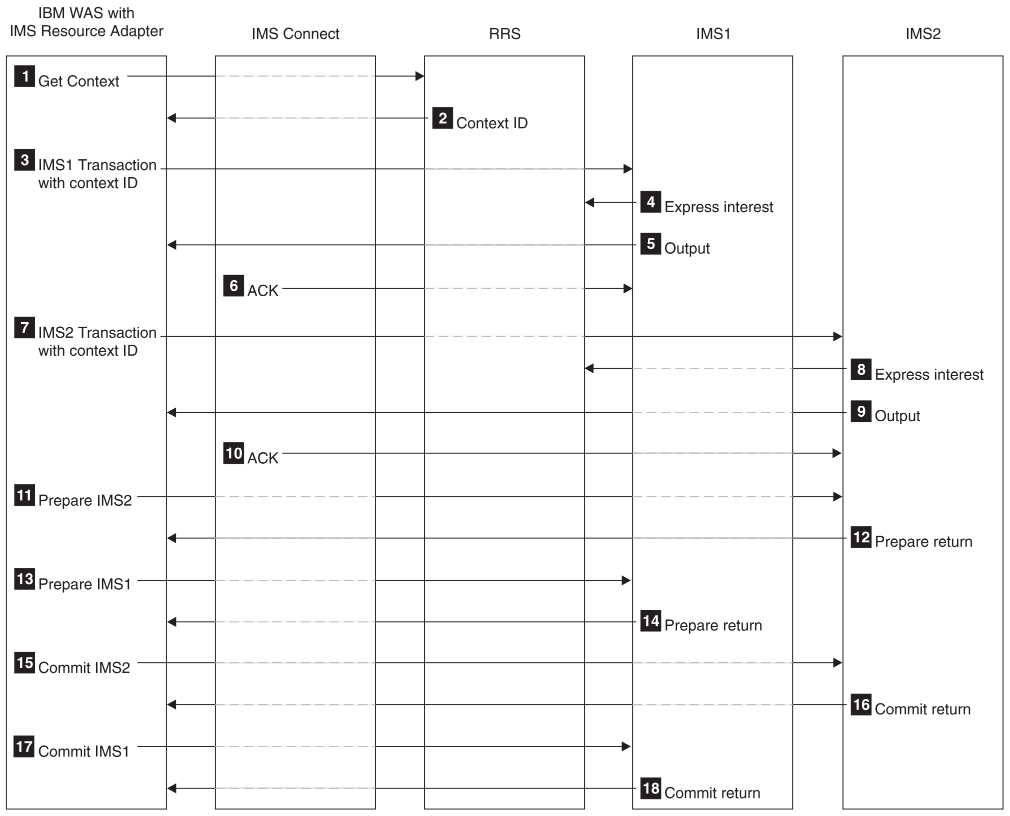


*Figure 8. Two-Phase Commit Flow for Local Option*

# Part 2. Programmer's Guide and Reference

# Chapter 10. Protocols

This chapter describes the transaction protocols, which are as follows:

- Conversational support
- Send only
- Resume Tpipe/Receive for asynchronous output
- Socket connections
- Asynchronous output support

**In this chapter:**

## Transaction Restrictions and Limitations

The following is a list of restrictions and limitations of specific transactions:

- IMS Fast Path, conversational, and non-recoverable transactions must be issued using commit mode 1. This is a restriction of IMS OTMA.
- Non-response transactions can be sent to IMS Connect using the SENDONLY option and must be issued using commit mode 0 on a transaction or persistent socket.

## Conversational Support

A conversational program is a message processing program (MPP) that processes transactions made up of several steps. The MPP does not process the entire transaction at once.

The conversational support for IMS Connect includes having conversational transactions that let you retain uninterrupted connection (continuity) for messages coming from a given client. Typically, a conversation is terminated when the message is sent and dequeued and the application program has placed blanks in the SPA, or the conversation is terminated when a COMMIT CONFIRMED messaged is received from the client. For conversational support for IMS Connect, conversations require a send-then-commit mode and are nonrecoverable.

This section describes and shows various conversational protocols as used with IMS Versions 6 and 7. For more information about conversational protocols that are used with IMS Version 5, see Table 46 on page 124.

## OTMA Conversational Protocols

### Send-then-commit, sync level=none

The send-then-commit flow (see Figure 9) sends IMS output before IMS completes synchronization-point (hereafter referred to as sync-point) processing. To use the send-then-commit flow, specify commit Mode 1 in the state-data section of the message prefix.

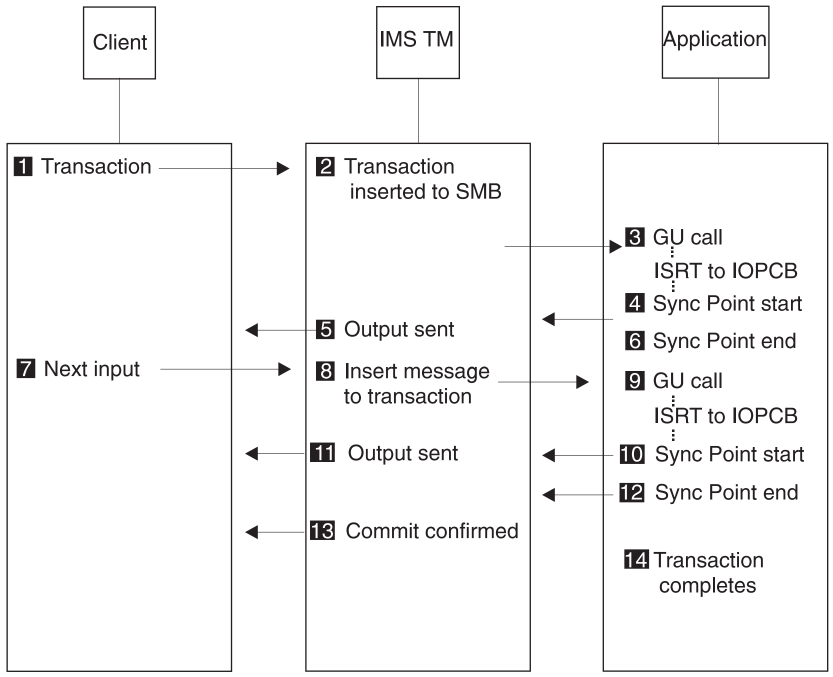


*Figure 9. Send-Then-Commit, Sync Level=None Flow for OTMA Conversational Protocols*

The sample flow shown in Figure 9 assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as NONE in the state-data section of the message prefix. Therefore, IMS does not request a response (an ACK) when sending output.

## Send-then-commit, sync level=confirm

The send-then-commit flow (see Figure 10) assumed no synchronization for the transactions as they are processed by IMS. This section shows a flow in which all transactions are confirmed as they are received (each message requests a response).

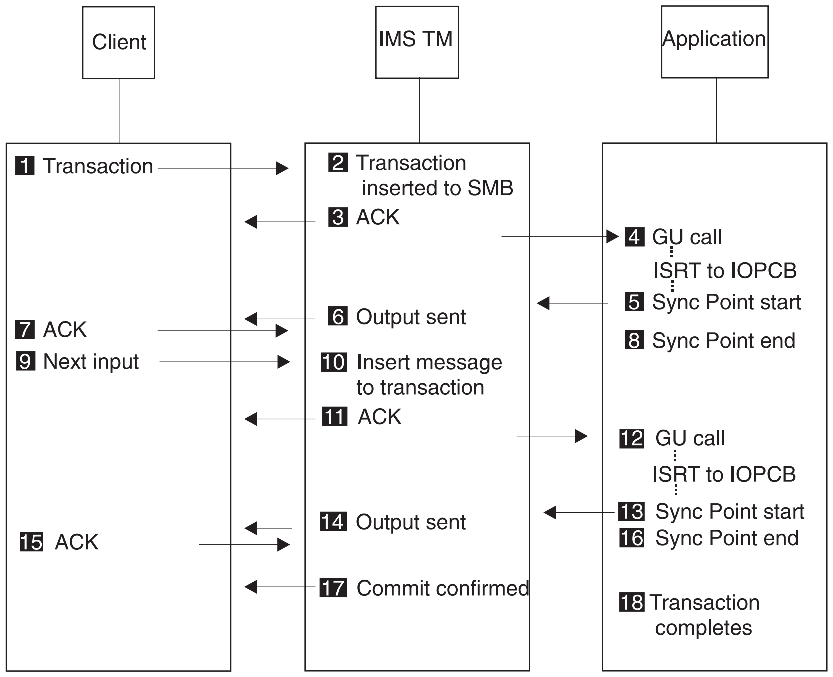


*Figure 10. Send-Then-Commit, Sync Level=Confirm Flow for OTMA Conversational Protocols*

The sample flow shown in Figure 10 assumes the following:

- Commit mode 1 is specified in the state-data section of the message prefix.
- The transaction pipe is not synchronized.
- The synchronization level is specified as Confirm in the state-data section.

## IMS Connect Conversational Protocols

### Send-then-commit, sync level=none, transaction terminated from the program

The send-then-commit flow (see Figure 11) sends IMS output before IMS completes sync-point processing. To use the send-then-commit flow, specify commit mode 1 in the state-data section of the message prefix.

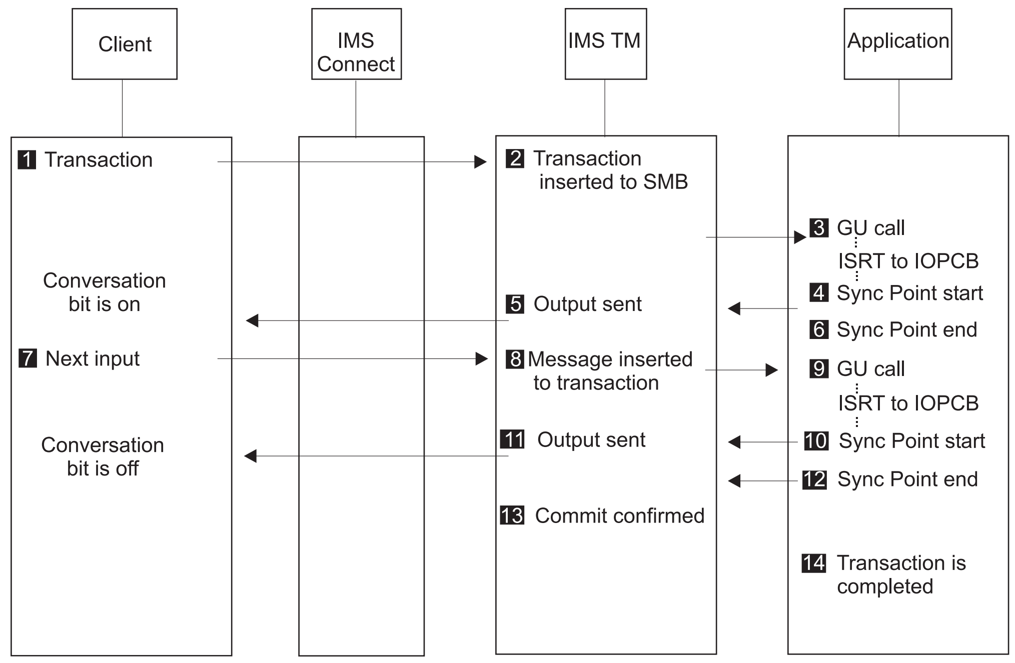


*Figure 11. Send-Then-Commit, Sync Level=None (Transaction Terminated from Program) Flow*

This sample flow shown in Figure 11 assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as NONE in the state-data section of the message prefix. Therefore, IMS does not request a response (an ACK) when sending output.
- The transaction is terminated from the program.
- IMS Connect will close the socket as soon as Commit confirmed has been sent by IMS.

## Send-then-commit, sync level=none, transaction terminated from the client

The send-then-commit flow (see Figure 12) sends IMS output before IMS completes sync-point processing. To use the send-then-commit flow, specify commit mode 1 in the state-data section of the message prefix.

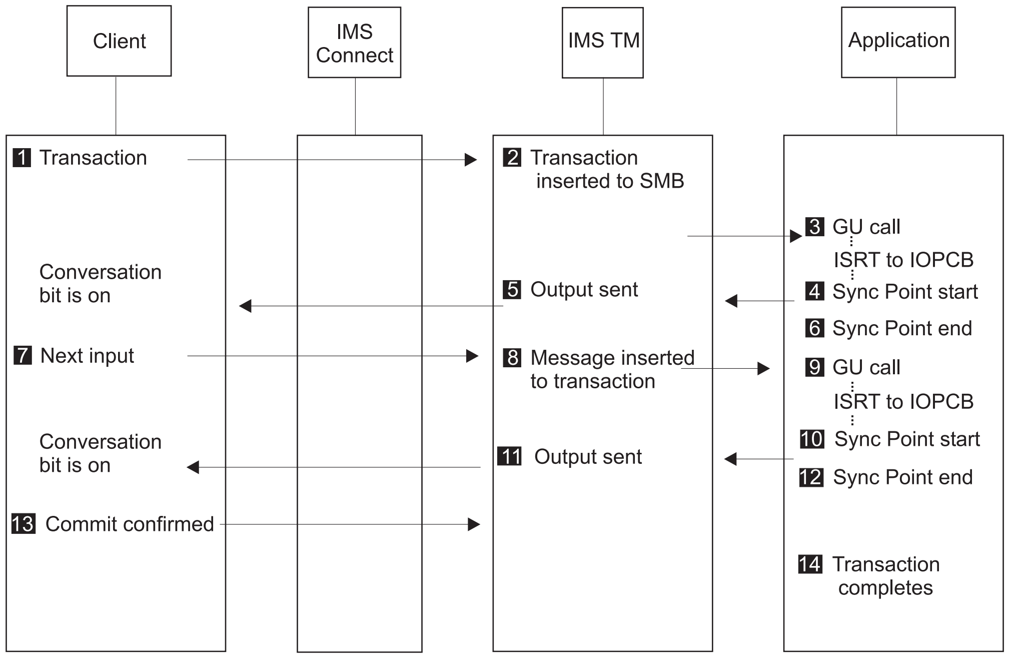


*Figure 12. Send-Then-Commit, Sync Level=None (Transaction Terminated from Client) Flow*

This sample flow shown Figure 12 assumes the following:

- The transaction pipe is not synchronized.
- The synchronization level is specified as NONE in the state-data section of the message prefix. Therefore, IMS does not request a response (an ACK) when sending output.
- The transaction is terminated from client.

## Send-then-commit, sync level=confirm, ACK response

The send-then-commit flow (see Figure 13) assumed no synchronization for the transactions as they are processed by IMS. This section shows a flow in which all transactions are confirmed as they are received (each message requests a response).

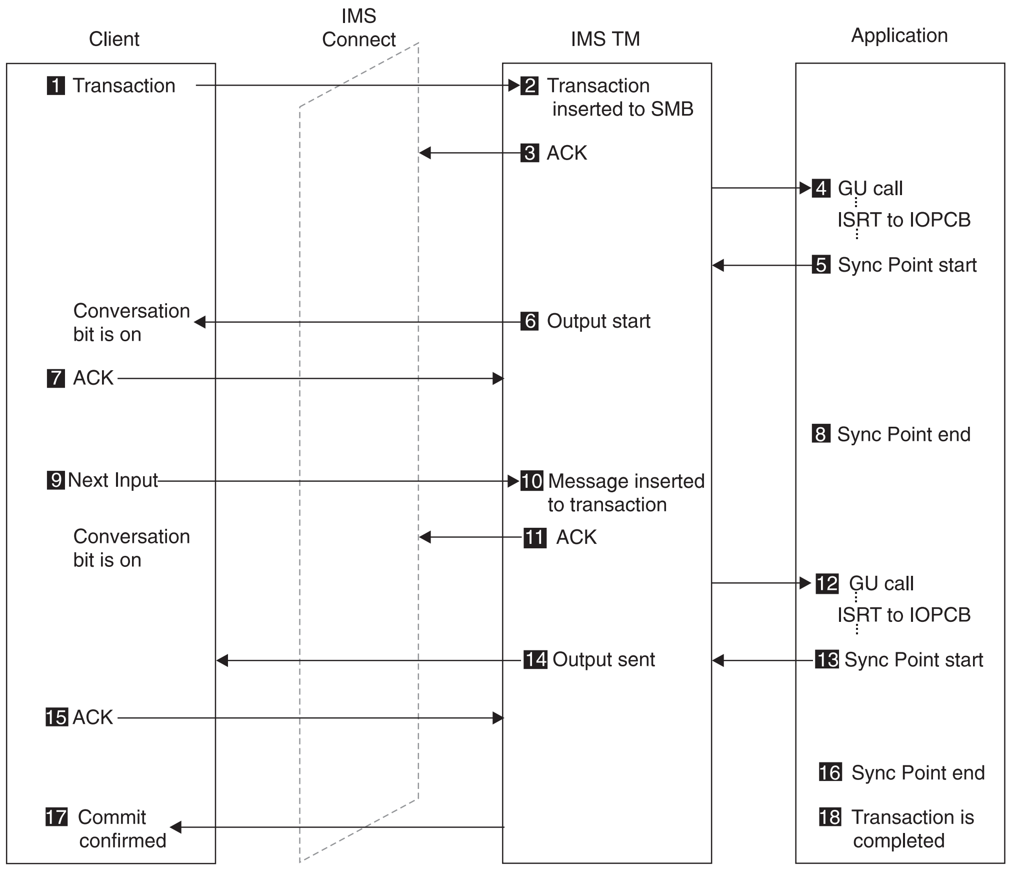


*Figure 13. Send-Then-Commit, Sync Level=Confirm (ACK Response) Flow*

The sample flow shown in Figure 13 assumes the following:

- Commit mode 1 is specified in the state-data section of the message prefix.
- The transaction pipe is not synchronized.
- The synchronization level is specified as Confirm in the state-data section.
- ACK can be replied to by a remote workstation before the check response requested bit.

### Send-then-commit, sync level=confirm, NAK response

The send-then-commit flow (see Figure 14) assumed no synchronization for the transactions as they are processed by IMS. This section shows a flow in which all transactions are confirmed as they are received (each message requests a response).

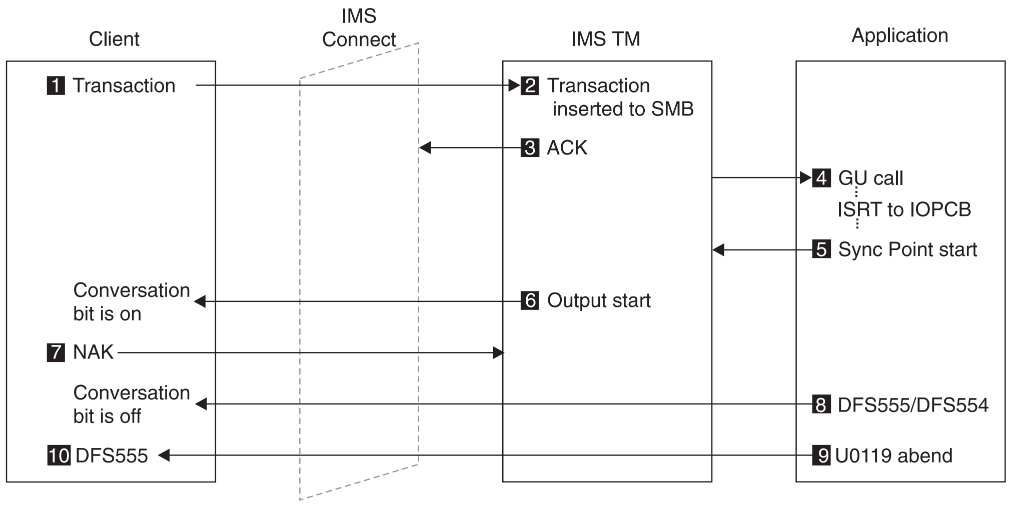


*Figure 14. Send-Then-Commit, Sync Level=Confirm (NAK Response) Flow*

The sample flow shown in Figure 14 assumes the following:

- Commit mode 1 is specified in the state-data section of the message prefix.
- The transaction pipe is not synchronized.
- The synchronization level is specified as Confirm in the state-data section.
- NAK can be replied to by either IMS Connect or a remote workstation before the check requested bit.
- If the client forgets to send the NAK/ACK before it closes the socket, IMS Connect will send the NAK to IMS and it will cause a U0119 abend.

## Commit Mode and Synch Level Definitions

This section defines the different types of commit modes and synch levels.

**Commit mode 0**
: Commit mode 0 is also called Commit-Then-Send. Commit mode 0 is supported on both persistent and transaction sockets (see “Socket Connections” on page 110) and supports only synch level CONFIRM.

**Commit mode 1**
: Commit mode 1 is also called Send-Then-Commit. Commit mode 1 is supported on both persistent and transaction sockets (see “Socket Connections” on page 110) and supports synch levels NONE, CONFIRM, and SYNCH.

**Synch Level=NONE**
: The synchronization level specifies the level of acknowledgement for each transaction. If a transaction is specified with Synch Level=NONE, no acknowledgement is required from the client. The database changes are still committed if the output message is sent to IMS Connect, but not to the client. However, if OTMA is unable to deliver the output message to IMS

Connect, the input and output message are discarded, the database changes are backed out, and the IMS application terminates and returns with a 119 ABEND.

**Synch Level=CONFIRM**
If a transaction is specified with Synch Level=CONFIRM, the client is required to send an acknowledgement to signal to IMS Connect whether or not the output message was successfully (ACK) or unsuccessfully (NAK) processed by the client.

The processing of CONFIRM is dependent on the type of commit mode that you specify:

- If Synch Level=CONFIRM is requested with commit mode 0, and the client responds with ACK, the transaction processing is completed. If the client responds with NAK, the output message will be requeued in IMS for later delivery.
- If Synch Level=CONFIRM is requested with commit mode 1, and the client responds with ACK, the database changes are committed. If the client responds with NAK, the database changes are backed out and the output message is discarded by IMS.

**Synch Level=SYNCH**
If a transaction is specified with Synch Level=SYNCH, two-phase commit processing is required. Use Synch Level=SYNCH when multiple participants are involved in sync point processing. Synch Level=SYNCH is managed through RRS.

## Purge Not Deliverable

If commit mode 0 with purge not deliverable option is selected, an IMS application output message that is not delivered to a client application is removed from the IMS queue. IMS Connect responds to OTMA and sends an ACK, rather than a NAK. The purge not deliverable function is specified on the IRM_F3 flag in the fixed IRM format.

The purge not deliverable function is only applicable to commit mode 0 transactions on persistent and transaction sockets for both RYO (Roll Your Own) and IMS Connector for Java applications. Purge not deliverable is not supported for commit mode 1, nor is it supported for RESUMETPIPE or SENDONLY.

If the purge not deliverable option is not selected with commit mode 0 and the output message cannot be delivered to the client, IMS Connect sends a NAK notification to OTMA and requests that the output message remain on the IMS queue for later retrieval. The output message can be retrieved later by issuing a RESUME TPIPE request.

If commit mode 0, which requires synch level CONFIRM, is selected with the purge not deliverable option, the output message will be discarded if the message is not delivered to the client. If an IMS Connect `STOPCLNT` command is issued for a CLIENTID that has requested purge not deliverable, prior to the IMS application issuing an ISRT to the IO PCB, the message will not be dequeued from IMS. The purge not deliverable function is not supported for IMS application output to ALTPCBs. If the IMS application is commit mode 0 and the purge not deliverable option has been specified, the purge not deliverable function does not apply to ISRTs to ALTPCBs.

## Recoverable IMS Transactions

This section contains some scenarios when running recoverable transactions in the IMS Connect environment. For each of the following scenarios:

- OTMA will have deleted the input message.
- Requeuing of the input message will not occur.
- For commit mode 1 (send-then-commit), none of the output is placed (ENQUEUED) in the IMS queue.

Only commit mode 0 (commit-then-send) is treated as recoverable; Commit mode 1 is not recoverable. With the use of commit mode 0, IMS Connect creates a separate TPIPE for each client that uses commit mode 0. This TPIPE remains in IMS, so a fixed client name is highly recommended for each client that intends to use commit mode 0.

The combination of commit mode and Sync level is critical. The following scenarios describe the different uses and the results.

- With commit mode 1 and SYNC LEVEL = NONE:

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends the message to the client, and any ACK/NAK from the client in response to the output message would become an error because the ACK/NAK are not expected and IMS Connect would have received a message from the client with no application data.
- With commit mode 1 and SYNC LEVEL = CONFIRM:

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it to the client, and an ACK from the client will result in the successful completion of the application. This scenario works as expected.

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it back to the client, and a NAK from the client will result in an IMS MPP 119 abend and an IMS message, DFS555. The 119 abend will back out the database changes, and both the input and output messages are discarded. The result would be as if the system had never seen the transaction, and a reentry of the transaction would be necessary.
- With commit mode 0 and SYNC LEVEL = CONFIRM:

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it to the client, and an ACK from the client will result in the successful completion of the application. Commit mode 0 forces the Synch level to Confirm. This scenario works as expected.

  The input message is processed by IMS and an output message is sent back to IMS Connect, IMS Connect sends it back to the client, and a NAK from the client will result in the database changes not being backed out. The input message is discarded and the output message is requeued to the IMS queue for representation. These output messages will be moved to the hold asynchronous queue by OTMA, and will be retrievable only with the RESUME TPIPE, RECEIVE and ACK process.

**Recommendation:** To run recoverable transactions in the IMS Connect environment, use commit mode 0 and SYNC LEVEL = CONFIRM, and use a single unique CLIENT_ID for each client that uses commit mode 0 and SYNC LEVEL = CONFIRM

## Send Only Protocol

- **Commit-then-send** with commit confirmed flag on

  The commit-then-send flow, also known as the IMS standard flow, enqueues IMS output before sending it to the client. However, in this case for non-response transactions, the client does not expect any output from IMS (see Figure 15).

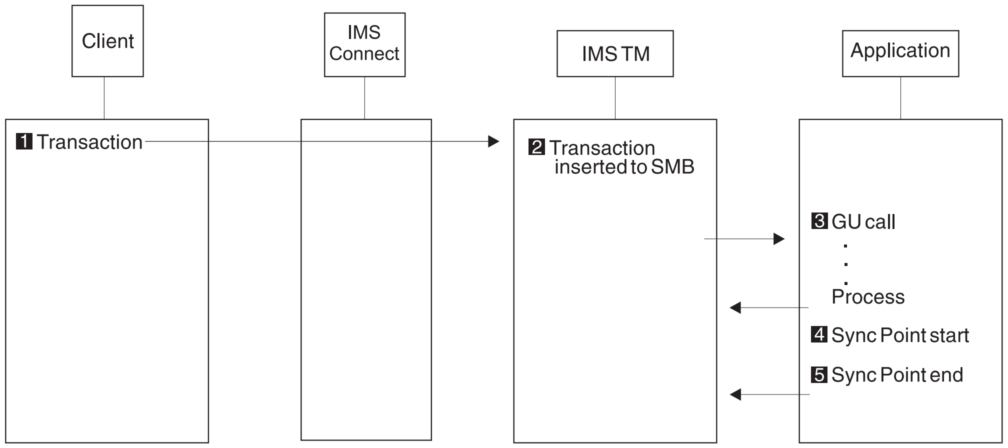


*Figure 15. Send Only Protocol Flow*

The sample flow shown assumes the following:

- Commit mode 0 is specified in the state-data section of the message prefix.
- The transaction bit and the commit confirmed bit is specified in the control-data section of the message prefix.

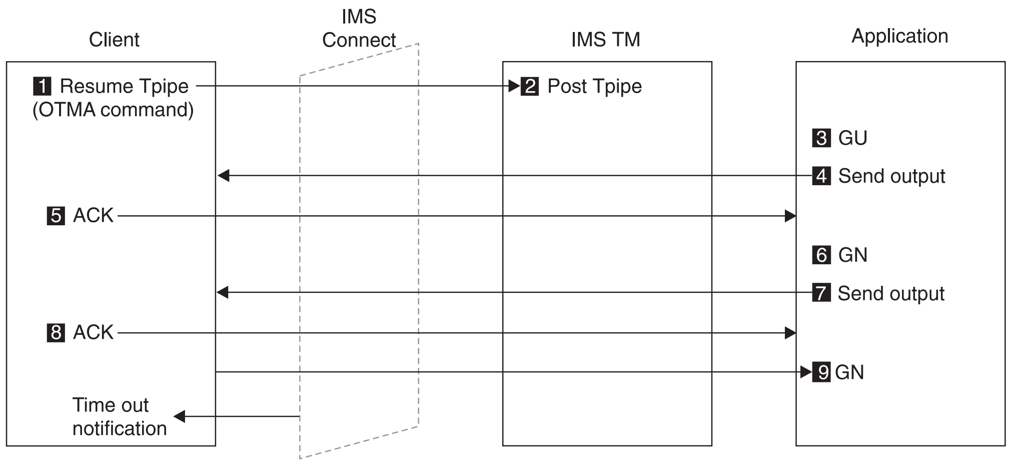
## Resume Tpipe/Receive Protocol for Asynchronous Output

- **Commit-then-send** (receive asynchronous output)

  The commit-then-send flow, also known as the IMS standard flow, enqueues IMS output before sending it to the client (see Figure 16 on page 109) with the client application sending a positive acknowledgement (ACK) for both outputs. This removes the output from the IMS queue.

  **Requirement:** Use this protocol to retrieve asynchronous output from IMS. The client signals how long to wait for output from IMS by specifying an IRM timeout value with the IRM_TIMER field; the IRM timeout value affects the RESUME TPIPE command sent to IMS Connect and the ACK/NAK sent to IMS Connect.

|



| *Figure 16. Commit-Then-Send, Receive Asynchronous Output (Client Waits for Output) Flow*

The sample flow shown assumes the following:

– The client sends the OTMA command `RESUME TPIPE` to ask IMS OTMA to post the named Tpipe (the client name).
– The client issues a RECEIVE request to receive the output from IMS.
– The client sends ACK to IMS (required for commit-then-send).
– The client receives the next output from IMS.
– The client sends ACK to IMS.
– The client waits for the next output from IMS, or for Time out notification.

- **Commit-then-send** (receive asynchronous output)

  The commit-then-send flow, also known as the IMS standard flow, enqueues IMS output before sending it to the client (see Figure 17 on page 110) with the client application sending a positive acknowledgement (ACK) for the first output (removing the output from the IMS queue) and a NAK to the second output (which results in the output remaining in the queue).

  **Requirement:** Use this protocol with the timeout function. Otherwise, the client will hang if there are no more messages to send.

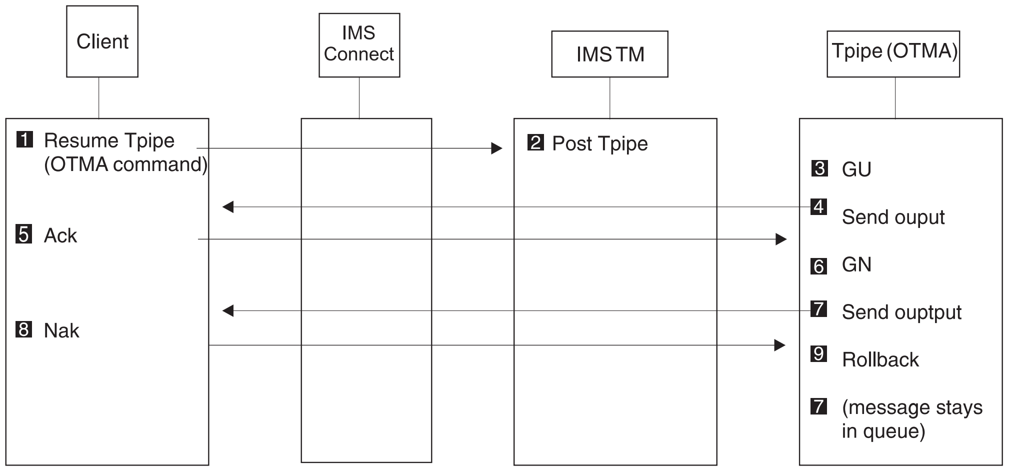


*Figure 17. Commit-Then-Send, Receive Asynchronous Output (Output Remains in Queue) Flow*

The sample flow shown assumes the following:

– The client sends the OTMA command `RESUME TPIPE` to ask IMS OTMA to post the named Tpipe (the client name).
– The client receives the output from IMS.
– The client sends ACK to IMS (required for commit-then-send).
– The client receives the next output from IMS.
– The client sends NAK to IMS.
– The message stays in the queue.

## Socket Connections

IMS Connect provides three kinds of client TCP/IP connection protocols, which are called *sockets*. The TCP/IP sockets define how IMS Connect manages client TCP/IP connections when IMS Connect sends a disconnect message. The three socket types provided by IMS Connect are:

- Persistent
- Transaction
- Non-persistent

**Important:** IMS Connect supports the three socket types when used to communicate with IMS Version 7 and later releases. The three socket types are also operational when IMS Connect communicates with IMS Version 5 and IMS Version 6.

## Persistent Sockets

A *persistent* socket is a connection between the client and IMS Connect that remains connected until either the client or IMS Connect specifically make a disconnect request. A persistent socket can exist across multiple transactions.

There are two ways that the client can force a termination:

- By sending IMS Connect a disconnect request.
- By changing the socket type to ″transaction″ for the last transaction entered, such as a logoff transaction.

IMS Connect can also terminate the connection when an error occurs.

The IMS Connect user message exits HWSIMSO0, HWSIMSO1, HWSSMPL0, HWSSMPL1, and HWSJAVA0 support the use of persistent sockets. IMS Connector for Java also supports the use of persistent sockets.

A persistent socket supports both commit mode 1 and commit mode 0 processing.

## Transaction Sockets

A *transaction* socket is a connection between the client and IMS Connect that remains connected for a single transaction or IMS conversation. The connection can be terminated only by IMS Connect, either when IMS itself terminates, or when an error occurs.

A transaction socket supports both commit mode 1 and commit mode 0 processing.

## Non-Persistent Sockets

A *non-persistent* socket maintains a connection for a single input-and-output pair to IMS Connect. IMS Connect terminates the connection after sending the output to the client for non-conversational and conversational transactions. If three exchanges of input and output occur, the disconnect is issued three times, one for each output from IMS Connect.

**Restrictions:** The HWSIMSO0, HWSIMSO1, HWSSMPL0, HWSSMPL1, and HWSJAVA0 user message exits do not support non-persistent sockets, nor does IMS Connector for Java.

## Setting Socket Types

Client code controls the socket settings, and the IMS Connect user message exits and the user initialization exit enforce the socket settings.

The client selects the socket connection type by setting a flag in IRM, in the field IRM_SOCT. The IRM_SOCT flag values are seen in Table 44.

*Table 44. IRM_SOCT Flags*

| Flag | Definition | Socket Type |
|---|---|---|
| IRM_SOCT_PER | X’10’ | Persistent |
| IRM_TRAN | X’00’ | Transaction |
| IRM_SOCT_NONPER | X’40’ | Non-persistent |

The IRM_SOCT flag must be set for each message that is sent to IMS Connect.

**Recommendation:** Set all messages that are associated with a single transaction to the same socket type. If you do not, unexpected results can occur, as described in the following examples:

- If the first message of a conversational transaction is set to persistent, and the last message is set to transaction, then the socket connection will be terminated following the last message.
- If one of the messages in the middle of the conversational transaction set the socket type to transaction, and the IMS transaction terminates for some reason, then IMS Connect will disconnect the socket. This is because ″transaction″ was the last known socket type.

The user message exits will determine the socket type, then move the socket type information to the OTMA User Header. To transfer the socket type information to the OTMA User Header, the user exits set the OMUSR_FLAG1 field with one of the following flags as seen in Table 45:

*Table 45. OMUSR_FLAG1 Flags*

| Flag | Definition | Socket Type |
|---|---|---|
| OMUSR_PSOCKET | X’10’ | Persistent |
| OMUSR_TRAN | X’00’ | Transaction |
| OMUSR_NPSOCKET | X’40’ | Non-persistent |

**Related Reading:**

- For information about the OTMA User Header layout, see ″HWSOMPFX″ in Appendix B, “OTMA Headers,” on page 235.
- For information about the IRM layout, see “How IMS Connect Communicates with a TCP/IP Client” on page 39.

## Socket Processing for Transactions

For a transaction on either a transaction socket or persistent socket, the client application must always issue a TCP/IP READ following all TCP/IP SENDs. The exceptions are for a TCP/IP SEND of SENDONLY or a TCP/IP SEND of an ACK with IRM_TIMER set to NO_WAIT (X'E9' char Z), which is issued in response to a READ of a RESUME_TPIPE single request.

The following scenarios describe transactions on a transaction socket. For transactions on a persistent socket, the process is the same as transactions on a transaction socket. However, the client application and IMS Connect do not disconnect. Also, the client application will receive a return code of X'28' if there is a timeout. The return code states a disconnect is not required.

For a Commit mode 0, Synch Level Confirm, non-conversational transaction on a transaction socket, the following scenario occurs:

1. The client application issues a SEND to send the transaction data to IMS Connect.
2. IMS Connect returns the output to the client application.
3. The client application receives the output, sends an ACK, and must issue a READ to receive the next output or the timeout notification.
4. IMS Connect issues a timeout notification with the return code of either X'20' or X'24' for a transaction socket, or an X'28' for a persistent socket. IMS Connect will disconnect the socket for the X'20' and X'24' return codes, and will keep the connection for the X'28' return code.

5. The client application issues a disconnect for return codes X'20' and X'24'. The client can issue a disconnect for return code X'28' or send in the next input.

For a Commit mode 1, Synch Level Confirm, non-conversational transaction on a transaction socket, the following scenario occurs:

1. The client application issues a SEND of the transaction data to IMS Connect.
2. IMS Connect returns the output to the client application.
3. The client application receives the output, sends an ACK or NAK, and issues a READ.
4. After an ACK is sent, the client receives one of the following responses:
   - Deallocate commit if the IMS transaction completes successfully.
   - A DFS message if the IMS transaction failed.
   - A timeout notification with a return code of X'20' or X'24' for a transaction socket or a return code of X'28' for a persistent socket. The client application is required to issue a disconnect for return codes X'20' and X'24'.
5. The client application issues a disconnect.

For a Commit mode 1, Synch Level Confirm, conversational transaction on a transaction socket, the following scenario occurs:

1. The client application issues a SEND to send the transaction data to IMS Connect.
2. IMS Connect returns the output to the client application.
3. The client application receives the output, sends an ACK, and issues the next input. The client continues SEND, READ, ACK until the transaction is complete.
4. IMS Connect issues an RSM deallocate commit, deallocate abort, or a timeout notification. The timeout notification returns either X'20' or X'24', which indicates that IMS Connect will disconnect.
5. The client application issues a disconnect.

## Asynchronous Output Support

This addresses asynchronous (unsolicited) output processing from a user-written client application. For information on how to process asynchronous output messages, see the *IMS Connector for Java* online help in WebSphere Studio Application Developer Integration Edition Version 5.0.1.

IMS Connect can manage asynchronous output by not allowing it to flow while a transaction is being processed. There are two types of asynchronous output:

- Output that is sent to a client from an IMS application using the ALTPCB.
- Any commit-then-send (commit mode 0) output that is being sent to the client for which the client or IMS Connect sends a NAK in response to the output message.

IMS Connect communicates the presence of asynchronous output to the client from a commit mode 0 (commit-then-send) output response message in one of the following ways:

- By returning the flag `CSM_AMSG` in the CSM_FLG1 field in the CSM (complete status message)
- By returning the flag `RSM_AMSG` in the RSM_FLG1 field in the RSM (request status message)

If you do not want to implement IMS Connect asynchronous output support, your client application does not need to analyze the CSM or the RSM. IMS Connect communicates the presence of asynchronous output regardless of whether a client application requests the asynchronous output.

Use the RESUME TPIPE function to retrieve the asynchronous output from the client. You can retrieve asynchronous output on both persistent and transaction sockets.

**Restrictions:**

- Asynchronous output is supported only in IMS Version 7 and later releases. Asynchronous output support is not operational when IMS Connect is communicating with earlier IMS versions.
- IMS Connector for Java supports only the asynchronous option, SINGLE.

## Implementing Asynchronous Output Support

You implement asynchronous output support by enabling the receipt of the asynchronous output. The end user of the client application can decide when to request the asynchronous output, or the client application itself can decide when to request the asynchronous output.

**Recommendation:** Implement asynchronous output support so that the *end user*, not the client application, decides when to request the asynchronous output. Such an implementation provides these benefits:

- Ensures that the transaction input and output is separated from the asynchronous output.
- Enables the end user to select, at a time interval of their choice, when to retrieve the asynchronous output.

Regardless of whether or not the end user or the client application requests the asynchronous output, the following actions must occur, in this order:

1. Issue a CONNECT command.
2. A TCP/IP SEND of an OTMA RESUME TPIPE command, immediately followed by a TCP/IP READ function from the primary client application.
3. A TCP/IP SEND of an ACK or NAK response on the receipt of the output message. If the ACK was sent with a timer value of NOWAIT (NOWAIT is only valid for RESUME TPIPE with SINGLE or SINGLE with WAIT option), go to step 5. If NAK was sent, go to step 5.
4. A TCP/IP READ function from the primary client application. Repeat steps 2 and 3 until either all messages have been received, until the end user has received all of the messages that they want, until an error occurs, or until time out notification occurs.
5. Issue a DISCONNECT command, if you are using transaction sockets. If you are using persistent sockets, the connection is still connected.

### Enabling End User Asynchronous Output Requests

You can easily implement the CONNECT, RESUME TPIPE, READ, ACK/NAK, and DISCONNECT functions on the client application’s screen with the buttons on the graphical user interface.

- Create a CONNECT button.
- Create a RESUME TPIPE button to send a RESUME TPIPE command request to IMS Connect. IMS Connect will then send a RESUME TPIPE request to OTMA.

- Create a READ button to issue a TCP/IP READ request. OTMA will send a message to IMS Connect following the RESUME TPIPE or ACK response.
- Create an ACK/NAK button.
- You can also combine the READ and ACK requests into a single button that issues the READ request, then sends an ACK on receiving the message.
- Create a DISCONNECT button.

## Managing and Controlling Asynchronous Output Messages

Asynchronous output message functions are controlled by information that is passed in the IRM and then set in the OTMA header by the message exit. There are five types of asynchronous output message control: single, single with wait, noauto, nooption, and auto. The IMS Connect user message exits, HWSSMPL1, HWSSMPL0, HWSIMSO1, and HWSIMSO0 support all these options. To choose a type of message control, the client code sets the IRM field IRM_FLG5 to be one of the following values:

**IRM_F5_ONE**
Retrieves a single message (single).

**IRM_F5_SWAIT**
Waits for a single message if none are currently present in the IMS message queue (single with wait).

**IRM_F5_NOAUTO**
Retrieves all messages that have been queued (noauto).

**IRM_F5_AUTO**
Retrieves all messages that have been queued, then retrieves any additional messages that are queued later (auto).

**IRM_F5**
Makes RESUME TPIPE function like NOAUTO (nooption) when set to X'00'.

The HWSSMPL0, HWSIMSO0, HWSIMSO1, and HWSSMPL1 user message exits default to the noauto type of asynchronous output message management.

The rest of this section describes the asynchronous output message control options in detail.

### Single Message Control

When using the single message control option (by setting field IRM_F5 to IRM_F5_ONE), the client can receive only a **single message**. If there are no messages in the IMS OTMA Asynchronous Queue for the client ID when the request is made, no message will be returned and a time out will occur. Using the single message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
   a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE function processing.
   b. If the socket type is a transaction socket, the RESUME TPIPE function must be issued after the CONNECT function.
2. Client issues RESUME TPIPE function with the correct IRM settings.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK or NAK to IMS Connect.
   a. The ACK or NAK can be sent with a user-specified numeric timeout value.

   Or

   b. Specify NOWAIT for the timeout value.

5. If a numeric timeout value is specified, the client must issue a RECEIVE function to receive the timeout notification. If the NOWAIT option is specified, no timeout notification is sent. Therefore, the client must not issue a RECEIVE function if NOWAIT is specified.
6. IMS Connect disconnects the Socket from the Host end if the socket connection is a transaction socket. If the socket connection is a persistent socket, IMS Connect does not disconnect the socket.
7. Client must issue a DISCONNECT function if the socket connection is a transaction socket. If the socket is a persistent socket, the client can either DISCONNECT the socket or choose to send in a new request such as SENDONLY, SEND of Tran code and Data, or issue another RESUME TPIPE request.

If the client responds with a NAK rather than an ACK, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, and can be re-retrieved later. IMS Connect will continue to process as described in events five through seven when a NAK is sent to IMS Connect by the Client.

## Single with Wait Message Control

When using the single with wait message control option (by setting IRM_F5 to IRM_F5_SWAIT), the client can receive only one single message; however, unlike single message control, the single with wait message control can receive a message that is placed in the IMS OTMA Asynchronous Queue for the client ID. Using the single with wait message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
   a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE function processing.
   b. If the socket type is transaction socket, then the RESUME TPIPE function processing must be issued after the CONNECT function.
2. Client issues RESUME TPIPE function, with the correct IRM settings.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK or NAK to IMS Connect.
   a. The ACK or NAK can be sent with a user-specified timeout notification.

      Or

   b. Specify NOWAIT for the timeout value.
5. If a numeric timeout value is specified, the client must issue a RECEIVE function to receive the timeout notification. If the NOWAIT option is specified, no timeout notification is sent. Therefore, the client must not issue a RECEIVE function if NOWAIT is specified.
6. IMS Connect disconnects the Socket from the Host end if the socket connection is a transaction socket. If the socket connection is a persistent socket, IMS Connect does not disconnect the socket.
7. Client must issue a DISCONNECT function if the socket connection is a transaction socket. If the socket is a persistent socket, the client can either DISCONNECT the socket or choose to send in a new request such as SENDONLY, SEND of Tran code and Data, or issue another RESUME TPIPE request.

If the client responds with a NAK rather than an ACK, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, and can be re-retrieved later. IMS Connect will continue to process as described in events five through seven when a NAK is sent to IMS Connect by the Client.

## Noauto Message Control

When using the noauto message control option (by setting field IRM_F5 to IRM_F5_NOAUTO), the client can receive **all** of the messages on the OTMA Asynchronous Queue. Using the noauto message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
   a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE function processing.
   b. If the socket type is transaction socket, then the RESUME TPIPE function processing must be issued after the CONNECT function.
2. Client issues RESUME TPIPE function.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK to IMS Connect.
5. Client repeats events three and four until event six occurs.
6. IMS Connect disconnects the Socket from the Host end.
7. Client issues DISCONNECT function.

Using the noauto message control option, the client can always terminate by issuing a DISCONNECT function after sending an ACK to IMS Connect.

If the client responds with a NAK rather than an ACK, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, and can be re-retrieved later. IMS Connect will terminate the socket as described in event six.

## Nooption Message Control

When using the nooption message control option (by setting field IRM_F5 to X'00'), the client can receive **all** of the messages on the OTMA Asynchronous Queue. Using the nooption message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
   a. Following the CONNECT function, if the socket type is persistent socket, one or more transactions can be sent and the responses received before RESUME TPIPE function processing.
   b. If the socket type is transaction socket, then the RESUME TPIPE function processing must be issued after the CONNECT function.
2. Client issues RESUME TPIPE function.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK to IMS Connect.
5. Client repeats events three and four until event six occurs.
6. IMS Connect disconnects the Socket from the Host end.
7. Client issues DISCONNECT function.

Using the nooption message control option, the client can always terminate by issuing a DISCONNECT function after sending an ACK to IMS Connect.

If the client responds with a NAK rather than an ACK, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, and can be re-retrieved later. IMS Connect will terminate the socket as described in event six.

## Auto Message Control

When using the auto message control option (by setting field IRM_F5 to IRM_F5_AUTO), the client can receive all of the messages on the OTMA Asynchronous Queue, ***and*** any messages that are placed on the OTMA Asynchronous Queue after the current messages are all removed. Using the auto message control option will force the following sequence of events to occur:

1. Client issues CONNECT function.
2. Client issues RESUME TPIPE function.
3. Client issues RECEIVE function to receive the Asynchronous output.
4. Client sends ACK to IMS Connect.
5. Client repeats events three and four.

If all messages have been removed from the queue, event three will remain active (that is, in receive state) until the user-specified timer supplied in the IRM has expired. IMS Connect will then terminate the socket. See “Values for Asynchronous Output Processing” on page 120 for information on the timer value.

**Recommendation:** If event three or event five receives a disconnect of the socket, the client should disconnect and then wait for a time interval before repeating events one through five.

Using the auto message control option, the client can always terminate the connection by either

- responding to the output message with a NAK response, or
- sending a DEALLOCATE request rather than an ACK.

The message being processed is put back on the IMS output queue, and IMS Connect terminates the socket.

If the client responds with an ACK, then issues a DISCONNECT, the connection is only terminated between the client and TCP/IP; the client remains in a CONN state with IMS Connect. When IMS Connect attempts to send the next asynchronous output message, IMS Connect is notified that the connection has been lost. IMS Connect does not acknowledge (NAK) OTMA, and the message is put back on the IMS output queue. IMS Connect then terminates the socket. If the client issues an ACK and then issues a disconnect, followed by a connect and transmittal of data, IMS Connect responds with duplicate client ID and disconnects the socket connection.

If the client responds with a NAK rather than an ACK in events three or five, the message that has been NAKed will be put back on the OTMA Asynchronous Hold Queue, IMS Connect will terminate the socket, and then those messages can be re-retrieved later.

**Note:** The IMS Connect AUTO support is based on the premise that the socket connection is dedicated as an output-only device. Combining RESUME TPIPE (with auto asynch option specified) with transactions on the same socket connection or SENDONLY on a persistent socket, can yield unpredictable results. If you wish to change from RESUME TPIPE auto option mode to a mode that will allow for transaction processing, you must change the auto asynch option by performing one of the following options:

1. NAK one of the RESUME TPIPE outputs. This will change the asynch mode from auto to noauto. To return to auto mode, a RESUME TPIPE with auto must be specified.
2. On a timeout notification associated with the RESUME TPIPE AUTO, the client application can disconnect, reconnect, and issue a RESUME TPIPE with single, single with wait, noauto, or nooption with a very short IRM_TIMER value. The IRM_TIMER value should be small, so that a timeout notification can be returned immediately. Issuing a RESUME TPIPE with one of the four asynch mode options, changes the mode from auto to one of the specified options. After the RESUME TPIPE is issued and a timeout notification is returned, the client application can send in a transaction.
3. On a timeout notification associated with the RESUME TPIPE AUTO, the client application can disconnect, reconnect, and issue a RESUME TPIPE with single, single with wait, noauto, or nooption with any valid IRM_TIMER value. Upon receiving an output message, send an ACK with IRM_TIMER set to nowait or a valid value. If the IRM_TIMER value is set to nowait, the client can then send in a transaction. If the IRM_TIMER is set to a valid value, after receiving the timeout notification, the client application can then send in a transaction.

## Execution Time Out During RESUME TPIPE with Auto Message Control Option

If you are using RESUME TPIPE with the auto message control option and the IRM_TIMER value times out, you may experience some unpredictable results. If the auto option is selected on the RESUME TPIPE and a timeout occurs, to get the timeout notification and send transactions again, you must change the auto option processing mode to noauto. To get out of the auto option processing mode, you can choose one of the following options:

- Issue RESUME TPIPE with the auto option and set a large IRM_TIMER value to ensure that the client application will NAK the output. When the output is NAK, OTMA will change the asynchronous mode from auto to noauto to stop the sending of asynchronous output. The client application then issues READ to retrieve the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue RESUME TPIPE with the noauto option and set any value in the IRM_TIMER field. After receiving ACK output, repeat READ of asynch output and SEND of ACK until a timeout notification is received. (Issuing RESUME TPIPE with noauto changed the processing mode from auto to noauto. This also reset the asynchronous mode in OTMA to noauto where OTMA no longer supports the automatic sending of asynch output when the IMS Message Queue is empty.) The client application then issues READ to retrieve the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue RESUME TPIPE with noauto option and set any value in the IRM_TIMER field. If you receive NAK output, the processing mode and OTMA Asynch mode is reset to noauto. Resetting the OTMA Asynch mode to noauto stops the sending of asynch output and the NAK output terminates the process. The client application then issues READ to retrieve the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.
- Issue RESUME TPIPE with single option and set any value in the IRM_TIMER field. The OTMA Asynch mode is reset from auto to single and no more asynchronous messages are sent. After you receive ACK or NAK output with an IRM_TIMER setting that is anything other than NO_WAIT, the single option has

been completed and the client application can issue a READ to get the timeout notification. Upon receiving the timeout notification, the client can begin sending transactions to IMS Connect.

- Issue RESUME TPIPE with single option and set any value in the IRM_TIMER field. The OTMA Asynch mode is reset from auto to single and no more asynchronous messages are sent. After you receive ACK or NAK output with an IRM_TIMER setting of NO_WAIT, the single option has been completed and the client application does not have to issue a READ to get timeout notification. The client application can start sending transactions to IMS Connect.

## Values for Asynchronous Output Processing

This section provides values for asynchronous output processing for socket type, commit mode, sync level, timer setting, and resume TPIPE options.

For a RESUME TPIPE request, set the values as follows:

**Socket Type**
Transaction or Persistent

**Commit Mode**
Zero

**Sync level**
Confirm

**Timer setting**
The timeout range required by your enterprise.

**Resume TPIPE options**
Single, single with wait, auto, noauto, or nooption.

For example, if you want to create a dedicated output client that only receives unsolicited output, start a client application to complete the following sequence:

1. The client application performs a connection sequence.
2. The client application sends a RESUME TPIPE request with the correct settings in the IRM.

   **Recommendation:** Set the IRM_TIMER value to X'FF', which causes IMS Connect to override the TIMEOUT value in the configuration file and wait forever.
3. The client application sends a TCP/IP READ to receive the output message.
4. The client application sends an acknowledgment (ACK[1]) and returns to the TCP/IP READ.

The timer interval that is set in IRM_TIMER is a different timer value from the one that is set in the IMS Connect configuration file (that value is TIMEOUT=).

The IRM_TIMER value is the wait value to wait for a RECEIVE issued from the client following a RESUME TPIPE, or an ACK to the RECEIVEs following the RESUME TPIPE.

See “IRM_TIMER Usage” on page 52 for more information.

1. Set the IRM_TIMER value to the same value you set on the RESUME TPIPE.

## Asynchronous Output Message Flow

Implementing asynchronous output support forces a commit-then-send (commit mode 0) message flow. This flow requires an acknowledgment (ACK/NAK) from the client.

If an IMS transaction running in commit-then-send message flow sends a message to the client, and that message cannot be delivered, OTMA will react as though a NAK had been sent to OTMA from IMS Connect, and the message will be placed on the OTMA Hold Queue. OTMA will behave in this manner for whatever reason that the NAK gets sent (for example, because the XCF connection is not available, because IMS Connect has terminated, or because IMS Connect has lost communications with TCP/IP).

**Related Reading:**

- For more information about the format of the OTMA headers, see ″HWSOMPFX″ in Appendix B, "OTMA Headers," on page 235.
- For detailed information about the IRM layout, see "How IMS Connect Communicates with a TCP/IP Client" on page 39.
- For detailed information about the CSM and RSM layouts, see "Output Message From Message Exit" on page 80.
- For detailed information about the purge not deliverable support, see "Purge Not Deliverable" on page 106.

## IMS Connect Client Call Flows

This section illustrates several sample IMS Connect client flows for conversational and non-conversational transactions. Figure 18 on page 122, Figure 19 on page 122, Figure 20 on page 122, Figure 21 on page 123, Figure 22 on page 123, and Figure 23 on page 123 are all examples of IMS Connect client flows for conversational and non-conversational transactions. All sample flows shown apply to both persistent and transaction TCP/IP sockets, and all flows use this protocol: commit mode 1 (send-then-commit), synch level = confirm, with ACK and NAK. The following sample flows are illustrated:

- Non-conversational, running to successful completion using ACK
- Conversational, running to successful completion using ACKs
- Non-conversational, where client sends NAK in response to message
- Conversational, where client sends NAK in response to one of the messages
- Non-conversational, terminated by Host application before successful completion of transaction
- Conversation terminated by Host application before successful completion of transaction

**Important:** These figures describe and show various protocols as used with IMS Versions 6 and 7. For more information about protocols that are used with IMS Version 5, see Table 46 on page 124.

```
CLIENT                  FLOW               IMS CONNECT
REQUEST                                      REQUEST

  SEND----------------->IRM/TRAN/DATA ------------>RECEIVE
  RECEIVE<-----------------DATA/CSM<---------------SEND
  SEND-------------------->IRM/ACK--------------->RECEIVE

  RECEIVE<--------------------RSM<-----------------SEND DEALLOCATE CONFIRM
       RSM reason code = DEALLOCATE CONFIRM  X'61' (97)
     (97 = IMS Host application has committed the transaction)
```

*Figure 18. Non-conversational, Commit Mode=1, Synch Level=Confirm, and ACK (Transaction Runs to Successful Completion)*

```
CLIENT                  FLOW               IMS CONNECT
REQUEST                                      REQUEST

  SEND----------------->IRM/TRAN/DATA ------------>RECEIVE
  RECEIVE<--------------- DATA/CSM<----------------SEND
  SEND------------------->IRM/ACK---------------->RECEIVE
  SEND------------------->IRM/DATA--------------->RECEIVE
  RECEIVE<-----------------DATA/CSM<---------------SEND
  SEND-------------------->IRM/ACK---------------->RECEIVE
                             .
                             .
                             .
  SEND------------------->IRM/DATA -------------->RECEIVE
  RECEIVE<-----------------DATA/CSM<---------------SEND

  SEND------------------->IRM/ACK---------------->RECEIVE

  RECEIVE<-------------------RSM<------------------SEND DEALLOCATE CONFIRM
         RSM reason code = DEALLOCATE CONFIRM  X'61' (97)
     (97 = IMS Host application has committed the transaction)
```

*Figure 19. Conversational, Commit Mode=1, Synch Level=Confirm, and ACK (Transaction Runs to Successful Completion)*

```
CLIENT                  FLOW               IMS CONNECT
REQUEST                                      REQUEST

  SEND----------------->IRM/TRAN/DATA ------------>RECEIVE
  RECEIVE<-----------------DATA/CSM<---------------SEND
  SEND-------------------->IRM/NAK--------------->RECEIVE

  RECEIVE<------------------------------------ IMS MESSAGE "DFS555.."
```

*Figure 20. Non-conversational, Commit Mode=1, Synch Level=Confirm, and NAK (Transaction Terminates with a NAK from Client Application)*

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST

  SEND----------------->IRM/TRAN/DATA ----------->RECEIVE
  RECEIVE<-------------- DATA/CSM<---------------SEND
  SEND-------------------->IRM/ACK--------------->RECEIVE
  SEND------------------->IRM/DATA-------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND------------------->IRM/ACK--------------->RECEIVE
                              .
                              .
                              .
  SEND------------------->IRM/DATA-------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND------------------->IRM/NAK--------------->RECEIVE

  RECEIVE<------------------------------------ IMS MESSAGE "DFS555.."
```

*Figure 21. Conversational, Commit Mode=1, Synch Level=Confirm, and NAK (Transaction Terminates with a NAK from Client Application)*

```
 CLIENT                  FLOW              IMS CONNECT
 REQUEST                                     REQUEST

  SEND----------------->IRM/TRAN/DATA ----------->RECEIVE
            Host Application abnormally terminates
  RECEIVE<------------------------------------ IMS MESSAGE "DFS555.."
```

*Figure 22. Non-conversational, Commit Mode=1, Synch Level=Confirm, and ACK (Transaction Terminated by Host Application Before Successful Completion)*

```
CLIENT                  FLOW              IMS CONNECT
REQUEST                                     REQUEST

  SEND----------------->IRM/TRAN/DATA ----------->RECEIVE
  RECEIVE<---------------DATA/CSM<----------------SEND
  SEND------------------->IRM/ACK---------------->RECEIVE

  SEND------------------->IRM/DATA-------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/ACK-------------->RECEIVE
                               .
                               .
                               .
  SEND------------------->IRM/DATA-------------->RECEIVE
  RECEIVE<----------------DATA/CSM<--------------SEND
  SEND-------------------->IRM/ACK-------------->RECEIVE

            Host Application abnormally terminates

  RECEIVE<------------------------------------ IMS MESSAGE "DFS555.."
```

*Figure 23. Conversational, Commit Mode=1, Synch Level=Confirm, and NAK (Transaction Terminated by Host Application)*

Table 46 and Table 47 show the required actions to be taken when different IMS `DFSnnnnn` messages or IMS command output is sent to the IMS Connect client. The two tables illustrate whether or not an ACK is required to be sent, both for synch level Confirm and synch level None, when the client receives an IMS DFS™ message or output from an IMS command.

**Note:** The client code can test the CSM_FLG1 byte for the presence of the CSM_ACK_NAK flag; it can also test the RSM_FLG1 byte for the presence of the RSM_ACK_NAK flag. It performs this test to determine if an ACK or NAK is required. Otherwise, it performs the analysis outlined in Table 46 and Table 47.

Table 46 and Table 47 also define whether or not the client requires a READ in order to receive the ″Deallocate Abort″ response (RSM) from IMS Connect. Notes for both tables immediately follow Table 47.

*Table 46. IMS Connect Client Message Protocol Sequence for IMS DFS Messages and IMS Command Output: Persistent Socket*

| | **Persistent Socket** | | | |
|---|---|---|---|---|
| | **Commit Mode 1** | | **Commit Mode 0** | |
| **Message Output to Client** | **Synch Level Confirm** | **Synch Level None** | **Synch Level Confirm** | **Synch Level None** |
| Invalid transaction code DFS0064 | DFS0064[1] | DFS0064[1] | N/A | N/A |
| Transaction stopped DFS0065 | DFS0065[1] | DFS0065[1] | N/A | N/A |
| Transaction abended DFS555 | DFS555[7] | DFS555[1] | N/A | N/A |
| Output DFS2082 | DFS2082[2] | DFS2082[1] | N/A | N/A |
| IMS Command Output | Cmd output[1] | Cmd output[1] | N/A | N/A |
| Security Failure DFS1292 | DFS1292[1] | DFS1292[1] | N/A | N/A |
| Segment greater than 32 K | DFS1294[5] | DFS1294[5] | N/A | N/A |

*Table 47. IMS Connect Client Message Protocol Sequence for IMS DFS Messages and IMS Command Output: Transaction Socket*

| | **Transaction Socket** | | | |
|---|---|---|---|---|
| | **Commit Mode 1** | | **Commit Mode 0** | |
| **Message Output to Client** | **Synch Level Confirm** | **Synch Level None** | **Synch Level Confirm** | **Synch Level None** |
| Invalid transaction code DFS0064 | DFS0064[1] | DFS0064[1] | DFS0064[1] | N/A |
| Transaction stopped DFS0065 | DFS0065[1] | DFS0065[1] | DFS0065[1] | N/A |
| Transaction abended DFS555 | DFS555[7] | DFS555[1] | DFS555[7] | N/A |

*Table 47. IMS Connect Client Message Protocol Sequence for IMS DFS Messages and IMS Command Output: Transaction Socket (continued)*

| | **Transaction Socket** | | | |
|---|---|---|---|---|
| | **Commit Mode 1** | | **Commit Mode 0** | |
| **Message Output to Client** | **Synch Level Confirm** | **Synch Level None** | **Synch Level Confirm** | **Synch Level None** |
| Output DFS2082 | DFS2082[2] | DFS2082[1] | No output[3] | N/A |
| IMS Command Output | Cmd output[1] | Cmd output[1] | Cmd output[4] | N/A |
| Security Failure DFS1292 | DFS1292[1] | DFS1292[1] | DFS1292[1] | N/A |
| Segment greater than 32 K | DFS1294[5] | DFS1294[5] | DFS1297[6] | N/A |

**Notes:**

1. Does not require an ACK to DFS messages.
2. Requires both an ACK to DFS messages and a second read to get a deallocate response.
3. The read to receive the transaction output will time out. No data will be received. OTMA treats commit mode=0 and Synch level=Confirm as asynchronous output. If the IMS Host application does not return a message (ISRT to IOPCB), OTMA does not send a deallocate. The TIMEOUT= value specified in the IMS Connect configuration file will have to expire before the disconnect is complete.
4. Requires an ACK to command output. A second read is not required to get a deallocate response. The command output gets treated as asynchronous output.
5. Does not require ACK to DFS1294 output. A second receive is required to receive the DFS555 message.
6. Client will receive DFS1297 rather than DFS1294. The DFS1294 message does not require an ACK. No DFS555 message gets sent, so a second receive is not required. The application is committed, and the application output gets discarded because the segment is larger than 32 K.
7. For IMS Versions 6 and 7, does not require an ACK to DFS messages.

   For IMS Version 5, requires an ACK to DFS messages. To receive the deallocate response (RSM), a second read is required.

For commit mode 1, there are three reason codes associated with a zero (0) return code, and two reason codes associated with an X’04’ return code, which provide information to the client application. The sample flows illustrate how each of these codes are used. The code meanings are listed in Table 48.

*Table 48. Information Reason Codes for Commit Mode=1, Synch Level=Confirm*

| Return Code | Reason Code | Description |
|---|---|---|
| X’00’ | 94 | Response - only output from host from non-conversation |
| X’00’ | 95 | Conversation - last output from host from on conversation |
| X’00’ | 96 | Conversation/response - middle of conversation |
| X’04’ | 97 | Deallocate commit - successful completion of host application |

*Table 48. Information Reason Codes for Commit Mode=1, Synch Level=Confirm (continued)*

| Return Code | Reason Code | Description |
|---|---|---|
| X’04’ | 98 | Deallocate abort - abnormal termination of host application |

# Chapter 11. Security Support

IMS Connect allows security support by checking the RACF flag. There are two ways to activate the RACF flag:

- Set the RACF flag in the configuration file, HWSCFG00, by setting the flag to Y as follows:

  ```
  HWS ID=HWS01 RACF=Y
  ```

- Use the HWS command `SETRACF` to set the RACF flag as follows:

  ```
  SETRACF ON
  ```

To check the setting of the RACF flag, you can issue the `VIEW HWS` command. After you issue this command, you should see: `HWSC0001 HWSID=HW01 RACF=Y`

If you turn the RACF flag ON, IMS Connect calls `RACROUTE REQUEST=VERIFY` to verify a user ID and password.

Security information is passed from clients in the IRM. See Table 7 on page 45 for the security field data, and see Appendix C, "HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 Security Actions," on page 251 for descriptions of the USERID and GROUPID results.

**In this chapter:**

- "RACF PassTicket Support"
- "SSL Connections" on page 129

## RACF PassTicket Support

An alternative to the RACF password is a PassTicket. PassTicket allows you to communicate with a host without using a RACF password. You can use PassTicket to authenticate user IDs and log on to computer systems that contain RACF.

You can select PassTicket support through an IMS Connect client and send a PassTicket in the IRM in place of a RACF password. IMS Connect issues a RACF call using PassTicket and blanks out the PassTicket field in the OTMA User Data Header before sending the message to IMS. Because PassTicket occupies the same field as the RACF password and PassTicket cannot be translated to uppercase, the RACF password is also not translated to uppercase. You can use a user message exit to provide uppercase translation.

The IMS Connect PassTicket support parallels IMS PassTicket support.

- You can use existing APPLID definitions for newly connecting IMS Connect clients.
- Each data store statement will have a new parameter `APPLID=`*`APPLname`*, where:
  - each `APPLID=` can be a unique RACF APPLname for each data store
  - each `APPLID=` can be the same name for each data store, as required for VGR support, or can be unique per data store
- The default `APPLID=`*`APPLname`* value is blank.
- The IMS Connect client can pass an `APPLID` in the IRM to the user message exit which sets the `APPLID` in the OTMA User Data Header or the user message exit can pass and set the appropriate `APPLID` in the OTMA User Data Header.

HWSIMSO0 and HWSIMSO1 do not support passing an APPLID in the IRM. However, they do support passing PassTicket in the IRM. The APPLID used by HWSIMSO0 and HWSIMSO1 must be defined on the DATASTORE statement.

For PassTicket support, you are responsible for all definitions to RACF. You need to establish the RACF encoding and decoding routines and to supply the encoding routine to the distributed platform.

RACF PassTicket is only supported for customer-written client/server support. This support will eventually be extended to IMS Connector for Java.

This support may require changes to the customer-written user message exits and customer-provided Client/Server code. The following list describes options you may select for PassTicket support:

- **Support for passing an APPLname in the IRM to IMS Connect**

  This support has been added to the IRM definition. A new 8 byte field, IRM_APPL_NM, has been added to the end of the IRM structure. If you want to implement the PassTicket function, then the client code must pass the APPLname to IMS Connect in this field.

  **Note:** This **will** change the length of the IRM by 8 bytes and the total length of the message by 8 bytes.

  The supplied user message exits (HWSSMPL1 and HWSSMPL0) have been modified so that a client can send an APPLname to IMS Connect in the IRM_APPL_NM field.

  If you choose this option, the only action you need to do is to pass the APPLname in the IRM. HWSIMSCB and IMS Connect have been modified to support this function.

- **No support for passing an APPLname in the IRM to IMS Connect**

  This support has been added to the IRM definition. A new 8 byte field IRM_APPL_NM has been added to the end of the IRM structure. If you do not want to implement the PassTicket function, you have two options:

  – **Option 1: Blank APPLname**

    You can choose to pass a blank APPLname to IMS Connect in the IRM_APPL_NM field to IMS Connect.

    **Note:** This **will** change the length of the IRM by 8 bytes and the total length of the message by 8 bytes.

    The supplied user message exits (HWSSMPL1 and HWSSMPL0) have been modified so that a client can send a blank APPLname in the IRM_APPL_NM field to IMS Connect.

    If you choose this option, the only action you need to do is to pass a blank APPLname in the IRM. HWSIMSCB and IMS Connect have been modified to support this blank APPLname function.

  – **Option 2: No APPLname**

    The customer can choose to pass no APPLname to IMS Connect in the IRM_APPL_NM field to IMS Connect.

    **Note:** This **will not** change the length of the IRM or the total length of the message.

    The supplied user message exits (HWSSMPL1 and HWSSMPL0) have been modified so that a client does not have to send an APPLname in the IRM_APPL_NM field to IMS Connect.

User exits, HWSIMSO0 and HWSIMSO1 do not support PassTicket with APPLname in the IRM.

If you choose this option, you do not need to perform any action. HWSIMSCB and IMS Connect have been modified to support this function of not passing an APPLname.

### PassTicket Replay Protection Considerations

You may want to consider bypassing PassTicket replay protection if you have multiple end-users sharing the same user ID. If you have multiple users with the same user IDs, it is possible for them to request access to an application during the same time interval. In this situation, the same PassTicket is generated for different users. As a result, if PassTicket replay protection is not bypassed, the users will be using the same PassTicket and be denied access to the application. Bypassing the PassTicket replay protection allows the same PassTicket to be used by multiple users.

Similarly, if you are stress testing your system where there is no think time driving requests to IMS Connect and have numerous requests to the same application occurring in the same time interval, you may want to consider bypassing PassTicket replay protection. This option allows the same PassTicket to be used within a ten minute period.

You can specify NO REPLAY PROTECTION in the APPLDATA field of the PTKTDATA profile for one or more of the selected applications to allow the same PassTicket to be generated within a ten minute period.

For additional information about no replay options, see *z/OS Security Server RACF Security Administrator’s Guide* (SA22-7683), Chapter 7: Protecting General Resources.

## SSL Connections

TCP/IP consistently and reliably transfers information across the internet domain, but it does not secure the information that is transferred.

IMS Connect supports Secure Sockets Layer (SSL) Version 2.0, Version 3.0, and Transport Layer Security (TLS) Version 1.0. TLS V1.0 is the latest version of the secure sockets layer protocol. (Throughout this book, the term SSL is used to describe both the SSL and TLS protocols.) SSL protects information from:

- Eavesdropping
- Data theft
- Traffic analysis
- Data modification
- Trojan horse browser / server

SSL ensures the transfer of sensitive information over the internet by securing sockets through a combination of public and private and symmetric key encryption. The public and private keys are used to initiate contact between the client and the server and to establish authentication between one another. During this handshake protocol, the client and server agree on how to encrypt and decrypt information and define the format used to transmit the encrypted data. Symmetric key encryption is used to encrypt and decrypt all of the data transferred between the client and the server.

X.509 certificates are used by both the client and server when securing communications. The client must verify the server's certificate based on the certificate of the Certificate Authority (CA) that signed the certificate or based on a self-signed certificate from the server. The server must verify the client's certificate (if requested) using the certificate of the CA that signed the client's certificate. The client and the server then use the negotiated session keys and begin encrypted communications.

## z/OS Key Management

SSL connections use public and private key mechanisms for authenticating each side of the SSL session (the server side and the client side) and agree on bulk encryption keys to be used for the SSL session. To use public and private key mechanisms (PKIs), public and private key pairs must be generated. In addition, X.509 certificates (which contain public keys) may either need to be created, or certificates must be requested, received, and managed.

SSL for z/OS supports the following two methods for managing PKI private keys and certificates:

- A z/OS shell-based program named *gskkyman*. gskkyman creates, fills in, and manages a z/OS HFS file that contains PKI private keys, certificate requests, and certificates. This z/OS HFS file is called a key database and, by convention, has a file extension of .kdb.
- The z/OS Security Server (RACF) RACDCERT command. The RACDCERT command installs and maintains PKI private keys and certificates in RACF. See *z/OS: Security Server RACF Command Language Reference*, SA22-7687 for details about the RACDCERT command.

  RACF supports the management of multiple PKI private keys and certificates as a group. These groups are called key rings. RACF key rings are the preferred method for managing PKI private keys and certificates for SSL.

For more information about z/OS and SSL, see *z/OS System Secure Sockets Layer Programming*, SC24-5901-02.

## SSL Initialization

The TCP/IP variable SSLENVAR points to a member file that contains SSL initialization information. You can add comments and multiline variable assignments in the input file. Specify comments by placing a number sign (#) at the beginning of a line. Create multiline variable assignments by placing a dash (-) at the end of the line. Following is an example of an input file (this is also the default setup of SSL):

```
###################################################
#   This is my SSL interface configuration file    #
###################################################
GSK_PROTOCOL_SSLV2=GSK_PROTOCOL_SSLV2_ON
GSK_PROTOCOL_SSLV3=GSK_PROTOCOL_SSLV3_ON
GSK_PROTOCOL_TLSV1=GSK_PROTOCOL_TLSV1_ON
GSK_KEYRING_FILE=IMSCONNECT
GSK_KEYRING_LABEL=IMS CONNECT
GSK_KEYRING_PW=
GSK_KEYRING_STASH_FILE=
GSK_CLIENT_AUTH_TYPE=GSK_CLIENT_AUTH_FULL_TYPE
GSK_SESSION_TYPE=GSK_SERVER_SESSION
GSK_V2_CIPHER_SPECS=642
GSK_V3_CIPHER_SPECS=0906030201
```

The variable assignment format is:

```
<variable name>=<value>
```

The possible variables and the values associated with the variables are as follows:

**GSK_KEYRING_FILE**
Name of the key database or RACF keyring. If a RACF key ring is specified, it must be an existing key ring and the current user ID must have READ access to the IRR.DIGTCERT.LISTRING and the IRR.DIGCERT.LIST resources in the FACILITY class.

**GSK_KEYRING_LABEL**
Label name in the key database file of RACF key ring used. If this is not set, or set to NULL, the default key database or key ring entry is used.

**GSK_KEYRING_PW**
Password of the key database. This must be NULL when a RACF key ring is used or when a stash file is specified.

**GSK_KEYRING_STASH_FILE**
Name of the file that contains the password for the keyring. This value must be NULL when a RACF key ring is used.

**GSK_V2_CIPHER_SPECS**
A null-terminated character string which specifies the ciphers to enable for SSL V2.0. If this is not specified, the default cipher spec list is used. The default list is 713642 if the Security Level 3 update is installed and 624 if it is not installed.

1 - RC4 US

2 - RC4 Export

3 - RC2 US

4 - RC2 Export

6 - DES 56-bit Export

7 - Triple DES US

Usage example: GSK_V2_CIPHER_SPECS=6321

**GSK_V3_CIPHER_SPECS**
A null-terminated character string that specifies the ciphers to enable for SSL V3.0 and TLS 1.0. If no value is specified, the default cipher spec list is used. The default list is 05040A0306090201 if the Security Level 3 update is installed and 0306090201 if it is not installed.

01 - NULL MD5

02 - NULL SHA

03 - RC4 MD5 Export

04 - RC4 MD5 US

05 - RC4 SHA US

06 - RC2 MD5 Export

09 - DES SHA Export

0A - Triple DES SHA US

Usage example: GSK_V3_CIPHER_SPECS=0306090201

**GSK_PROTOCOL_SSLV2**
Used to enable or disable SSL V2.0. Possible values are GSK_PROTOCOL_SSL_V2_ON and GSK_PROTOCOL_SSLV2_OFF.

**Note:** All SSL V2.0 non-US encryption schemes have been decrypted. Therefore, SSL V2.0 should not be enabled unless the client does not support SSL V3.0 or TLS V1.0 communication.

**GSK_PROTOCOL_SSLV3**
Used to enable or disable SSL V3.0. Possible values are GSK_PROTOCOL_SSLV3_ON and GSK_PROTOCOL_SSLV3_OFF.

**GSK_PROTOCOL_TLSV1**
Used to enable or disable TLS V1.0. Possible values are GSK_PROTOCOL_TLSV1_ON and GSK_PROTOCOL_TLSV1_OFF.

**GSK_CLIENT_AUTH_TYPE**
Indicates the type of client authentication to take place. Two options are available: GSK_CLIENT_PASSTHRU_TYPE and GSK_CLIENT_AUTH_FULL_TYPE. The GSK_CLIENT_PASSTHRU_TYPE specifies to not authenticate if the client sends a certificate. GSK_CLIENT_AUTH_FULL_TYPE validates all received certificates. If the certificate cannot be validated, the connection is terminated. If no certificate is sent by the client, the connection is unsuccessful.

**GSK_SESSION_TYPE**
Indicates whether or not to require client authentication. A value of GSK_SERVER_SESSION does not require authentication. GSK_SERVER_SESSION_WITH_CL_AUTH does require client authentication.

**GSK_V2_SESSION_TIMEOUT**
The number of seconds before the SSL V2.0 session identifier expires. The valid range is from 0 to 100 seconds. If the session timeout value has not expired, the client and server, as well as peer clients (multiple client connections from same client computer) do not need to perform a handshake when starting a new connection.

**GSK_V3_SESSION_TIMEOUT**
The number of seconds before the SSL V3.0 session identifier expires. The valid range is from 0 to 100 seconds. If the session timeout value has not expired, the client and server as well as peer clients (multiple client connections from same client computer) do not need to perform a handshake when starting a new connection.

**GSK_V2_SIDCACHE_SIZE**
The maximum number of session ID elements that can be stored in the SSL V3.0 cache. The range is 0 to 32000 entries.

**DEBUG_SSL**
Indicates whether or not to turn on SSL debugging information. If the debug information is requested, it can be found in the job output after the IMS Connect job has completed. Possible assignment values are ON and OFF.

Usage example: DEBUG_SSL=ON

## SSL Default Setup

If the SSL initialization file does not exist, a default setup of SSL occurs. The default setup variables and their values are:

```
GSK_PROTOCOL_SSLV2=GSK_PROTOCOL_SSLV2_ON
GSK_PROTOCOL_SSLV3=GSK_PROTOCOL_SSLV3_ON
GSK_PROTOCOL_TLSV1=GSK_PROTOCOL_TLSV1_ON
GSK_KEYRING_FILE=IMSCONNECT
GSK_KEYRING_LABEL=IMS CONNECT
GSK_KEYRING_PW=
GSK_KEYRING_STASH_FILE=
GSK_CLIENT_AUTH_TYPE=GSK_CLIENT_AUTH_FULL_TYPE
GSK_SESSION_TYPE=GSK_SERVER_SESSION
GSK_V2_CIPHER_SPECS=642
GSK_V3_CIPHER_SPECS=0906030201
```

Note: When creating a new PROCLIB file, ensure that sequence numbers are not automatically inserted in the SSL configuration file. The sequence numbers will cause parsing errors of the SSL options.

## SSL Sample JCL

RACF as a security manager can create certificates and keyrings, authorize certificates, and store the certificates in a keyring. RACF can also be configured to be a certificate authority. The following sample JCL illustrates how to set up keyrings and certificates in RACF:

```
//SSLRACF JOB MSGLEVEL=(1,1),USER=OMVSADM,PASSWORD=,
//        CLASS=A,MSGCLASS=A
//*
/*ROUTE PRINT THISCPU/CSDM09
//   EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTSIN DD *
* Remove labels from mykey and delete the key.
RACDCERT REMOVE(LABEL('CLIENT') RING(mykey))
RACDCERT DELETE(LABEL('CLIENT'))
RACDCERT DELRING(mykey)
RACDCERT LIST(LABEL('CLIENT')
RACDCERT LISTRING(mykey)
* Remove labels from SSLRING and delete the key.
RACDCERT REMOVE(LABEL('CONNECT') RING(SSLRING))
RACDCERT DELETE(LABEL('CONNECT'))
RACDCERT REMOVE(LABEL('CERTAUTH') RING(SSLRING))
RACDCERT DELETE(LABEL('CERTAUTH'))
RACDCERT DELRING(SSLRING)
RACDCERT LIST(LABEL('CONNECT')
RACDCERT LIST(LABEL('CERTAUTH')
RACDCERT LISTRING(SSLRING)
* Create CLIENT certificate, export to dataset, and connect to mykey.
RACDCERT GENCERT-
 SUBJECTSDN(CN('CLIENT') OU('IMS') O('IBM') C('US'))-
 SIZE(512) WITHLABEL('CLIENT')
RACDCERT EXPORT(LABEL('CLIENT')) DSN(CLIENT.CERT) FORMAT(CERTB64)
RACDCERT ADD(CLIENT.CERT) TRUST WITHLABEL('CLIENT')
RACDCERT ADDRING(mykey)
RACDCERT CONNECT(LABEL('CLIENT') DEFAULT RING(mykey))
RACDCERT LISTRING(mykey)
RACDCERT LIST(LABEL('CLIENT')
* Create SSLRING.
RACDCERT ADDRING(SSLRING)
* Create CERTAUTH certificate, export to dataset, and connect to SSLRING
RACDCERT CERTAUTH GENCERT-
 SUBJECTSDN(CN('CERTAUTH') OU('IMS') O('IBM') C('US'))-
 KEYUSAGE(CERTSIGN) WITHLABEL('CERTAUTH')
```

```
RACDCERT CERTAUTH EXPORT(LABEL('CERTAUTH')) DSN(CERTAUTH.CERT)
RACDCERT CONNECT(CERTAUTH LABEL('CERTAUTH') RING(SSLRING))
* Create CONNECT certificate, export to dataset, and connect to SSLRING.
RACDCERT GENCERT-
 SUBJECTSDN(CN('CONNECT') OU('IMS') O('IBM') C('US'))-
 WITHLABEL('CONNECT')-
 SIGNWITH(CERTAUTH LABEL('CERTAUTH'))
RACDCERT EXPORT(LABEL('CONNECT')) DSN(CONNECT.CERT)
RACDCERT CONNECT(LABEL('CONNECT') DEFAULT RING(SSLRING))
* ?????
RACDCERT ADD(CLIENT.CERT) TRUST WITHLABEL('IMS CONNECT CLIENT')
* ?????
RACDCERT LIST(LABEL('CONNECT')
RACDCERT CERTAUTH LIST(LABEL('CERTAUTH')
SETROPTS RACLIST(DIGTCERT) REFRESH
RDELETE FACILITY (IRR.DIGTCERT.LIST    IRR.DIGTCERT.LISTRING)
RDEFINE  FACILITY  IRR.DIGTCERT.LISTRING   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.LIST       UACC(NONE)
PERMIT   IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(OMVSADM) ACCESS(ALTER)
PERMIT   IRR.DIGTCERT.LIST     CLASS(FACILITY) ID(OMVSADM) ACCESS(ALTER)
SETROPTS CLASSACT(FACILITY)
//
```

In this sample JCL, SSLRING is the sample keyring name and CONNECT is the sample certificate name. RACF stores the certificates in the OMVSADM data set. You must provide a copy of OMVSADM.CERTAUTH.CERT to the client's keyring so RACF can authorize the client.

# Chapter 12. Ping Support

To determine whether or not IMS Connect is available, you can send a ping request to IMS Connect. The ping support operates like a transaction and has the appearance of a transaction code and data. When you send the request `PING IMS_CONNECT`, the response is `PING RESPONSE`. The client sequence is:

1. Connect.
2. Send `PING IMS_CONNECT` (must be sent in uppercase).
3. Receive `PING RESPONSE`.
4. Disconnect.

The user message exits, HWSSMPL1, HWSSMPL0, and HWSJAVA provide ping support. User message exits, HWSCSLO0, HWSCSLO1, HWSIMSO0, and HWSIMSO1 do not support the ping function. If you write your own user message exit, you can choose to add the ping function support in your exit.

# Chapter 13. IMS Connect Commands

This chapter describes the IMS Connect commands.

All IMS Connect commands must be immediately preceded on the command line of the MVS system console by the reply number of the outstanding IMS Connect reply message (for example, `nnHWSCMD` where *nn* is the reply number).

**In this chapter**:

## CLOSEHWS

The `CLOSEHWS` command terminates IMS Connect.

**Parameters** *quiesce*

Specifies that termination is to end all client and datastore connections in a controlled manner. If no parameter is specified for `CLOSEHWS`, this parameter is used by default.

All work that is currently in progress, or that is queued for processing, is completed before IMS Connect is terminated. No new work is accepted after this command has been entered and accepted.

IMS Connect shuts down in the following order:

1. All active units of work for clients/browsers are completed.
2. Communication between IMS Connect and IMS is terminated.
3. IMS Connect terminates.

*force*

Specifies that termination is to end all client and datastore connections immediately, which forces any IMS applications that are executing for the connected clients to abnormally terminate.

**Usage** Use this option to terminate IMS Connect.

Using the FORCE parameter will terminate client and datastore activity immediately. Using the QUIESCE parameter enables client and IMS Host applications to execute to completion. You can issue a CLOSEHWS FORCE command after issuing a CLOSEHWS QUIESCE command.

**Example** To close IMS Connect, you can use any one of the following command sequences:

```
nnCLOSEHWS QUIESCE
nnCLOSEHWS FORCE
nnCLOSEHWS QUIESCE
```

followed by

```
nnCLOSEHWS FORCE
```

**Note:** If you use the MVS CANCEL command to terminate IMS Connect, the MVS cancel command functions as follows:

```
CANCEL ims_connect_jobname,dump
```

or

```
CANCEL ims_connect_jobname
```

The results of the MVS cancel command can leave IMS Connection functions, such as RRS, in unknown states. Instead, the MVS STOP command is recommended. The MVS STOP command functions as follows:

```
STOP ims_connect_jobname,dump
```

or

```
STOP ims_connect_jobname
```

The results of the MVS STOP command are the same as the IMS Connect `CLOSEHWS QUIESE` command.

## OPENDS or STARTDS

The `OPENDS` or `STARTDS` command starts communication between the IMS Connect and a datastore.

**Parameters** *datastore_id*

Specifies the name of the datastore. This name must be defined to the IMS Connect through the configuration member `HWSCFGxx`, and must match one of the IDs that is defined in the DATASTORE configuration statement or statements.

**Usage** Use this command to reestablish communication with a datastore after communication fails between the IMS Connect and the datastore. For example, use this command to restart communication when all activity for the datastore in the IMS Connect is terminated, or after a `STOPDS` command has terminated communication with the datastore.

Use the `VIEWDS` command to display information about datastores if you are not sure about the activity of a particular datastore.

The `OPENDS` or `STARTDS` command does not affect a datastore that is already active or a datastore that is not defined to the IMS Connect in the configuration member `HWSCFGxx`.

**Example**

To open communication to datastore IMSA:

```
nnOPENDS IMSA
```

or

```
nnSTARTDS IMSA
```

## OPENIP or STARTIP

The `OPENIP` or `STARTIP` command starts communication between the IMS Connect and the IMSplex that contains OM which is connected to SCI.

**Parameters** *imsplex_id*

Specifies the name of the IMSplex. This name must be defined to the IMS Connect through the configuration member HWSCFGxx, and must match the TMEMBER that is defined in the IMSplex configuration statement.

**Usage** Use this command to reestablish communication with the IMSplex that is being used to communicate with OM if communication has failed between IMS Connect and the IMSplex. For example, use this command to restart communication when all activity for the IMSplex in the IMS Connect is terminated, or after a `STOPIP` command has terminated communication with the IMSplex IMS that contains OM.

Use the `VIEWIP` command to display information about the IMSplex if you are not sure about the activity of the IMSplex.

The `OPENIP` or `STARTIP` command does not affect the IMSplex if the IMSplex is already active or if the IMSplex is not defined to IMS Connect in the configuration member HWSCFGxx.

**Example**

To open communication to IMSplex with TMEMBER name of IMSPLEX1:

```
nnOPENIP IMSPLEX1
```

or

```
nnSTARTIP IMSPLEX1
```

## OPENPORT or STARTPT

The `OPENPORT` or `STARTPT` command reestablishes IMS Connect communication with TCP/IP to allow listening on TCP/IP ports.

**Parameters** *portid*

Identifies the number of the port to be opened. This port number must match one of the port numbers that is defined in the `PORTID` substatement of the `TCPIP` configuration statement in the `HWSCFGxx` configuration member. For the local option port, specify a portid value of LOCAL.

**Usage** Use this command to reestablish a TCP/IP connection to allow listening on a TCP/IP port. Use this command when communication stops between the IMS Connect and a TCP/IP port, but the IMS Connect has not terminated.

**Example** To reestablish the TCP/IP connection between the IMS Connect and port 9999 so that the IMS Connect can listen on that port:

```
nnOPENPORT 9999
```

or

```
nnSTARTPT 9999
```

## RECORDER

The `RECORDER` command opens and closes the line trace data set.

**Parameters** *open*

*close*

**Usage** Use this command to open or close the line trace data set, HWSRCDR.

**Example**

```
nnRECORDER OPEN
nnRECORDER CLOSE
```

## SETRACF

The `SETRACF` command turns on and off the RACF flag.

**Parameters** *ON/OFF*

Identifies if the RACF flag is turned on or off.

**Usage** To enable or disable the RACF user identification and verification.

**Example** To turn on the RACF:

```
nnSETRACF ON
```

## SETRRS

The `SETRRS` command enables or disables communication between IMS Connect and RRS.

**Parameters**
*ON* or *OFF*

Identifies whether or not to enable RRS communication.

**Usage** To enable or disable communication between IMS Connect and RRS. RRS is required for two-phase-commit support.

**Example**
To disable communication between IMS Connect and RRS:

```
nnSETRRS OFF
```

## STOPCLNT

The `STOPCLNT` command immediately terminates communication with a client using a specific TCP/IP port.

**Parameters** *portid*

Identifies the port that the client is using for the TCP/IP connection with the IMS Connect. This port number must match a port number that is defined in the `PORTID` substatement of the `TCPIP` configuration statement in the `HWSCFGxx` configuration member. For the local option port, specify a portid value of LOCAL.

*clientid*

Specifies the name of the client (the client name is dynamically generated by IMS Connector for Java).

**Usage** Work currently in progress for that client is ended.

Use this command whenever a client is unable to accept response messages being sent to it, or when a client is waiting for a nonexistent response message (for example, when an error occurred that caused a response message to be lost before it was sent back to the client).

Use the `VIEWPORT` command to display the name and state of the client.

**Example** To force the IMS Connect to terminate communication with client `CLIENT01`, who is communicating with the IMS Connect using port `9999`:

```
nnSTOPCLNT 9999 CLIENT01
```

## STOPDS

The `STOPDS` command immediately terminates communication between the IMS Connect and a datastore.

**Parameters** *datastore_id*

Specifies the name of the datastore. This name must match an `ID` that is defined in a `DATASTORE` configuration statement of the `HWSCFGxx` configuration member.

**Usage** Work currently in progress for a datastore is ended and communications with that datastore and its threads are terminated. Messages that are queued for the datastore are released and the originator of the queued messages is notified. No new messages are accepted after the `STOPDS` command is accepted.

Use this command to release messages that are queued for an unavailable datastore or for a datastore whose queued work belongs to unavailable clients. It can also be used for any type of error situation that requires immediate termination of communication with a datastore.

Use the `OPENDS` command to open communication with the datastore at a later time.

**Example** To stop communication to datastore IMSA:

```
nnSTOPDS IMSA
```

## STOPIP

The `STOPIP` command stops communication between the IMS Connect and the IMSplex that contains OM which is connected to SCI.

**Parameters** *imsplex_id*

Specifies the name of the IMSplex. This name must be defined to the IMS Connect, through the configuration member HWSCFGxx, and must match the TMEMBER that is defined in the IMSplex configuration statement.

**Usage** Work currently in progress for the IMSplex has ended and communication with the IMSplex and its threads are terminated. Commands that are queued for the Control Center are unavailable. `STOPIP` can also be used for any error situation that requires immediate termination of communication with the IMSplex.

Use this command to release IMS Control Center commands that are queued for an unavailable IMSplex or for the IMSplex whose queued work belongs to unavailable Control Center clients. The `STOPIP` command can also be used for any error situation that requires immediate termination of communication with the IMSplex.

Use the `OPENIP` command to start communication with the IMSplex at a later time.

**Example** To stop communication to IMSplex with TMEMBER name of IMSPLEX1:

```
nnSTOPIP IMSPLEX1
```

## STOPPORT

The `STOPPORT` command immediately terminates listening on a TCP/IP port.

**Parameters** *portid*

Identifies the number of the port on which listening is to stop. This port number must match one of the port numbers that is defined in the `PORTID` substatement of the `TCPIP` configuration statement in the `HWSCFGxx` configuration member. For the local option port, specify a portid value of LOCAL.

**Usage**

Work currently in progress is allowed to continue for existing clients. Only the listening for new request messages on the port is terminated immediately. When existing work has completed, the port is no longer active.

Use the `VIEWPORT` command to display the state of the port and any clients using that port.

**Example** To stop listening on port `9999`:

```
nnSTOPPORT 9999
```

## VIEWDS

The `VIEWDS` command displays the current activity of a datastore.

**Parameters** *datastore_id*

Specifies the name of the datastore for which information is to be displayed or `ALL`. If a datastore name is used, this name must match the `ID` parameter of a `DATASTORE` configuration statement of the `HWSCFGxx` configuration file and only the information for this datastore is displayed. If `ALL` is used, information for all datastores that are defined in a `DATASTORE` configuration statement in the `HWSCFGxx` configuration member is displayed.

**Usage**
This command displays the current information for one or all datastores. The information displayed for each datastore is:

**Datastore Name**
Name of the datastore, as defined in the ID substatement of the `DATASTORE` configuration statement in the IMS Connect configuration member `HCTCFGxx`.

**Status**
State of the datastore, `ACTIVE`, `NOT ACTIVE` or `DISCONNECT`.

If the datastore goes down, IMS Connect is notified (from IMS OTMA through XCF) of the status of the datastore. When the datastore is brought back up and restarted, IMS Connect is notified and automatically reconnects to the datastore.

**Group**
XCF group name for the group to which the IMS Connect and IMS OTMA belong.

**Member**
IMS Connect member name in the XCF group listed.

**Target Member**
IMS OTMA member name in the XCF group listed.

**Example**
To view the information for a single datastore, IMSA:

```
nnVIEWDS IMSA
```

To view the information for all datastores defined to IMS Connect:

```
nnVIEWDS ALL
```

## VIEWHWS

The `VIEWHWS` command displays the current activity of IMS Connect.

**Parameters**
None.

**Usage**
Information displayed includes:

**HWS ID=**
Name of the IMS Connect, as defined in the `ID` substatement of the `HWS` configuration statement in the `HWSCFGxx` configuration member.

**DATASTORE=**
Name of the datastore, as defined in the `ID` substatement of the `DATASTORE` configuration statement in the `HWSCFGxx` configuration member or `No active Datastores`.

**IMSPLEX=**
Name of the IMSplex as defined in the TMEMBER parameter of the IMSplex configuration statement in the IMS Connect configuration member HWSCFGxx.

**STATUS=(DATASTORE= or IMSPLEX=)**
State of the datastore or IMSplex, whether `ACTIVE`, `NOT ACTIVE`, or `DISCONNECT`.

**GROUP=**
XCF group name for the group to which the IMS Connect and IMS OTMA belong.

**MEMBER=**
IMS Connect member name in the XCF group list for a datastore.
or
IMS Connect member name as defined in the IMS Connect configuration file IMSplex statement for MEMBER= parameter.

**TARGET MEMBER=**
IMS OTMA member name in the XCF group list.
or
IMS Connect target member name as defined in the IMS Connect configuration file IMSplex statement for TMEMBER= parameter.

**PORT=**
Identifies the port or ports that are defined in the `PORTID` substatement of the `TCPIP` configuration statement in the `HWSCFGxx` configuration member or `No active Ports`.

**STATUS=(for PORT=)**
State of the port, whether `ACTIVE` or `INACTIVE`.

**RRS=**
Specifies if RRS is set to Y or N in the HWS configuration file.

**STATUS= (for RRS=)**
State of RRS. The RRS state can be one of the following:

- `ACTIVE` - IMS Connect restart with RRS has completed.
- `NOT ACTIVE` - IMS Connect has not registered with RRS.
- `REGISTERED` - IMS Connect has registered with RRS.

The following keywords are header settings of the output.

**CLIENT**
Specifies the name of the client or `NO active Clients`.

**USERID**
Specifies the USERID name passed to IMS Connect.

**TRAN CODE**
Specifies the transaction code submitted by the client.

**STATUS (for current CLIENTID)**
State of the client's thread. The client thread state can be one of the following values:

- `RECV` - Waiting for input from client (in other words, in a receive state).
- `CONN` - Waiting for output from IMS.
- `XMIT` - Sending data to client.
- `CONV` - In a conversational state.
- `WFCM` - Waiting for confirmation (ACK, NAK, or DEALLOCATE) from client.

**SECONDS**
Number of seconds that the client has been in the specified status.

**IP ADDRESS**
IP address being used by the connection of the client to IMS Connect. If IPV6 is enabled, the IP address format consists of eight hexadecimal numbers divided by colons. If IPV6 is not enabled, the IP address format of IPV4 is used. The following example is for an IPV6 IP address displayed using IPV6 format:

```
FEDC:ABCD:2222:3333:FEDC:DB55:6666:3322
```

The following example is for an IPV4 IP address displayed using IPV6 format:

```
0:0:0:0:0:FFFF:945:33FF
```

For more information on the IP address format for IPV6, see *IPv6 Network and Application Design Guide*.

**CLIENT PORT**
A random number that TCP/IP generates to represent a connection from a client.

You can use the `VIEWDS` command to display information for datastores only or the `VIEWPORT` command to display information for ports only.

**Example** To view the IMS Connect:

```
nnVIEWHWS
```

## VIEWIP

The `VIEWIP` command displays the current activity for the IMSplex.

**Parameters** *imsplex_id*

Specifies the name of the IMSplex for which information is to be displayed. If the IMSplex name is used, this name must match the ID parameter of the IMSplex configuration statement in the HWSCFGxx.

**Usage** The `VIEWIP` command displays the current information for the IMSplex. The information displayed for the IMSplex is:

**IMSPLEX=**
Name of the IMSplex, as defined in the ID parameter of the IMSplex configuration statement in the IMS Connect configuration member, HWSCFGxx.

**STATUS=**
State of the IMSplex, `ACTIVE`, `NOT ACTIVE`, or `DISCONNECT`.

If the IMSplex does down, IMS Connect is notified (through SCI) of the status of the IMSplex. When the IMSplex is brought back up and restarted, IMS Connect is notified and automatically reconnects to IMSplex.

**MEMBER=**
Name of the member as defined in the Member parameter

of the IMSplex configuration statement in the IMS Connect configuration member, HWSCFGxx.

**TARGET MEMBER=**
Name of the target member of the IMSplex SCI to which IMS has connected and defined in the TMEMBER parameter of the IMSplex configuration statement in the IMS Connect configuration member, HWSCFGxx.

**Example** To view the information for the IMSplex, with TMEMBER name of IMSPLEX1:

```
nnVIEWIP IMSPLEX1
```

## VIEWPORT

The `VIEWPORT` command displays the current activity of a port.

**Parameters** *portid*

Specifies either the specific port for which information is to be displayed, or ALL, for all ports. If a port number is specified, the number must match a port number already defined in the PORT substatement of the TCPIP configuration statement in the IMS Connect configuration member. Information displayed is for that port only. If LOCAL is specified, information displayed is for all clients using local port communications through IMS Connector for Java. If ALL is specified, information displayed is for all ports defined in the TCPIP configuration member.

**Usage** Information displayed includes:

**PORT=**
Identifies the port or ports that are defined in the `PORTID` substatement of the `TCPIP` configuration statement in the `HWSCFGxx` configuration member.

**STATUS=(for PORT=)**
State of the port, whether `ACTIVE` or `INACTIVE`.

The following keywords are headers for displayed output:

**CLIENTID**
Specifies the name of the client or `NO active Clients`.

**USERID**
Specifies the USERID name passed to IMS Connect.

**TRAN CODE**
Specifies the transaction code submitted by the client.

**STATUS (for displayed CLIENTID)**
State of the client's thread. The client thread state can be one of the following values:

- `RECV` - Waiting for input from client (in other words, in a receive state).
- `CONN` - Waiting for output from IMS.
- `XMIT` - Sending data to client.
- `CONV` - In a conversational state.
- `WFCM` - Waiting for confirmation from client.

**SECONDS**
Number of seconds that the client has been in the specified status.

**IP ADDRESS**
IP address being used by the connection of the client to IMS Connect. If IPV6 is enabled, the IP address format consists of eight hexadecimal numbers divided by colons. If IPV6 is not enabled, the IP address format of IPV4 is used. The following example is for an IPV6 IP address displayed using IPV6 format:

```
FEDC:ABCD:2222:3333:FEDC:DB55:6666:3322
```

The following example is for an IPV4 address displayed using IPV6 format:

```
0:0:0:0:0:FFFF:945:33FF
```

For more information on the IP address format for IPV6, see *IPv6 Network and Application Design Guide* (SC31-8885-00).

**CLNTPORT**
A random number that TCP/IP generates to represent a connection from a client.

**Example**
To view the information for a single port, `9999`:

```
nnVIEWPORT 9999
```

To view the information for all ports defined to the IMS Connect:

```
nnVIEWPORT ALL
```

## VIEWUOR

The `VIEWUOR` command displays the current status of a specific unit of recovery identifier (URID) or all URIDs in IMS Connect.

**Parameters**
*URID* or *ALL*

Specifies the status of a unit of recovery identifier or all unit recovery identifiers to be displayed.

**Usage**
Information displayed includes:

**URID** Identifies the 16-byte character string of a specific unit-of-recovery identifier.

**STATE**
State of the UR. The UR state can be one of the following values:

- `IN_RESET` - The UR is starting and has not yet changed any resources.
- `IN_FLIGHT` - The UR can access resources and has the potential to change resources, but the changes are not committed.
- `IN_STATE_CHECK` - The UR issues a commit and waits for the resource manager's STATE_CHECK exit routine to check if the resources are in the correct state.

- `IN_PREPARE` - The UR in the proper state issues a commit and RRS invokes the PREPARE exit routine.
- `IN_DOUBT` - RRS is waiting for the resource manager to tell it whether to resolve the UR by a commit or by a backout.
- `IN_COMMIT` - One of the following actions occurred:
  - The PREPARE exit routines replied YES.
  - The DSRM or SDSRM told RRS to commit an IN_DOUBT UR.
  - The installation used the RRS panels to commit an IN_DOUBT UR.
- `IN_BACKOUT` - One of the following actions occurred:
  - One or more PREPARE exit routines replied NO.
  - The application issued a backout.
  - The DSRM or SDSRM told RRS to back out an IN_DOUBT UR.
  - The installation used the RRS panels to back out an IN_DOUBT UR.
  - Before phase 2 of the two-phase-commit protocol, the system, application, RRS, or a resource manager failed.
- `IN_END` - The resources have been updated.
- `IN_ONLY_AGENT` - Only one resource manager expressed interest in the UR.
- `IN_COMPLETION` - The resources have been updated and RRS has completed processing the UR.
- `IN_FORGET` - During distributed processing, the UR has completed but RRS is waiting for the SDSRM to indicate how long to process the log records for the UR.
- `FORGOTTEN` - The UR has completed, and RRS has deleted its log records.

**XID** (X/Open identifier) Identifies the distributed transaction used by the X/Open architecture. The XID is comprised of four parts:
- `FMID` - 4-byte fixed format id
- `GTRID` - 4-byte fixed GTRID length
- `BQUAL` - 4-byte fixed BQUAL length
- `XID` - 128-byte character XID

**TOTAL UOR**
The total number of all UORs in any state.

**INDOUBT**
The total number of UORs in IN_DOUBT state.

**INBACKOUT**
The total number of UORs in IN_BACKOUT state.

**INCOMMIT**
The total number of UORs in IN_COMMIT state.

**OTHER**
The total number of UORs in other states.

## Tips on Using IMS and IMS Connect Commands

This section describes tips for using IMS and IMS Connect commands to determine the status of the TCP/IP network and IMS Connect.

In the following examples the datastore definition is:
`DATASTORE=(ID=DSNAME,MEMBER=ICONNAME,TMEMBER=IMSNAME,GROUP=GRPNAME...)`

1. You can check the status of IMS Connect from IMS using the following IMS commands:
   - `/DIS OTMA`
   - `/DIS TMEMBER IMSNAME TPIPE DSNAME`

   **/DIS OTMA**
   When IMS Connect is ready for use, the output of the `/DIS OTMA` command appears as follows:

```
GROUP/MEMBER       XCF-STATUS     USER-STATUS          SECURITY
GRPNAME
-IMSNAME           ACTIVE         SERVER               FULL*
-ICONNAME          ACTIVE         ACCEPT TRAFFIC

* - CHECK, FULL, NONE or PROFILE depending on the OTMA Security
    setting (for example, enter /SEC OTMA NONE on the MVS system console
    to turn off RACF security for IMS OTMA clients). FULL is
    the default setting for OTMA security at IMS startup.
```

   **/DIS TMEMBER IMSNAME TPIPE DSNAME**
   When a message is sent to the datastore, the output appears as follows:

```
MEMBER/TPIPE     ENQCT     DEQCT    QCT    STATUS
ICONNAME
DSNAME             n         n       0
```

   **Note**: n = count

2. You can also check status using the following IMS Connect display commands:
   - `VIEWHWS`
   - `VIEWPORT`
   - `VIEWDS`
3. If you fail to receive a response to a request message sent from a client (or to check the readiness of the host datastore), you can enter the following IMS commands:
   - `/DIS A REG`
   - `/DIS TRAN TranName`

   **/DIS A REG**
   You can use this command to verify that the dependent region where your host application runs is properly configured and ready to accept messages:

```
REGID JOBNAME   TYPE  TRAN/STEP PROGRAM  STATUS   CLASS
    1    Job1   TP                       WAITING  1,  2,  3,  4
```

   **/DIS TRAN TranName**
   You can use this command to verify the class and status of a transaction and whether or not a transaction is currently queued for processing:

```
TRAN     CLS ENQCT   QCT   LCT  PLCT CP NP LP SEGSZ SEGNO PARLM    RC
TRANNAME   2     1     1 65535 65535  8  8  8     0     0  NONE     0
```

QCT is the number of transactions that are currently queued. ENQCT includes transactions that have been dequeued (processed), as well as those that are currently on the queue.

4. If, when using the IMS or IMS Connect commands (or whenever you are using IMS Connect), you receive an HWSXnnnn error message either on the MVS system console or in the response message at the client, where X is an alphabetic character and nnnn is a four-digit number, see the explanations and references cited in Part 3, “Messages and Codes,” on page 165.
5. IMS Connect requires that all active clients have unique client names. IMS Connector for Java creates a unique CLIENTID, which identifies each request that an application makes to execute an IMS transaction. If you are using TCP/IP clients other than IMS Connector for Java, you must ensure that those clients each use a unique client name. This client name value is displayed for a client in the “CLIENT=” or “ORIGIN=” fields as documented in Part 3, “Messages and Codes,” on page 165.

**Important:** There are OTMA restrictions on IMS commands. See chapter 3, ″Using IMS with OTMA″ of the *IMS Open Transaction Manager Access Guide* for more information about IMS command restrictions.

# Chapter 14. IMS Connect z/OS Commands

This section discusses the IMS Connect z/OS commands that can be used and issued through the z/OS (MVS) interface. The z/OS modify interface enables you to direct commands to IMS Connect using only the IMS *jobname*.

**In this chapter:**

## IMS Connect z/OS Command Syntax

IMS Connect supports the verb-resourcetype format. The verb-resourcetype format consists of a verb, a resource type, and zero or more keyword value pairs, with the values enclosed in parentheses.

```
                     ┌─────────────────────────────┐
                     ▼                             │
►►──verb──resourcetype──┬───────────────────────┬─┴─────────────────────►◄
                        │           ┌─,──┐      │
                        │           ▼    │      │
                        └─keyword(──value─┴─)───┘
```

**verb** A command verb representing an action. Some verb examples are QUERY, SHUTDOWN, UPDATE, and DELETE.

**resourcetype**
: The type of resource that is operated on by the verb. Some resource examples are DATASTORE, PORT, MEMBER, and UOR.

**keyword(*value*)**
: A set of zero or more keywords and values that represent attributes, filters, or other modifiers that apply to the command. For example, NAME() to identify the specific resources or SET() to set an option.

## IMS Connect z/OS Invocation

Some IMS Connect commands can be issued as a z/OS Modify command. The following syntax diagram illustrates the general syntax for entering commands through the modify interface.

```
►►──F──jobname,command──────────────────────────────────────────────►◄
```

**F** The z/OS modify command.

**jobname**
: The jobname of the address space to which the command is directed.

**command**
The command being issued.

## IMS Connect Wildcard Character Support

Some parameters on IMS Connect commands support wildcard characters for pattern matching. For such parameters, you can use the following wildcard parameters:

* Matches zero or more characters

% Matches exactly one character

The following examples illustrate some uses of wildcard characters.

| | |
|---|---|
| **CO*** | Matches any string beginning with ″CO″ of any length. For instance: CO, COO, COOP. |
| **%%S** | Matches any three-character string ending with an ″S″. For instance: IMS, CQS. |

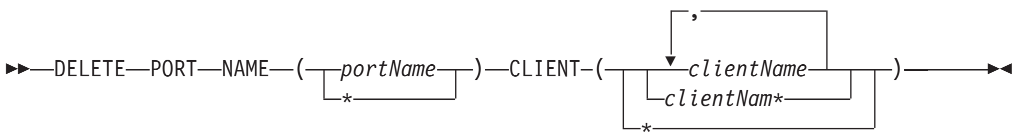
## IMS Connect DELETE Command

The IMS Connect `DELETE` command is used to delete a specific resource type.

## DELETE CLIENT

This command is used to delete a client within a specified port. `DELETE PORT NAME` performs similar functions as the STOPCLNT command.

### Format



### Usage

`DELETE PORT` is used to delete the client within the specified port name. `DELETE` or `DEL` is used to delete the client of the specified resource, which in this case is PORT. `PORT` is an IMS Connect managed resource.

**NAME ( )**
Specifies the name of the port where you want the client to be deleted. You can specify name(*) for the command to be processed for all ports.

**CLIENT ( )**
Specifies the name of the client, a wildcard, or a list of client names which you want to be deleted. You can specify client(*) for the command to be processed for all clients.

### Example

Command input:

```
F HWS01,DEL PORT NAME(9999) CLIENT(CLIENT01)
```

Command output:

```
HWSS0761I TCPIP COMMUNICATION WITH CLIENT=9999_CLIENT01 STOPPED; M=SCCM
```

Explanation: The client, `CLIENT01`, within port number 9999 is deleted.

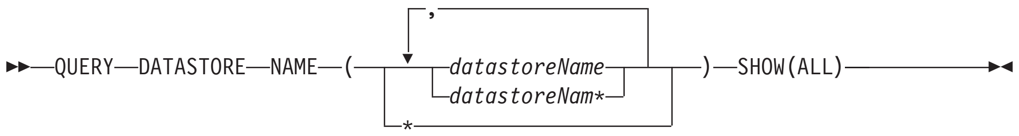
## IMS Connect QUERY Command

The IMS Connect `QUERY` command is used to display the current status of a specified resource type.

## QUERY DATASTORE

The datastore resource type refers to the destination that is accessing IMS OTMA. With the datastore resource type, IMS Connect can communicate with IMS through IMS OTMA. If a datastore name is used, this name must match the ID parameter of a DATASTORE configuration statement of the HWSCFSxx configuration file and only the information for this datastore is displayed or updated. `QUERY DATASTORE` performs similar functions as the VIEWDS command.

### Format



### Usage

`QUERY DATASTORE` is used to display the datastore status. `QUERY` or `QRY` is used to query the status or attributes of a specified resource. For example, the specified resource is `DATASTORE` which is an IMS Connect managed resource.

**NAME (datastoreName)**
Specifies the datastore name to be displayed. You can specify a single datastore name, a wildcard name, or a list of datastore names separated by commas. You can specify name(*) for the command to be processed for all datastores.

**SHOW ( )**
Specifies the output fields to be returned.

| | |
|---|---|
| **ALL** | Returns all output fields. This is the default. |

### Example

Command input:

```
F HWS1, QRY DATASTORE NAME(SOCKEYE) SHOW(ALL)
```

Command output:

```
DATASTORE=SOCKEYE     STATUS=ACTIVE
   GROUP=XCFGRP1     MEMBER=HWS1
   TARGET MEMBER=SOCKEYE
   RACF APPL NAME=APPLID1
```

Explanation: The status of the IMS Connect datastore name SOCKEYE is displayed.

## QUERY MEMBER

This command is used to display the status of IMS Connect. `QUERY MEMBER` performs similar functions as the VIEWHWS command.

### Format

```
►►──QUERY──MEMBER──TYPE─(IMSCON)──SHOW─(ALL)──────────────────────────────────────►◄
```

### Usage

QUERY or QRY is used to query the status or attributes of a specified resource and MEMBER is an IMS Connect managed specified resource. The command, QUERY MEMBER, is used to display the status of IMS Connect.

**TYPE ( )**
Specifies the target type for action.

**IMSCON**
Specifies that IMS Connect is the target type.

**SHOW ( )**
Specifies the target type to be returned.

**ALL** Returns all output fields. This is the default.

### Example

Query the status of IMS Connect.

Command input:

```
F HWS1,QRY MEMBER TYPE(IMSCON)
```

Command output:

```
HWSC0001I   HWS ID=HWS1     RACF=N
HWSC0001I      MAXSOC=50  TIMEOUT=5000
HWSC0001I    RRS=N      STATUS=REGISTERED
  VERSION=210 IP-ADDRESS=009.030.124.032
DATASTORE=IMS1     STATUS=ACTIVE
GROUP=XCFGRP1  MEMBER=HWS1
  TARGET MEMBER=IMS1
  RACF APPL NAME=APPLID1
DATASTORE=IMSA     STATUS=DISCONNECT
  GROUP=XCFGRP1  MEMBER=HWSA
  TARGET MEMBER=IMSA
  RACF APPL NAME=APPLID2
IMSPLEX=PLEX1    STATUS=NOT ACTIVE
  MEMBER=IMSPLEX1 TARGET=PLEX1
PORT=9999     STATUS=ACTIVE
  NO ACTIVE CLIENTS
PORT=LOCAL    STATUS=ACTIVE
  NO ACTIVE CLIENTS
PORT=9998S    STATUS=NOT ACTIVE
PORT=8888S    STATUS=NOT ACTIVE
```

Explanation: The status of IMS Connection is displayed. The status of each datastore and port number is listed.

## QUERY PORT

This command is used to display the current status of a requested port. The PORT resource type refers to the port number that binds a socket with TCP/IP. If a port name is specified, the name must match a port name already defined in the PORT substatement of the TCP/IP configuration statement in the IMS Connect configuration member. QUERY PORT performs similar functions as the VIEWPORT command.

### Format

```
                                 ┌─,──────────────┐
                                 ▼                │
►►──QUERY──PORT──NAME──(──┬──┬─portName──┬──┴──┬──)SHOW(ALL)──────────────────►◄
                          │  └─portNam*──┘     │
                          └─*──────────────────┘
```

### Usage

QUERY or QRY is used to query the status or attributes of a specified resource and PORT is an IMS Connect managed specified resource. The command, QUERY PORT, is used to display the requested port status.

**NAME (datastoreName)**
: Specifies the port name to be displayed. You can specify a single port name, a wildcard name, or a list of port names separated by commas. You can specify name(*) for the command to be processed for all ports.

**SHOW ( )**
: Specifies the output fields to be returned.

| | |
|---|---|
| **ALL** | Returns all output fields. This is the default. |

### Example

Command input:

```
F HWS1,QUERY PORT NAME(9999) SHOW(ALL)
```

Command output:

```
HWSC0001I     PORT=9999     STATUS=ACTIVE
HWSC0001I     NO ACTIVE CLIENTS
```

Explanation: The status of the IMS Connect port number 9999 is displayed and shows no active clients.

## QUERY UOR

This command is used to display the current status of the request unit of recovery (UOR) identifier. QUERY UOR performs similar function as the VIEWUOR command.

### Format

```
                               ┌─,──────────┐
                               ▼            │
►►──QUERY──UOR──NAME──(──┬──┬─urid──┬──┴──┬──)SHOW(ALL)──────────────────►◄
                         │  └─urid*─┘     │
                         └─*──────────────┘
```

### Usage

QUERY or QRY is used to query the status or attributes of the specified resource. UOR is an IMS Connect managed resource. QUERY MEMBER is used to display the status of IMS Connect. This command can be issued through the Operations Manager API or through the z/OS modify interface.

**NAME ( )**
: Specifies the name of the urid to be displayed. You can specify a single urid, a wildcard name, or a list of urids separated by commas. You can specify name(*) for the command to be processed for all uors.

*urid*
: Specifies the urid to be displayed.

**SHOW ( )**
: Specifies the target type to be returned.

| | |
|---|---|
| **ALL** | Returns all output fields. This is the default. |

### Example

Query the status of IMS Connect urids.

Command input:

```
F HWS1,QRY UOR NAME(*)
```

Command output:

```
HWSC0050I NO ACTIVE UOR
```

## IMS Connect SHUTDOWN Command

The IMS Connect `SHUTDOWN` command is used to shut down a specified resource type.

## SHUTDOWN MEMBER

This command is used to shut down IMS Connect. `SHUTDOWN MEMBER` performs similar functions as the CLOSEHWS command.

### Format

```
►►──SHUTDOWN──MEMBER──OPTION(──┬─QUIESCE─┬─)──────────────────────────────────►◄
                               └─(FORCE)─┘
```

### Usage

`SHUTDOWN` or `SHUT` is used to shutdown a specified resource. `MEMBER` is an IMS Connect specified managed resource. The command, `SHUTDOWN MEMBER`, is used to shutdown IMS Connect.

**OPTION ( )**
: Specifies the attributes to be stopped.

**QUIESCE**
: Specifies that termination is to end all client and datastore connections in a controlled manner.

**FORCE ( )**
: Specifies that termination is to end all client and datastore connections immediately. Immediate termination forces any IMS application that is running with connected clients to abnormally terminate.

### Example

Shutdown HWS with force option.

Command input:

```
F HWS07,SHUTDOWN HWS OPTION(FORCE)
```

Command output:

```
HWS07 PURGED
```

Explanation: The HWS member is shut down.

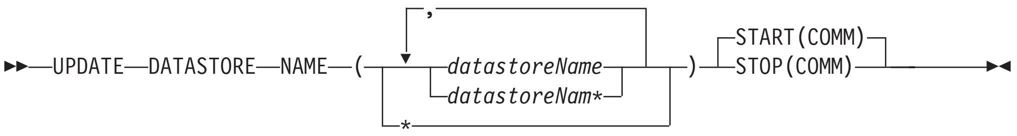
## IMS Connect UPDATE Command

The IMS Connect `UPDATE` command is used to update the current status of a specified resource type.

## UPDATE DATASTORE

This command is used to update the requested datastore. `UPDATE DATASTORE` performs similar functions as the OPENDS and STOPDS commands.

### Format



### Usage

`UPDATE` or `UPD` is used to update the status or attributes of a specified resource. `DATASTORE` is an IMS Connect managed specified resource. The command `UPDATE DATASTORE` is used to update the current status of the requested datastore.

**NAME ( )**
: Specifies the name of the datastore to be updated. You can specify a single datastore, a wildcard name, or a list of datastores separated by commas. You can specify name(*) for the command to be processed for all datastores.

*datastoreName*
: Specifies the datastore to be updated.

**START ( )**
: Specifies the attributes to be started.

| | |
|---|---|
| **COMM** | Starts the communication with the datastore. |

**STOP ( )**
: Specifies the attributes to be stopped.

| | |
|---|---|
| **COMM** | Stops the communication with the datastore. |

### Example

Command input:

```
F HWS1,UPD DATASTORE NAME(SOCKEYE) STOP(COMM)
```

Command output:

```
HWSD028I COMMUNICATION WITH DS=SOCKEYE    STOPPED;
M=DSCM
```

Explanation: The communication with the datastore, SOCKEYE, is stopped.

## UPDATE MEMBER

This command is used to update the attributes of IMS Connect.

### Format

```
►►──UPDATE──MEMBER──TYPE(IMSCON)──┬─STOP(TRACE)──────────┬──────────────────────────►◄
                                  ├─START(TRACE)─────────┤
                                  └─SET(RACF(─┬─ON──┬─))─┘
                                              └─OFF─┘
```

### Usage

`UPDATE` or `UPD` is used to update the status or attributes of a specified resource. `MEMBER` is an IMS Connect managed resource. The command, `UPDATE MEMBER`, is used to update the status of IMS Connect.

**TYPE ( )**
Specifies the target type for action.

**IMSCON**
Specifies IMS Connect as the target type.

**START ( )**
Specifies the attributes to be started.

**TRACE** Specifies that the line trace will be started.

**STOP ( )**
Specifies the attributes to be stopped.

**TRACE** Specifies that the line trace will be stopped.

**SET**
Specifies the attribute values to be changed.

**RACF ( )**
Specifies the attributes to be set.

**ON** Enables the RACF user identification and verification.

**OFF** Disables the RACF user identification and verification.

### Example

Command input:

```
F HWS1,UPD MEMBER TYPE(IMSCON) SET(RACF(OFF))
```

Command output: None

Command input:

```
F HWS1,QUERY MEMBER TYPE(IMSCON) SHOW(ALL)
```

Command output:

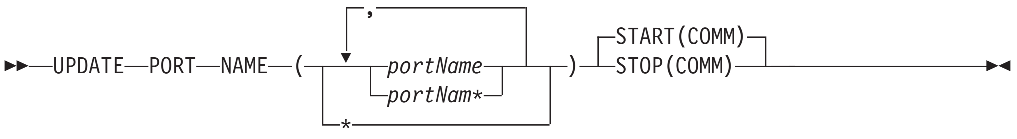
```
HWSC0001I   HWS  ID=HWS1     RACF=N
```

Explanation: RACF security check is turned off.

## UPDATE PORT

This command is used to update the port that is used by IMS Connect. `UPDATE PORT` performs similar functions as the OPENPORT and STOPPORT command.

## Format



## Usage

UPDATE or UPD is used to update the status or attributes of a specified resource. PORT is an IMS Connect managed resource. The command, UPDATE PORT, is used to update the status of a requested port.

**NAME ( )**
Specifies the port name to be updated. You can specify a single port, a wildcard name, or a list of port names separated by commas. You can specify name(*) for the command to be processed for all ports.

*portName*
Specifies the port to be displayed.

**START ( )**
Specifies the attributes to be started.

| | |
|---|---|
| **COMM** | Starts the communication with the TCPIP port. |

**STOP ( )**
Specifies the attributes to be stopped.

| | |
|---|---|
| **COMM** | Stops the communication with the TCPIP port. |

## Example

Command input:

```
F HWS1,UPD PORT NAME(9999) STOP(COMM)
```

Command output:

```
HWSS0770I LISTENING ON PORT=9999    TERMINATED;
M=SSCH
```

Explanation: The port has been updated.

# Chapter 15. User Message Exits for IMS Connect

Three user message exits are provided with IMS Connect. Each of the message exits allows you to call IMSLSECX, the security message exit, issue the RACF function in these user message exit routines, or use the IMS Connect user RACF function.

This chapter contains Product-Sensitive Programming Interface and Associated Guidance Information.

**In this chapter:**

## HWSIMSO0 and HWSIMSO1 User Message Exits

IMS Connect Version 9 is the final and last release with which HWSIMSO0 and HWSIMSO1 is shipped. It is recommended that you move to HWSSMPL1. You may also move to HWSSMPL0. There are two ways of migrating to HWSSMPL0 or HWSSMPL1.

1. Change the client code to support HWSSMPL0 or HWSSMPL1 by changing the ASCII or EBCDIC settings in your code to one of the following options:
   - Change *IRMREQ* to *SAMPLE* to move from HWSIMSO0 to HWSSMPL0.
   - Change *IRMRE1* to *SAMPL1* to move from HWSIMSO1 to HWSSMPL1.
   - Change *IRMREQ* to *SAMPL1* to move from HWSIMSO0 to HWSSMPL1. You must also modify the client code to accept the fullword output message length field preceding the LLZZ data.
2. Change either HWSSMPL1, HWSSMPL0, or both to accept the current settings as they exist in the client code by changing the ASCII or EBCDIC setting in the user message exit. You can select one of the following options:
   - Change *SAMPLE* to *IRMREQ* to move from HWSIMSO0 to HWSSMPL0.
   - Change *SAMPL1* to *IRMRE1* to move from HWSIMSO1 to HWSSMPL1.
   - Change *SAMPLE* to *IRMRE1* to move from HWSIMSO0 to HWSSMPL1. You must also modify the client code to accept the fullword output message length field preceding the LLZZ data.

HWSIMSO0 and HWSIMSO1 are shipped as object code only (OCO) with IMS Connect. The exits are shipped as OCO to allow IMS Connect and the user message exit provided by TCP/IP to stay in sync without requiring a simultaneous upgrade of both products when message exit functions change. If your installation uses either HWSIMSO0 or HWSIMSO1 as its default, you must concatenate the RESLIB that contains the IMS Connect-supplied message exit in front of the TCP/IP RESLIB to ensure that the correct message exit is used.

**Recommendation**: If HWSIMSO0 or HWSIMSO1 is inadequate for your installation, modify and use HWSSMPL0 or HWSSMPL1. See Part 1, “User’s Guide and Reference,” on page 1, for information about customizing user message exits.

HWSIMSO0 and HWSIMSO1 provide the following functions:

- Perform data translation of ASCII to EBCDIC for input messages.
- Perform data translation of EBCDIC to ASCII for output messages.
- Build the IMS Connect message structure (BPE and OTMA headers).
- If IMSLSECX (the security message exit) is link-edited with either of these message exits, then the security message exit is called.
- Default to COMMIT MODE=1.
- Default to SYNC LEVEL=NONE.
- Set up RACF options.
- Analyze the following message header options:
  - COMMIT MODE to override default.
  - SYNC LEVEL to override default.
  - MFS MOD name.
  - ACK/NAK/DEALLOCATE.
  - RACF options.
  - If no Client ID is passed to the exit, then the message exits generate the Client ID.

**Restriction**: Do not use the TCP/IP supplied message exit (EZAIMSO0) for IMS Connect. This message exit does not support IMS Connect.

If any errors occur with the TCP/IP translate table, report those problems to TCP/IP and notify them of the TCP/IP release you are currently using. When corrections have been made to the translate table, link-edit this exit again to pick up the corrected translate tables. If any errors occur with the IMS Connect user message exit (HWSIMSO0 and HWSIMSO1) code, report those problems to IMS Connect for corrections.

## HWSSMPL0 and HWSSMPL1 User Message Exits

HWSSMPL0, HWSSMPL1, and the related MACROS are shipped as source code. This allows you to modify the message exit for your installation's requirements.

See Part 1, "User's Guide and Reference," on page 1 for information about customizing this user message exit.

The HWSSMPL0 and HWSSMPL1 user message exits provides the following functions:

- Perform data translation of ASCII to EBCDIC for input messages.
- Perform data translation of EBCDIC to ASCII for output messages.
- Build the IMS Connect message structure (BPE and OTMA headers).
- If IMSLSECX (the security message exit) is link-edited with either of these message exits, then the security message exit is called.
- Default to COMMIT MODE=1.
- Default to SYNC LEVEL=NONE.
- Set up RACF options.
- Analyze the following message header options:
  - COMMIT MODE to override default.
  - SYNC LEVEL to override default.
  - MFS MOD name.
  - ACK/NAK/DEALLOCATE.

  - RACF options.
  - If no Client ID is passed to the exit, the message exit generates the Client ID.

## HWSJAVA0 User Message Exit

HWSJAVA0 and the related macros are shipped as source code. The reason this user message exit is shipped as source code is to allow you the ability to modify the message exit for installation uniqueness. HWSJAVA0 gives you the flexibility to exit your messages and do your own security checking.

See Part 1, “User’s Guide and Reference,” on page 1, for information about customizing this user message exit.

## HWSCSLO0 and HWSCSLO1 User Message Exits for Control Center

HWSCSLO0 and HWSCSLO1 are delivered as object code only (OCO) with IMS Connect. The exit is formatted OCO to allow IMS Connect and the user message exit for the Control Center to be synchronized without requiring simultaneous upgrades of other products when message exit functions change. If your installation activates the Control Center to communicate with OM, you must include the HWSCSLO0 and HWSCSLO1 exit names in the EXIT= parameter of the TCPIP statement.

HWSCSLO0 provides the following functions required by the Control Center:

- Performs data translation of ASCII to EBCDIC for input messages.
- Performs data translation of EBCDIC to ASCII for output messages.
- Builds the IMS Connect message structure (BPE and OM headers required by IMS Connect) for input messages.
- Removes the IMS Connect internal OM headers for output messages.
- Defaults to COMMIT MODE=1.
- Defaults to SYNCH LEVEL=NONE.
- Analyzes the following message header options:
  - COMMIT MODE override of the default
  - SYNC LEVEL override of the default
  - If no client ID is passed to the exit, then the message exit generates the client ID

HWSCSLO1 provides the following functions required by the Control Center:

- Performs no translation output messages.
- Builds the IMS Connect message structure (BPE and OM headers required by IMS Connect) for input messages.
- Removes the IMS Connect internal OM headers for output messages.
- Defaults to COMMIT MODE=1.
- Defaults to SYNCH LEVEL=NONE.
- Analyzes the following message header options:
  - COMMIT MODE override of the default
  - SYNC LEVEL override of the default
  - If no client ID is passed to the exit, then the message exit generates the client ID.

**Note:** You do not need to specify the HWSCSLO0 and HWSCSLO1 exit names in the TCPIP statement EXIT= parameter if the Control Center is not used.

# Part 3. Messages and Codes

# Chapter 16. IMS Connect Error Codes and Messages

This chapter describes and explains all of the IMS Connect error codes and messages, and suggests corrective action.

Some messages include an **M**=*mc* value. The *mc* represents the IMS Connect module suffix that issued the message. For example, in **M=CPAR**, the issuing module is HWSCPAR0.

Some messages include an **E** =*ec* value. The *ec* represents the TCP/IP error code. Refer to your TCP/IP Messages and Codes documentation for more information on the *ec* value.

This chapter contains Diagnosis, Modification, or Tuning Information.

**HWSC0000I nn HWSC0000I *IMS CONNECT READY* *hid***

**Explanation:** An MVS outstanding reply message used for entering IMS Connect commands.

*hid* identifies the HWS (the `ID` parameter of the `HWS` statement in the `HWSCFGxx` configuration files).

**HWSC0001I HWSC0001I HWS ID=*hid***

**Explanation:** *hid* identifies the HWS (from the ID parameter of the HWS statement in the IMS Connect configuration member).

**HWSC0001I HWSC0001I Maxsoc=*mso* Timeout=*time***

**Explanation:** *mso* identifies maximum number of sockets per port that IMS Connect can open (from the MAXSOC parameter of the TCPIP statement in the IMS Connect configuration member).

*time* identifies the time, in hundredths of seconds, to be used as a timeout interval in waiting for a response (from the TIMEOUT parameter of the TCPIP statement in the IMS Connect configuration member).

**HWSC0001I HWSC0001I DS ID=*did* Status=*state***

**Explanation:** *did* identifies the datastore (from the ID parameter of the DATASTORE statement in the IMS Connect configuration member).

*state* identifies the state of the datastore:
- ACTIVE
- NOT ACTIVE
- DISCONNECT

**HWSC0001I HWSC0001I IMSPLEX=*ipid* Status=*state***

**Explanation:** *ipid* identifies the IMSplex (from the ID parameter of the IMSplex statement in the IMS Connect configuration member).

*state* identifies the state of the IMSplex
- ACTIVE
- NOT ACTIVE
- DISCONNECT

**HWSC0001I HWSC0001I Group=*group* Member=*member***

**Explanation:** *group* identifies the XCF group name (from the GROUP parameter of the DATASTORE statement in the IMS Connect configuration member.

*member* identifies the XCF member name (from the MEMBER parameter of the DATASTORE statement in the IMS Connect configuration member).

**HWSC0001I HWSC0001I Target Member=*tmember***

**Explanation:** *tmember* identifies the XCF target member name (from the TMEMBER parameter of the DATASTORE statement in the IMS Connect configuration member).

**HWSC0001I HWSC0001I RACF APPL NAME=*name***

**Explanation:** *name* identifies the RACF PTKTDATA class that contains the PassTicket Key.

**HWSC0001I HWSC0001I Member=*member* Target=*tmember***

**Explanation:** *member* identifies the group name (from the GROUP parameter of the IMSplex statement in the IMS Connect configuration member).

*tmember* identifies the target member name (from the TMEMBER parameter of the IMSplex statement in the IMS Connect configuration member).

**HWSC0001I HWSC0001I Port=*port* Status=*state***

**Explanation:** *port* identifies the appropriate port, either as a TCP/IP port number or LOCAL (from the PORTID parameter of the TCPIP statement in the IMS Connect configuration member).

*state* identifies the state of the port:
- ACTIVE
- NOT ACTIVE
- NOT DEFINED

**HWSC0001I HWSC0001I cli usid tran stat time [ipaddress clientport]**

**Explanation:**
**cli** identifies the IMS Connect client ID.
**usid** identifies the userid value of the client.
**tran** identifies the transaction code.
**stat** identifies the state of the datastore or port:
WFCM - IMS Connect is waiting for the client to confirm the last response.
CONF - the client is in a conversation.
**time** indicates the number of seconds since anything was received from the client.
**ipaddress** identifies the TCP/IP address of the client.
**clientport** identifies the TCP/IP port for the client.

**HWSC0001I HWSC0001I Client=*clientname* Status=*c_state***

**Explanation:** *clientname* is the TCP/IP client; IMS Connector for Java client names are dynamically generated. If the client name is DELDUMMY, the connection process has completed; however, no data has been received.

*c_state* is the state of the connection from IMS Connect:
ACTV - the client is in the connection process.
CERR - the client is in the communications error state.
INTE - the client is in the interface error state.
RECV - IMS Connect is in receive state, waiting for data from the client.
SHUT - the client is in the shutdown state.
STOP - the client is in the process of being stopped by a STOPCLNT command.
TERM - the client is in the termination state (shutting down).
XMIT - the client has not issued a receive; IMS Connect is holding the output message.
CONN - the client is in a connected state, waiting for output from IMS.

**HWSC0001I HWSC0001I NO ACTIVE DATASTORE**

**HWSC0001I HWSC0001I NO ACTIVE PORTS**

**HWSC0001I HWSC0001I NO ACTIVE CLIENTS**

**HWSC0001I HWSC0001I TOTAL CLIENTS=*xx* RECV=*xx* CONN=*xx* XMIT=*xx* OTHER=*xxx***

**Explanation:** Combinations of these messages are issued when any of the VIEW commands (VIEWDS, VIEWHWS, VIEWPORT) are executed.

**System Action:** The messages are issued, and IMS Connect continues to run.

**HWSC0010I HELLO, WELCOME TO IMS CONNECT!**

**Explanation:** Indicates that IMS Connect is ready.

**HWSC0020I IMS CONNECT IN TERMINATION**

**Explanation:** Indicates that IMS Connect has shut down.

**HWSC0100W UNABLE TO ALLOCATE STORAGE FOR COMMAND; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage for the command buffer cannot be allocated.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 49 for service and return codes.

*Table 49. Service and Return Code Explanations*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETC01L | BPECBGET, the system service used to acquire the C01K. | 4 | An incorrect CBTE address is passed to the CG get routine. This is an internal system error. |
| 8 | Storage is unavailable to satisfy the request. | | |

*Table 49. Service and Return Code Explanations (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETFWEB | BPECBGET, the system service used to acquire the FWEB. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:** CMDC- HWSCMOP0

**System Action:** This message is issued and, if possible, the requestor of the command is notified. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSC0101E FUNCTION WORK ELEMENT PROCESSING FAILURE, FUNC=*func*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function work element (FWE) cannot be processed. The FWE requests work between components and within components. This structure contains the function and parameters that a service requires for processing.

In the message text:

- *func* identifies the function requested.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 50 for service and return code explanation.

*Table 50. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVFUNC | The function requested in the FWE is incorrect. | 4 | This is a processing error. |

**Module:**

- CMDC - HWSCMDC0
- CMOP - HWSCMOP0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSC0110W COMMAND VERB BLOCK PROCESS FAILURE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage for the command verb block (CVB) cannot be allocated. The CVB contains the command verb and its parameters and is the structure used by all command processors to process a command in IMS Connect. Without this block, a command cannot be processed.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 51 for service and return codes.

*Table 51. Service and Return Code Explanations*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETCVBB | BPECBGET, the system service used to acquire the CVB. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:** CMDC - HWSCMDC0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSC0112E COMMAND PARSER FAILED, COMMAND=*hwscmd*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An error occurs during an attempt to parse the command from the command buffer.

In the message text:

- *hwscmd* identifies the command.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 52 for service and return code information.

*Table 52. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| NODATA | No data exists in the command buffer. | 40 | This is a processing error. |
| INVCMD | The command verb cmd is not a valid HWS command. | 41 | This is a processing error. |
| NOPARM | The command requires parameters, but you did not supply any. | 42 | This is a processing error. |
| NO2PARM | The second parameter is missing for this command. | 43 | This is a processing error. |
| PARM1ERR | The first parameter is wrong. | 44 | Please use the correct syntax. |

**Module:** CPAR - HWSCPAR0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the command buffer is freed and IMS Connect continues to run.

**System Programmer Response:** Ensure that the correct command is entered. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSC0114W COMMAND=*hwscmd*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** During an attempt to propagate the command to the next level of command processing, an error is detected. The command is being forwarded to the component that can process it; however, a resource that this command is targeting might not be available.

In the message text:

- *hwscmd* identifies the command.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 53 for an explanation of service and return codes.

*Table 53. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVLCMD | The command is incorrect. | 4 | This is a processing error. |
| NFNDDCT | The datastore communication table cannot be found. This table contains the information that is retrieved from the configuration member HWSCFGxx for each datastore defined. | 8 | This is a processing error. |
| NFNDDST | The datastore table cannot be found. This table maintains the activity of a datastore. | 4 | This is a processing error. |
| NFNDSVT | The server table cannot be found. This table maintains the activity of a connected client. | 4 | This is a processing error. |
| NACTO/C | The open/close thread is not active. The command can only be processed by the open/close controller and the controller is no longer active. IMS Connect could be shutting down. | 4 | This is a processing error. |
| BPEGETM | System service used to acquire the response buffer. | 4 | An incorrect or unsupported subpool is specified. |
| | | 8 | A zero length is requested. |
| | | 12 | Unable to obtain the requested storage (MVS GETMAIN failed). |
| NFNDCOMP | The component that handles the requested function cannot be found. An HWS component issues an interface call for another component's service and the component being requested for service cannot be located. | 4 | This is a processing error. |
| NFNDFUNC | The requested function cannot be found. An HWS component issues an interface call for another component's service and the service being requested cannot be located. | 8 | This is a processing error. |
| PORTNACT STOPPORT | Command for inactive or already stopped PORT. | 4 | PORT not active. |

**Module:** CVBC - HWSCVBC0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the command buffer is freed and the IMS Connect continues to run.

**System Programmer Response:** Ensure that the correct command is entered. If the service code is NFNDCOMP or NFNDFUNC, this is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSC0120W UNABLE TO SEND COMMAND RESPONSE TO HWSHOST; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An error occurs during an attempt to send the command response back to the system console.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 54 for an explanation of service and return codes.

*Table 54. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVLTAG | The command response tag is incorrect. Command response tags represent the types of response that are being sent. | 4 | This is a processing error. |
| NFNDCOMP | The component that handles the requested function cannot be found. An HWS component issues a call to the call interface for another component's service and the requested component cannot be located. | 4 | This is a processing error. |
| NFNDFUNC | The requested function cannot be found. An HWS component issues a call to the call interface for another component's service and the requested service cannot be located. | 8 | This is a processing error. |

**Module:** CRSP - HWSCRSP0

**System Action:** This message is issued and the command response buffers are freed. IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSC0130I CLOSEHWS ALREADY IN PROGRESS; M=*mc***

**Explanation:** IMS Connect is in the process of closing. This message is issued when a `CLOSEHWS` command is entered more than once.

In the message text:

- *mc* identifies the module issuing the message.

**Module:** CHWS - HWSCHWS0

**System Action:** If IMS Connect does not terminate after the `CLOSEHWS`command is entered, use the `VIEWHWS` command to determine the status and queues for the datastores and clients. Ensure that no clients are active. If any clients are active, IMS Connect does not terminate. You can issue the IMS Connect command `CLOSEHWS FORCE` to force IMS Connect to terminate.

**HWSD0200E FUNCTION WORK ELEMENT PROCESSING FAILURE, FUNC=*func*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function work element (FWE) cannot be processed. The FWE requests work between and within components. This structure contains the function and parameters that a service requires for processing.

In the message text:

- *func* identifies the function requested.

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 55 for an explanation of service and return codes.

*Table 55. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVFUNC | The function requested in the FWE is incorrect. | 4 | This is a processing error. |

**Module:**
- DOCC — HWSDOCC0
- DSCH — HWSDSCH0
- DCVC — HWSDCVC0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSD0202W FWE FUNCTION=*func* FAILED FOR DS=*did*, COMMAND=*hwscmd* IN PROGRESS; M=*mc***

**Explanation:** The function *func* cannot be processed because the command identified by hwscmd is already being processed.

In the message text:
- *func* identifies the function requested.
- *did* identifies the datastore.
- *hwscmd* identifies the IMS Connect command in progress.
- *mc* identifies the module issuing the message.

**Module:** DSCM— HWSDSCM0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** The IMS Connect command in progress is terminating the datastore; therefore, any new function for that datastore cannot be processed.

**HWSD0204W COMMAND=*hwscmd* FAILED FOR DS=*did*, COMMAND=*prev_hwscmd* ALREADY IN PROGRESS; M=*mc***

**Explanation:** The IMS Connect command entered for the datastore, *hwscmd*, cannot be processed because a command for that datastore, *prev_hwscmd*, is already in progress.

In the message text:
- *hwscmd* identifies the IMS Connect command that was blocked from being run by *bhwscmd*.
- *did* identifies the datastore affected by *hwscmd* and *prev_hwscmd*.
- *prev_hwscmd* identifies the IMS Connect command that is blocking *hwscmd* from running.
- *mc* identifies the module issuing the message.

**Module:** DSCM— HWSDSCM0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** The IMS Connect command in progress is terminating the datastore; therefore, any new commands cannot be processed. If the IMS Connect command (*hwscmd*) was `CLOSEHWS`, the IMS Connect terminates after the processing of *prev_hwscmd* completes.

**HWSD0212E UNABLE TO START SCHEDULER CONTROLLER; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the scheduler controller structure, or the scheduler controller thread cannot be scheduled. A scheduler controller is started for each datastore that is defined to IMS Connect. The scheduler controller is the controller that schedules the threads associated with a datastore.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 56 on page 174 for an explanation of service and return codes.

*Table 56. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETDSTB | BPECBGET, the system service used to acquire the datastore table (DST). | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU) for the scheduler controller. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| INCLOSE | IMS Connect is in the process of closing. No datastore can be started. | 12 | This is a processing error. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the scheduler controller thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread failed. |

**Module:**

- DOCC — HWSDOCC0
- DOCM — HWSDOCM0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the

IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

---

**HWSD0222E UNABLE TO START TRANSMIT/RECEIVE THREADS FOR DS=*did*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the transmit or receive thread structure, or either the transmit thread or the receive thread cannot be scheduled. A transmit thread and receive thread are allocated for each datastore that is defined for message transmission and reception.

In the message text:

- *did* identifies the datastore.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 57 for an explanation of service and return codes.

*Table 57. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETDSBB | BPECBGET, the system service used to acquire the datastore block (DSB) for the transmit and receive threads. This is the execution block for a thread. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETC01K | BPECBGET, the system service used to acquire the common 1024 byte (C01K) for the conversation controller. The area is used as a work area. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU) for the transmit and receive threads. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the scheduler controller thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |

*Table 57. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread failed. |

**Module:**

- DSC1 — HWSDSC10
- DSCM — HWSDSCM0

**System Action:** This message is issued and IMS Connect continues to run with datastores that can be started.

**System Programmer Response:** On the subsequent close and startup of IMS Connect, ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSD0227W CLOSE FAILED FOR DS=*did*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to close the named datastore is unsuccessful during IMS Connect shutdown.

In the message text:

- *did* identifies the datastore.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 58 for an explanation of service and return codes.

*Table 58. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETFWEB | BPECBGET, the system service used to acquire an FWE to notify all datastore to close. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:** DOC3— HWSDOC30

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Storage cannot be allocated to notify the datastore to close. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSD0230I *type*=*id* ALREADY ACTIVE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An `OPENDS` or `OPENIP` command is issued for a datastore or IMSplex that is already active.

In the message text:

- *type* identifies the datastore (DS) or IMSplex (IP).
- *id* identifies the datastore or IMSplex name.
- *rc* identifies the return code.

- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 59 for an explanation of service and return codes.

Table 59. Service and Return Code Explanation

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| ACTIVDST | The datastore is active. | 0 | The process is successful. |
| ACTIVEIP | The IMSplex is active. | 0 | The process is successful. |
| ACTVDISC | The IMSplex is active, however, it is currently disconnected. | 0 | The IMSplex is disconnected and will remain disconnected until the SCI is started. When the SCI is started, IMS Connect will automatically reconnect to the SCI. |

**Module:** DOCM— HWSDOCM0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Ensure that the correct name is provided in the `OPENDS` or `OPENIP` command. If you are issuing the `OPENIP` command, determine if SCI has been initialized. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSD0250W UNABLE TO NOTIFY MSG ORIGIN=*clientid* OF OTMA COMMUNICATION ERROR; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to notify the TCP/IP client who originated a message, which is either being processed or queued for processing, that a communication error with IMS OTMA has occurred.

In the message text:

- *clientid* identifies the TCP/IP client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 60 for an explanation of service and return codes.

Table 60. Service and Return Code Explanation

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| COMMERR | Communication error with IMS OTMA. | 4 | This is a processing error. |

**Module:**

- DXMT— HWSDXMT0
- DSC3— HWSDSC30
- DSCE— HWSDSCE0

**System Action:** This message is issued and IMS Connect continues to run. The message whose processing caused the error is discarded.

**System Programmer Response:** This error can occur when the datastore is no longer active or the communication linkage to IMS Connect is broken.

**HWSD0252W UNABLE TO SEND RESPONSE RECEIVED FROM DS=*did* to CLIENT=*clientid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to send the response received from a datastore to the required TCP/IP client.

In the message text:

- *did* identifies the datastore.
- *clientid* identifies the TCP/IP client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 61 on page 178 for an explanation of return and service codes.

*Table 61. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVLDTOK | Invalid server token has been detected. | 4 | Use the correct server token for the conversation iteration. Or, a second client is starting a conversation and is using a duplicate ID while the first client is in a conversation. |
| LATEMSG | Message from IMS received after the socket was closed and cannot be delivered to the client at this time. The socket was closed before the IMS output was received by IMS Connect or TCP/IP had been terminated. | 4 | Message from IMS was received by IMS Connect and was not delivered to the client. |
| NFNDCOMP | The component that handles the requested function cannot be found. An IMS Connect component issues an interface call for another component's service and the requested component cannot be located. | 4 | This is a processing error. |
| NFNDFUNC | The requested function cannot be found. An IMS Connect component issues an interface call for another component's service and the requested service cannot be located. | 8 | This is a processing error. |
| NFNDSVT | The server table cannot be found. This table maintains the activity of a connected IMS Connect client. | 4 | This is a processing error or a timeout has occurred. |

**Module:** DREC— HWSDREC0

**System Action:** This message is issued and IMS Connect continues to run. The response message is discarded.

**System Programmer Response:** This error can occur when the client is no longer active and is not connected to IMS Connect. If the service code is NFNDCOMP or NFNDFUNC, this is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSD0254W UNABLE TO NOTIFY DS=*did* SCHEDULER OF COMMUNICATION ERROR; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to notify the scheduler controller for the named datastore that a communication error has occurred. When this condition occurs, IMS Connect views the named datastore as active. However, messages queued for the datastore are not sent to it.

In the message text:

- *did* identifies the datastore.

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 62 for an explanation of service and return codes.

*Table 62. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| COMMERR | Communication error. | 4 | This is a processing error. |

**Module:**

- DREC— HWSDREC0
- DXMT—HWSDXMT0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Issue the `STOPDS` command to terminate the datastore. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

---

**HWSD0260I DS=*did tname* THREAD TERMINATED; M=*mc***

**Explanation:** The datastore transmit thread or receive thread has terminated.

In the message text:

- *did* identifies the datastore.
- *tname* identifies the thread type.
- *mc* identifies the module issuing the message.

**Module:**

- DREC— HWSDREC0
- DXMT—HWSDXMT0

**System Action:** This message is issued when a datastore thread has terminated.

---

**HWSD0270I OTMA OPEN FAILED; R=*rc*, M=*mc***

**Explanation:** Communication with a datastore failed during IMS Connect startup or in response to an IMS Connect `OPENDS` command and resulted in the failure of the OTMA open function.

In the message text:

- *rc* identifies the return code.
- *mc* identifies the module issuing the message.

**Module:** DOC1— HWSDOC10

**System Action:** This message is issued when communication to OTMA fails due to a communications failure with a datastore. See message HWSO1105W on page 191 or message HWSO1110W on page 191 for additional information related to this failure.

**System Programmer Response:** This error can occur when the group and members of IMS OTMA are not correctly defined. Use the IMS Connect `VIEWDS` or `VIEWHWS` commands to view the status of the datastores in the system and determine which datastores were not able to be opened. If the problem persists, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

---

**HWSD0280I DATASTORE COMMUNICATION FUNCTION CLOSED; M=*mc***

**Explanation:** The communication facility for datastores has become inactive.

In the message text:

- *mc* identifies the module issuing the message.

**Module:** DOC3— HWSDOC30

**System Action:** This message is issued when all communications with the datastores have terminated and during IMS Connect shutdown.

---

**HWSD0282I COMMUNICATION WITH DS=*did* CLOSED; M=*mc***

**Explanation:** Communication for the named datastore has terminated.

In the message text:

- *did* identifies the datastore.
- *mc* identifies the module issuing the message.

**Module:**

- DSCL — HWSDSCL0
- DREC — HWSDREC0

**System Action:**

**HWSDSCL0**
: A CLOSEDS command has successfully completed.

**HWSDREC0**
: The connection to the named datastore has terminated.

This message is issued when a `CLOSEDS` command has successfully completed.

---

**HWSD0284I COMMUNICATION WITH DS=*did* STOPPED; M=*mc***

**Explanation:** Communication for the named datastore has stopped.

In the message text:
- *did* identifies the datastore.
- *mc* identifies the module issuing the message.

**Module:** DSCM— HWSDSCM0

**System Action:** This message is issued when a STOPDS command has successfully completed.

**HWSD0286I COMMUNICATION WITH DS=*did* STOPPED DUE TO COMMUNICATION ERROR; M=*mc***

**Explanation:** Communication for the named datastore stops because of an error.

In the message text:
- *did* identifies the datastore.
- *mc* identifies the module issuing the message.

**Module:** DSCM— HWSDSCM0

**System Action:** This message is issued when a communication error occurs with a datastore. Stop (/STOP OTMA) and restart (/START OTMA) OTMA and then close (CLOSEDS) and reopen (OPENDS) the datastore.

**HWSD0290I Connected to DATASTORE=*did*; M=*mc***

**Explanation:** Communication has been established with the named datastore.

In the message text:
- *did* identifies the datastore.
- *mc* identifies the module issuing the message.

**Module:**
- DSC1— HWSDSC10
- DREC — HWSDREC0

**System Action:**

**HWSDSC10**
A connection has been established with a datastore. This might occur during IMS Connect startup or at the successful completion of an OPENDS command.

**HWSDREC0**
A connection has been re-established with a datastore.

**HWSD0292I CONNECTION TO DATASTORE=*did*; FAILED; M=*mc***

**Explanation:** Communication has not been established with the named datastore. The datastore has not joined the XCF group yet.

In the message text:
- *did* identifies the datastore.
- *mc* identifies the module issuing the message.

**Module:** DSC1 — HWSDSC10

**System Action:** This message is issued when a connection has not been established with a CONFIG file defined datastore. This will occur during IMS Connect startup.

**HWSI1605W GETMAIN FOR OTOKEN AND REGISTRATION CONTROL BUFFER FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage for the OTOKEN buffer could not be allocated.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 63 for an explanation of service and return codes.

*Table 63. Service and Return Code Explanation*

| Service code | Short Explanation | Return code | Meaning |
|---|---|---|---|
| GETOTOKEN | BPEGETM, the system service used to acquire the OTOKEN. | 4 | An incorrect or unsupported subpool is specified. |
| | | 8 | A zero length is requested. |
| | | 12 | Unable to obtain the requested storage (MVS GETMAIN failed). |

**Module:** OMXR HWSOMXRG

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the reason size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide JCL, SYSLOG, and dump if available.

**HWSI1615W SCI FUNC=function, ERROR FOR IMSPLEX ENVIRONMENT; DS=*ipid*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function of an SCI call terminated in error for the named IMSplex.

- *ipid* identifies the IMSplex .
- *rc* identifies the SCI return code.
- *sc* identifies the SCI service code. Service code contains codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

**Module:**

- OXMT HWSOMXMT
- OMXRC HWSOMXRC

**System Action:** This message is issued when a transmit or receive to or from IMSplex occurs. The connection will be lost.

**HWSI1618W SCI IS NOT EXECUTING, FOR IMSPLEX=*ipid*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The IMS command request sent to the IMS OM was rejected.

- *ipid* identifies the IMSplex
- *rc* identifies the return code
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 64 for an explanation of the service and return codes.

*Table 64. Service and Return Code Explanation*

| Service code | Return code | Meaning |
|---|---|---|
| SCI Reason Code | X'01nnnnnn' | SCI return code. |
| OM Reason Code | X'02nnnnnn' | OM return code. |
| See *IMS Version 8: Common Service Layer Guide and Reference* for more information about SCI and OM return and reason codes. | | |

**Module:** OMXM HWSOMXMT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** An SCI request has been rejected. The SCI has been terminated by means other than an IMS Connect `STOPIP` command. If the return code is X'01nnnnnn', then SCI needs to be restarted. If the return code is X'02nnnnnn', then OM needs to be restarted.

**HWSI1619W OM IS NOT ACTIVE FOR IMSPLEX=*ipid*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The SCI interface has rejected the request. OM is not active.

- *ipid* identifies the IMSplex
- *rc* identifies the return code
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 65 for an explanation of service and return codes.

*Table 65. Service and Return Code Explanation*

| Service code | Return code | Meaning |
|---|---|---|
| SCI Reason Code | X'01nnnnnn' | SCI return code. |
| OM Reason Code | X'02nnnnnn' | OM return code. |
| See *IMS Version 8: Common Service Layer Guide and Reference* for more information about SCI and OM Return and Reason codes. | | |

**Module:** OMXM HWSOMXMT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** An SCI request has been rejected. OM has been terminated. If the return code is X'01nnnnnn', then SCI needs to be restarted. If the return codes is X'02nnnnnn', then OM needs to be restarted.

**HWSI1620W COMMAND FAILURE: CMD CMD ERROR FOR IMSPLEX=*ipid*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The SCI request was rejected and the OM command structure that passed is invalid.

- *ipid* identifies the IMSplex
- *rc* identifies the return code
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 66 for an explanation of service and return codes.

*Table 66. Service and Return Code Explanation*

| Service code | Return code | Meaning |
|---|---|---|
| SCI Reason Code. | X'01nnnnnn' | SCI return code. |

Table 66. Service and Return Code Explanation (continued)

| Service code | Return code | Meaning |
|---|---|---|
| OM Reason Code. | X'02nnnnnn' | OM return code. |
| See the *IMS Version 8: Common Service Layer Guide and Reference* for more information about SCI and OM Return and Reason codes. | | |

**Module:** OMXM HWSOMXMT

**System Action:** This message is issued and IMS Connects continues to run.

**System Programmer Response:** An SCI request has been rejected. OM has rejected the command. The command structure is invalid. Correct the command structure and retry the command.

**HWSI1705W GETMAIN FOR CTOKEN AND REGISTRATION CONTROL BUFFER FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** The storage buffer could not be allocated.

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 67 for an explanation of service and return codes.

Table 67. Service and Return Code Explanation

| Service code | Brief explanation | Return code | Meaning |
|---|---|---|---|
| GETCTOKN | BPEGETM, the system service used to acquire the CTOKEN failed. | 4 | An incorrect or unsupported subpool is specified or there is no storage available. |

**Module:** OMXO HWSOMXOT

**System Action:** This messaged issued and IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide JCL, SYSLOG, and dump, if available.

**HWSI1720W REGISTRATION TO SCI FAILED FOR IMSPLEX=*ipid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect attempt to register with the Structure Call Interface (SCI) has failed. This may be due to the fact that the SCI address space has not been started. As soon as SCI is started, the IMS Connect command `OPENIP` for the named IMSplex (ID=name that was specified in the configuration file) can be issued.

- *ipid* identifies the IMSplex .
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 68 for an explanation of the service and return code.

Table 68. Service and Return Code Explanation

| IMSplex name | Return code | Reason code |
|---|---|---|
| id name specified in the IMS Connect configuration file | See the IMS V8 library for SCI return and reason codes. | |

**Module:** OMXO HWSOMXOT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See the IMS V9 library to determine the SCI reason for the registration failure.

**HWSI1754W UNABLE TO NOTIFY IMSPLEX=*ipid*, SCHEDULER OF COMMUNICATION ERROR; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to obtain required storage to process the request, and is unable to notify the scheduler.

- *ipid* identifies the IMSplex .
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 69 on page 183 for an explanation of service and return codes.

*Table 69. Service and Return Code Explanation*

| Service code | Short Explanation | Return code | Meaning |
|---|---|---|---|
| GETFWEB | BPECBGET, the system service used to acquire an FWE to notify all datastores to close. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:** OMXM HWSOMXMT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSI1815W DEREGISTRATION FAILED FOR MEMBER=*member*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to deregister is unsuccessful.

- *member* identifies the IMS Connect IMSplex member name.
- *rc* identifies the return code.
- *sc* identifies the service code. Service code contains codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

**Module:** OMXC HWSOMXCN

**System Action:** This message is issued and IMS Connects continues to run.

**System Programmer Response:** See the IMS V8 library for an explanation of the specified return code.

**HWSI1816W THE SCI IS NOT AVAILABLE: MEMBER=*ipid*, STATE=*st*, M=*mc***

**Explanation:** The SCI address space has terminated.

- *ipid* identifies the IMSplex .
- *st* identifies the SCI state.
  - DISC Disconnected. The SCI address space was present; however, it has been terminated, either normally or abnormally.
- *mc* identifies the module issuing the message.

**Module:** OMXC HWSOMXCN

**System Action:** This message is issued and IMS Connects continues to run.

**System Programmer Response:** An SCI request has been rejected. The SCI has been terminated by means other than an IMS Connect `STOPIP` command. The SCI needs to be restarted.

**HWSL0101I HWS CLEANUP SUCCESSFUL**

**Explanation:** IMS Connect Local Option resource cleanup was successfully completed during termination.

**Module:** HWSRSM00

**System Action:** The message is issued and IMS Connect terminates.

**HWSL0103I CLEANUP SUCCESSFUL: Client=*cccccccc***

**Explanation:** The IMS Connect resource manager successfully cleaned the interface storage in the client address space identified in *cccccccc*. This message is issued in the client address space. In the message text:

*cccccccc*
: The client address space name. This name is typically the jobname of the web server where the client servlet is running.

  **Important:** The client address space name is different from the Client ID used in the input or output of IMS Connect commands such as STOPCLNT and VIEWHWS.

**Module:** HWSRSM20

**System Action:** The message is issued and IMS Connect terminates.

**HWSL0104W CLEANUP FAILED: CLIENT= *cccccccc*, RSN=*rrr***

**Explanation:** The IMS Connect resource manager encountered a problem while cleaning up the interface storage associated with the client in the client address space. The reason code identifies the problem. The message is issued in the client address space. In the message text:

*cccccccc* identifies the client address space name. This name is typically the job name of the web server (for example, IMWEBSRV) where the client servlet is running.

**Important:** the client address space is different from the Client ID that is used in the input or output of IMS Connect commands such as STOPCLNT and VIEWHWS.

*rrr* is one of the following reason codes:
- 104: A CGCT block was damaged.
- 108: The CCIB block was damaged.

- 10C: An error occurred when the CCIB storage was released.
- 110: A CRET block was damaged.
- 114: An error occurred when the storage for a CRET block was released.
- 118: HWSRSM20 abended for an unknown reason.
- 11C: An unknown error occurred.

**Module:** HWSRSM20

**System Action:** The message is issued, and IMS Connect terminates.

**System Programmer Response:** This error message indicates that CSA storage might not be available. Contact the IBM Support Center.

**HWSL0105I INTF CLEANUP SUCCESSFUL: Client=*cccccccc***

**Explanation:** Before terminating, IMS Connect successfully posted or resumed all outstanding HWS requests from the Local Option client. In the message text:

*cccccccc*
: The client address space name. This name is typically the jobname of the web server where the client servlet is running.

  **Important:** The client address space name is different from the Client ID used in the input or output of IMS Connect commands such as STOPCLNT and VIEWHWS.

**Module:** HWSRSM10

**System Action:** The message is issued and IMS Connect terminates.

**HWSL0106W INTF CLEANUP FAILED: CLIENT= *cccccccc*, RSN=*rrr***

**Explanation:** When the IMS Connect address space terminated, the IMS Connect resource manager that was monitoring IMS Connect for the client failed during cleanup. The reason code identifies the problem that was encountered. In the message text:

*cccccccc* identifies the client address space name. This name is typically the job name of the web server (for example, IMWEBSRV) where the client servlet is running.

*rrr* is one of the following reason codes:
- 104: The resource manager could not obtain common storage for a CXSH block to notify the client that IMS Connect terminated.
- 108: The resource manager could not schedule an SRB to the client address space to notify the client that IMS Connect terminated.
- 10C: The resource manager could not schedule an SRB to the client address space to clean up the IMS connect interface blocks.
- 110: The resource manager (HWSRSM10) abended.

**Module:** HWSRSM10

**System Action:** The message is issued, and IMS Connect terminates.

**System Programmer Response:** If the client address space terminates before the IMS Connect resource manager completes processing, you might receive message HWS0106w with either reason code 108 or 10C. If you do, other IMS Connect resource managers have cleaned up IMS Connect interface storage, and no action is required. If you do not, then contact the IBM Support Center. If you receive reason code 110, print the records in SYS1.LOGREC for information on the abend.Provide the JCL, SYSLOG, and dump if available.

**HWSL0111W HWS INTERFACE ABEND *abend_code* PSW=*psw* R15=*r15* MODULE *module_addr* STATUS**

**Explanation:** An abend occurred in the interface between the client and IMS Connect during the processing of an IMS Connect request. In the message text:

*abend_code*
: Identifies the abend that occurred (Sxxx for system abends and Uxxxx for user abends).

*psw*
: Identifies the PSW contents at the time of the abend.

*r15*
: Identifies the contents of Register 15 at the time of the abend. For some abends, this value is the abend subcode.

*module_addr*
: Identifies the name of the IMS Connect interface module that detected the abend. This value is not necessarily the module that abended, but the module whose recovery routine (ESTAE or FRR) was driven because of the abend. Possible modules are:

  **HWSREG20**
  : The abend occurred during registration with IMS Connect.

  **HWSRQS00**
  : The abend occurred on the input side of the interface, which sends the request to the IMS Connect address space.

  **HWSSRB00**
  : The abend occurred on the output side of the interface, which returns the result of an HWS request from IMS Connect to the client.

**Status**
: Status is a text string that indicates where the abend occurred. This information is not provided for all modules. For example, if the module is

HWSREG20, status is blank. If the module is HWSRQS00 or HWSSRB00, status can have the following values:

**BEFORE COPY**
The abend occurred before the requested data was copied to the IMS Connect address space.

**IN COPY**
The abend occurred while the requested data was being copied to the IMS Connect address space. This abend occurs when bad data is passed from the client.

**AFTER COPY**
The abend occurred after the data is copied and queued to the IMS Connect address space.

**STATUS UNKWN**
The FRR could not determine the status of the request when the abend occurred.

HWSRQS00 provides this additional value:

**IN ENQUEUE**
The abend occurred when the request was queued to the IMS Connect address space.

HWSSRB00 provides this additional value:

**IN POST**
The abend occurred when the client was being posted to wake it after a request had completed.

**Module:**

- HWSREG20
- HWSRQS00
- HWSSRB00

**System Action:** The message is issued to the client application, and IMS Connect continues to run.

**System Programmer Response:** Save a copy of the dump product, and save or print a copy of the LOGREC records related to this abend. If the client is an IBM product, contact the IBM Support Center. If the client is not an IBM product, contact the supplier of the client.

**HWSL0281I CONNECT REJECTED FOR CLIENT=*client*, USERID=*userid*; INSUFFICIENT AUTHORITY TO HWS ICON_NAME; RACROUTE AUTH R15=*r15*, RC=*rc*, RSN=*rsn***

**Explanation:** A client attempted to connect to IMS Connect using the local option but the client was not authorized to access IMS Connect. IMS Connect issues a RACROUTE REQUEST=AUTH call to determine if the connecting client has the appropriate authority to access IMS Connect using the local option. IMS Connect uses the job user ID of the client to perform the authorization. See “Defining IMS Connect Security” on page 21.

- *client* refers to the client ID that is attempting to connect.
- *userid* is the user ID associated with the address space of the client. If this field contains “NONE,” the client is running with no user ID specified.
- *icon_name* refers to the IMS Connect to which the client is trying to connect.
- *r15* refers to the value in register 15 from the RACROUTE call.
- *rc* refers to the RACF return code from the RACROUTE call.
- *rsn* refers to the RACF reason code from the RACROUTE call.

**System Action:** The connection request is rejected and the client is not allowed to access the requested IMS Connect.

**System Programmer Response:** If the indicated user should be allowed to access the requested IMS Connect, authorize the user to IMS Connect with at least RACF UPDATE authority. If the indicated user should not be allowed to access the requested IMS Connect, you should determine why the user is trying to connect to it and take appropriate action to protect against unauthorized or malicious access.

**HWSM0500E FUNCTION WORK ELEMENT PROCESSING FAILURE, FUNC=*func*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function work element (FWE) can not be processed. The FWE requests work between and within the components. This structure contains the function and parameters that a service requires for processing.

- *func* identifies the function requested.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 70 for an explanation of the service and return code.

*Table 70. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVFUNC | The function requested in the FWE is incorrect. | 4 | This is a processing error. |

**Module:**

- OOCC HWSOOC0
- OSCH HWSOSCH0
- DCVC HWSDCVC0

**System Action:** This message is issued and, if possible, the requester of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting database to find a correction for the problem. I none exists, contact the IMS Support Center. Provide the JCL, SYSLOG, and dump, if available.

**HWSM0502W FWE FUNCTION=*func* FAILED FOR IMSPLEX=*ipid*, COMMAND=*hwscmd* IN PROGRESS; M=*mc***

**Explanation:** The function *func* can not be processed because the command identified by *hwscmd* is already being processed.

- *func* identifies the function requested.
- *ipid* identifies the IMSplex connection.
- *hwscmd* identifies the IMS Connect command in progress.
- *mc* identifies the module issuing the message.

**Module:** DSCM HWSDSCM0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** The IMS Connect command is progress is terminating the IMSplex; therefore, any new function for that IMSplex can not be processed.

**HWSM0504W COMMAND=*hwscmd* FAILED FOR IMSPLEX=*ipid*, COMMAND=*prev_hwscmd* ALREADY IN PROGRESS; M=*mc***

**Explanation:** The IMS Connect command entered for the IMSplex, *hwscmd*, cannot be processed because a command for that IMSplex, *prev_hwscmd*, is already in progress.

- *hwscmd* identifies the IMS Connect command that was blocked by prev_hwscmd from being run.
- *ipid* identifies the IMSplex.
- *prev_hwscmd* identifies the IMS Connect command that is blocking *hwscmd* from running.
- *mc* identifies the module issuing the message.

**Module:** DSCM HWSDSCM0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** The IMS Connect command in progress is terminating the IMSplex; therefore, any new commands cannot be processed. If the IMS Connect command (*hwscmd*) was CLOSEHWS, the IMS Connect terminates after processing of *prev_hwscmd* completes.

**HWSM0510W STOPIP COMMAND FAILED DUE TO IMSPLEX IN DISCONNECT STATE; M=*mc***

**Explanation:** The `STOPIP` command was issued; however, the IMSplex connection is in a DISCONNECT state. Therefore, the `STOPIP` command cannot be processed. When the SCI address is restarted, IMS Connect will automatically reconnect to SCI. When the connection has been reestablished, the `STOPIP` command can be issued.

*mc* identifies the module issuing the message.

**Module:** OSCH HWSOSCH0

**System Action:** This message is issued and the `STOPIP` command is ignored.

**System Programmer Response:** The DISCONNECT state has the same effect as a STOPPED state. If the SCI address space is restarted, the connection will be reestablished. When the connection is reestablished, the `STOPIP` command can be issued.

**HWSM0522W UNABLE TO START TRANSMIT/RECEIVE THREADS FOR IMSPLEX=*ipid*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the transmit or receive thread structure, or the transmit thread or receive thread cannot be scheduled. A transmit thread and receive thread is allocated for each IMSplex that is defined for message transmission and reception.

- *ipid* identifies the IMSplex.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed request.
- *mc* identifies the module issuing the message.

See Table 71 on page 187 for an explanation of service and return codes.

*Table 71. Service and Return Code Explanation*

| Service Code | Short Explanation | Return code | Meaning |
|---|---|---|---|
| GETDSBB | BPECBGET, the system service used to acquire the IMSplex (DSB) for the transmit and receive threads. This is the execution block for a thread. | 4 | An incorrect CBTE address is passed to the CBGET routine. This is an internal error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETC01K | BPECBGET, the system service used to acquire the common 1024 byte (C01K) for the controller. The area is used as a work area. | 4 | An incorrect CBTE address is passed to the CBGET routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU) for the transmit and receive threads. | 4 | An incorrect CBTE address is passed to the CBGET routine. This is an internal error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the scheduler controller thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is on the ROUTINE= parameter. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread |
| | | 28 | The initial post of the thread failed. |

**Module:**

- DSC1 HWSDSC10
- DSCM HWSDSCM0

**System Action:** This message is issued and, IMS Connect continues to run without this IMSplex.

**System Programmer Response:** On the subsequent close and startup of IMS Connect, ensure that the region size of the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

**HWSM0527W CLOSE FAILED FOR IMSPLEX=*ipid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to close the named IMSplex is unsuccessful during IMS Connect shutdown.

- *ipid* identifies the IMSplex.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 72 for an explanation of service and return codes.

*Table 72. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETFWEB | BPECBGET, the system service used to acquire an FWE to notify all IMSplex to close. | 4 | An incorrect CBTE address is passed to the CBGET routine. This is an internal error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:** DOC3 HWSDOC30

**System Action:** This message is issued and, IMS Connect continues to run.

**System Programmer Response:** Storage cannot be allocated to notify the IMSplex to close. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

**HWSM0550W UNABLE TO NOTIFY MSG ORIGIN=*clientid* OF IMSPLEX COMMUNICATION ERROR; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to notify the TCP/IP client who originated the command message which is either being processed or queued for processing, that a communication error with IMS Operations Manager (OM) has occurred.

- *clientid* identifies the TCP/IP client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 73 for an explanation of the service and return code.

*Table 73. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| COMMERR | Communication error with the IMSplex | 4 | This is a processing error. |

**Module:**

- OXMT HWSOXMT0
- DSC3 HWSDSC30
- DSCE HWSDSCE0

**System Action:** This message is issued and IMS Connect continues to run. The message whose processing caused the error is discarded.

**System Programmer Response:** This error occurs when the IMSplex is no longer active or the communication linkage to IMS Connect is broken.

**HWSM0552W UNABLE TO SEND RESPONSE RECEIVED FROM IMSPLEX=*ipid* TO CLIENT=*clientid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to send the response received from the IMSplex to the required TCP/IP client.

- *ipid* identifies the IMSplex.
- *clientid* identifies the TCP/IP client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 74 on page 189 for an explanation of service and return codes.

*Table 74. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVLDTOK | Invalid server token has been detected. | 4 | Use the correct server token for the exchange of the command and command response. |
| NFNDCOMP | The component that handles the requested function cannot be found. An IMS Connect component issues an interface call for another component's service and the requested component cannot be located | 4 | This is a processing error. |
| NFNDFUNC | The requested function cannot be found. An IMS Connect component issues an interface call for another component's service and the requested service cannot be located. | 4 | This is a processing error. |
| NFNDSVT | The server table cannot be found. This table maintains the activity of a connected IMS Connect client. | 4 | This is a processing error. |

**Module:** OREC HWSOREC0

**System Action:** This message is issued and IMS Connect continues to run. The response message is discarded.

**System Programmer Response:** This error occurs when the client is no longer active and is not connected to IMS Connect. If the service code is NFNDCOMP or NFNDFUNC, this is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

---

**HWSM0554W UNABLE TO NOTIFY IMSPLEX=*ipid* SCHEDULER OF COMMUNICATION ERROR; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to notify the scheduler controller for the named IMSplex that a communication error has occurred. When this condition occurs, IMS Connect views the named IMSplex as active. However, messages queued for the IMSplex are not sent.

- *ipid* identifies the IMSplex.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain codes that identify specific errors or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 75 for an explanation of the service and return code.

*Table 75. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| COMMERR | Communication error with the IMSplex. | 4 | This is a processing error. |

**Module:**

- OREC HWSOREC0
- OXMT HWSOXMT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Issue the `STOPIP` command to terminate the IMSplex. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

---

**HWSM0560I IMSPLEX=*ipid* THREAD TERMINATED; M=*mc***

**Explanation:** The IMSplex transmit thread or receive thread has terminated.

- *ipid* identifies the IMSplex
- *mc* identifies the module issuing the message

**Module:**

- OREC HWSOREC0
- OXMT HWSOXMT0

**System Action:** This message is issued when the IMSplex thread has terminated.

**HWSM0570W IMSPLEX OPEN FAILED; R=*rc*, M=*mc***

**Explanation:** Communication with the IMSplex failed during IMS Connect startup or in response to an IMS Connect `OPENIP` command and resulted in the failure of the IMSplex open function.

- *rc* identifies the return code.
- *mc* identifies the module issuing the message.

**Module:** OOC1 HWSOOC10

**System Action:** This message is issued when communication to IMSplex fails due to a communications failure with the IMSplex. See message HWSI1605W on page 180 for additional information related to this failure.

**System Programmer Response:** This error can occur when IMSplex is not correctly defined. Use `VIEWIP` or `VIEWHWS` commands to view the status of the IMSplex. If the problem persists, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

**HWSM0580I IMSPLEX COMMUNICATION FUNCTION CLOSED; M=*mc***

**Explanation:** The communication facility for IMSplex has become inactive.

- *mc* identifies the module issuing the message.

**Module:** DOC3 HWSDOC30

**System Action:** This message is issued when all communications with the IMSplex have terminated and during IMS Connect shutdown.

**HWSM0582I COMMUNICATION WITH IMSPLEX=*ipid* CLOSED; M=*mc***

**Explanation:** Communication for the named IMSplex has terminated.

- *mc* identifies the module issuing the message.

**Module:** DSCL HWSDSCL0

**System Action:** This message is issued when the `CLOSEIP` command has successfully completed.

**HWSM0584 COMMUNICATION WITH IMSPLEX=*ipid* STOPPED; M=*mc***

**Explanation:** Communication for the named IMSplex has stopped.

- *ipid* identifies the IMSplex
- *mc* identifies the module issuing the message.

**Module:** DSCM HWSDSCM0

**System Action:** This message is issued when a `STOPIP` command has successfully completed.

**System Programmer Response:**

**HWSM0590I CONNECTED TO IMSPLEX=*ipid*; M=*mc***

**Explanation:** Communication has been established with the named IMSplex.

**Module:** OSC10 HWSOSC10

**System Action:** This message is issued when a connection has been established with the IMSplex. This might occur during IMS Connect startup or at the successful completion of an `OPENIP` command.

**HWSO1100W FAILED TO OBTAIN FREE STORAGE; R=*rc*, B=*bn*, M=*mc***

**Explanation:** The IMS Connect OTMA driver is unable to get free storage for internal buffers.

- *rc* identifies the return code.
- *bn* identifies the buffer name.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This error can occur when not enough storage is available to complete the process. If the problem persists, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSO1101W FAILED TO RELEASE STORAGE; R=*rc*, B=*bn*, M=*mc***

**Explanation:** The IMS Connect OTMA driver is unable to release storage for internal buffers.

In the message text:

- *rc* identifies the return code.
- *bn* identifies the buffer name.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSO1105W GETMAIN FOR OTOKEN + IXCQUERY CONTROL BUFFER FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage for the OTOKEN buffer could not be allocated.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 76 for an explanation of service and return codes.

*Table 76. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETOTOKN | BPEGETM, the system service used to acquire the OTOKEN. | 4 | An incorrect or unsupported subpool is specified. |
| | | 8 | A zero length is requested. |
| | | 12 | Unable to obtain the requested storage (MVS GETMAIN failed). |

**Module:**

- DDXR— HWSDDXRG

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSO1110W IXCQUERY FAILED FOR OTMA SYSPLEX ENVIRONMENT; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to query OTMA sysplex environment information (REQINFO=SYSPLEX) is unsuccessful.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the reason code.
- *mc* identifies the module issuing the message.

Return codes (decimal): See *MVS/ESA SP Authorized Assembler Reference*.

Reason codes (decimal): See *MVS/ESA SP Authorized Assembler Reference*.

**Module:**

- DDXR— HWSDDXRG

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See *MVS/ESA SP Authorized Assembler Reference* and take appropriate action.

**HWSO1115W XCF FUNC=*function*, ERROR FOR OTMA SYSPLEX ENVIRONMENT; DS=*did*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function on a XCF call terminated in error for the named datastore.

In the message text:

- *function* identifies the function (Transmit or Receive).
- *rc* identifies the XCF return code.
- *sc* identifies the XCF reason code.
- *mc* identifies the module issuing the message.

**Module:**

- DXMT— HWSDDXMT
- DXRC— HWSDDXRC

**System Action:** This message is issued when a transmit or receive to or from IMS occurs. The connection will be lost.

| **HWSO1120W XCF FUNC=TRANSMIT XCF ENVIRONMENT ERROR; DS=*did*, R=*rc*, S=*sc*, M=*mc***

| **Explanation:** The function on a XCF call terminated in error for the named datastore.

| In the message text:

- | *function* identifies the function (Transmit or Receive).
- | *rc* identifies the XCF return code.
- | *sc* identifies the XCF reason code.
- | *mc* identifies the module issuing the message.

| **Module:**

- | DXMT— HWSDDXMT
- | DXRC— HWSDDXRC

| **System Action:** This message is issued when a transmit or receive to or from IMS occurs. The connection to the named datastore is terminated.

**HWSO1205W GETMAIN FOR CTOKEN + IXCJOIN CONTROL BUFFER FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage for the CTOKEN + IXCJOIN buffer could not be allocated.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 77 for an explanation of service and return codes.

*Table 77. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETCTOKN | BPEGETM, the system service used to acquire the CTOKEN. | 4 | An incorrect or unsupported subpool is specified. |
| | | 8 | A zero length is requested. |
| | | 12 | Unable to obtain the requested storage (MVS GETMAIN failed). |

**Module:**

- DDXO— HWSDDXOT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSO1210W IXCQUERY FAILED FOR GROUP=*group*, MEMBER=*tmember*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to query OTMA group information (REQINFO=GROUP) is unsuccessful.

In the message text:

- *group* identifies the XCF group name.
- *tmember* identifies the IMS's XCF target member name.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 78 for an explanation of the service and return code,

*Table 78. Service and Return Code Explanation*

| Group name | Member name | Return code | Reason code |
|---|---|---|---|
| XCF group name | IMS XCF member name | See *OS/390 MVS Programming Sysplex Service Reference* manual. | |

**Module:**

- DDXO— HWSDDXOT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See *MVS/ESA SP Authorized Assembler Reference* and take the appropriate action.

**HWSO1215W XCF GROUP=*group*, MEMBER=*tmember* IS NOT ACTIVE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** The target XCF member is not active.

In the message text:

- *group* identifies the XCF group name.
- *tmember* identifies the IMS's XCF target member name.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 79 on page 193 for an explanation of the service and return code.

Table 79. Service and Return Code Explanation

| Group name | Member name | Service code | Return code | Meaning |
|---|---|---|---|---|
| XCF group name | IMS XCF member name | NOTACTV | 4 | The target member is not active. |

**Module:**
- DDXO— HWSDDXOT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Check the status of the target member and restart the target member.

**HWSO1220W IXCJOIN FAILED FOR GROUP=*group*, MEMBER=*member*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to join the XCF group is unsuccessful.

In the message text:
- *group* identifies the XCF group name.
- *member* identifies IMS Connect's XCF member name.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 80 for an explanation of the service and return code.

Table 80. Service and Return Code Explanation

| Group name | Member name | Return code | Reason code |
|---|---|---|---|
| XCF group name | IMS Connect's XCF member name | See *OS/390 MVS Programming Sysplex Service Reference* manual. | |

**Module:**
- DDXO— HWSDDXOT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See *MVS/ESA SP Authorized Assembler Reference* and take the appropriate action.

**HWSO1305W CBGET FOR C512 BLOCK FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage for the client bid buffer cannot be allocated.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 81 for an explanation of return and service codes.

Table 81. Service and Return Code Explanation

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETC512 | BPECBGET, the system service used to acquire C512. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:**
- DDXC— HWSDDXC

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This is probably a storage error. Ensure that the region size for IMS Connect is large enough. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSO1310W IXCMSGO FAILED FOR CLIENT BID GROUP=*group*, MEMBER=*member*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to send a client bid to IMS OTMA is unsuccessful.

In the message text:
- *group* identifies the XCF group name.
- *member* identifies the IMS Connect's XCF member name.

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 82 for an explanation of the return and reason code.

Table 82. Return and Reason Code Explanation

| Group name | Member name | Return code | Reason code |
|---|---|---|---|
| XCF group name | IMS Connect's XCF member name | See *OS/390 MVS Programming Sysplex Service Reference* manual. | |

**Module:**
- DDXC— HWSDDXCN

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See *MVS/ESA SP Authorized Assembler Reference* for the possible cause of the specified return and reason codes.

**HWSO1315W IXCLEAVE FAILED FOR GROUP=*group*, MEMBER=*member*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to leave the XCF group is unsuccessful.

In the message text:
- *group* identifies the XCF group name.
- *member* identifies the IMS Connect's XCF member name.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 83 for an explanation of the return and reason code.

Table 83. Return and Reason Code Explanation

| Group name | Member name | Return code | Reason code |
|---|---|---|---|
| XCF group name | IMS Connect's XCF member name | See *OS/390 MVS Programming Sysplex Service Reference* manual. | |

**Module:**
- DDXC— HWSDDXCN

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See *MVS/ESA SP Authorized Assembler Reference* for the possible cause of the specified return and reason codes.

**HWSO1320W CLIENT BID FAILED FOR GROUP=*group*, MEMBER=*member*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** A client bid with IMS OTMA is unsuccessful.

In the message text:
- *group* identifies the XCF group name.
- *member* identifies the IMS Connect's XCF member name.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

Table 84. Service and Return Code Explanation

| Group name | Member name | Service code | Return code | Meaning |
|---|---|---|---|---|
| XCF group name | IMS Connect's XCF member name | CBERROR | See the *IMS Open Transaction Manager Access Guide*. | This is a client bid error. |

**Module:**
- DDXC— HWSDDXCN

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See the *IMS Open Transaction Manager Access Guide* for the possible cause of the specified return code.

**HWSO1325W RACFOUTE REQUEST=TOKENXTR FAILED FOR R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to extract a utoken for IMS Connect ASID is unsuccessful.

In the message text:
- *rc* identifies the SAF return code. See the RACROUTE macro reference for MVS for more information.

- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

Return codes (decimal): See the RACROUTE macro reference for MVS.

Reason codes (decimal): See the RACROUTE macro reference for MVS.

**Module:**

- DDXC— HWSDDXCN

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** See the RACROUTE macro reference for MVS for the possible cause of the specified return and reason codes.

**HWSP1400W IPV6 PROCESSING NOT ENABLED; FUNC=*fn*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to get the IPv6 socket.

In the message text:

- *fn* identifies the function code.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain either codes that specifically identify the error or codes returned by called services which failed to complete the request.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run with IPV4 support.

**System Programmer Response:** Ensure that the requirement for z/OS R1V4 is met and enable the TCP/IP stack for IPV6 processing by tailoring the BPXPRMxx member. See “Enabling Support for Internet Protocol Version 6” on page 20 for more information.

**HWSP1402W SSL PROCESSING NOT ENABLED; FUNC=*fn*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to retrieve SSL support.

In the message text:

- *fn* identifies the function code.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes contain either codes that specifically identify the error or codes returned by called services which failed to complete the request.
- *mc* identifies the module issuing the message.

**Module:** HWSDOPN0

**System Action:** This message is issued. The SSL socket is closed. IMS Connect continues to run fully. If the message occurred during the SSL environment initialization, the port may be closed.

**System Programmer Response:** Ensure that the requirement for z/OS V1.4 is met.

**HWSP1405W FAILED TO OBTAIN FREE STORAGE; R=*rc*, B=*bn*, M=*mc***

**Explanation:** The IMS Connect OTMA driver is unable to get free storage for internal buffers.

In the message text:

- *rc* identifies the return code returned by MVS for an MVS GETMAIN failure.
- *bn* identifies the buffer name.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This error can occur when not enough storage is available to complete the process. If the problem persists, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSP1410W FAILED TO RELEASE STORAGE; R=*rc*, B=*bn*, M=*mc***

**Explanation:** The IMS Connect OTMA driver is unable to release storage for internal buffers.

In the message text:

- *rc* identifies the return code returned by MVS for an MVS GETMAIN failure.
- *bn* identifies the buffer name.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSP1415E HWSP1415E: TCP/IP SOCKET FUNCTION CALL FAILED; F=*fn*, R=*rc*, E=*ec*, M=*mc***

**Explanation:** The IMS Connect TCP/IP driver is unable to perform the specified socket function. HWSP1415E is issued during normal execution of IMS Connect. HWSP1415I is issued during shutdown of IMS Connect.

In the message text:

- *fn* identifies the TCP/IP socket function call.

- *rc* identifies the TCP/IP return code.
- *ec* identifies the TCP/IP error code.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** For the possible cause of the specified return code and error code, see *TCP/IP Application Programming Interface Reference*.

**HWSP1415I HWSP1415I: TCP/IP SOCKET FUNCTION CALL FAILED; F=*fn*, R=*rc*, E=*ec*, M=*mc***

**Explanation:** The IMS Connect TCP/IP driver is unable to perform the specified socket function. HWSP1415E is issued during normal execution of IMS Connect. HWSP1415I is issued during shutdown of IMS Connect.

In the message text:
- *fn* identifies the TCP/IP socket function call.
- *rc* identifies the TCP/IP return code.
- *ec* identifies the TCP/IP error code.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** For the possible cause of the specified return code and error code, see *TCP/IP Application Programming Interface Reference*.

**HWSP1420E PORT NUMBER CONTAINS NON-NUMERIC VALUE; P=*portid*, M=*mc***

**Explanation:** The IMS Connect TCP/IP driver is unable to convert the *portid* character string to a numeric value.

In the message text:
- *portid* identifies the port id character string in the `PORT` substatement of the `TCPIP` statement in the IMS Connect configuration member, `HWSCFGxx`.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Check the `PORT` substatement of the `TCPIP` statement in the IMS Connect configuration member, `HWSCFGxx`, for the correct numeric characters. Correct the problem and restart IMS Connect.

**HWSP1425E WAIT ECB FAILED; F=*fn*, C=*pc*, M=*mc***

**Explanation:** The IMS Connect TCP/IP driver is informed of an unsuccessful post code.

In the message text:
- *fn* identifies the function performed.
- *pc* identifies the post code set by IMS Connect.
- *mc* identifies the module issuing the message.

**Module:**
- HWSSDOTD
- HWSSDCON
- HWSSDDSC
- HWSSDRCV
- HWSSDTTD
- HWSSDXMT

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Check the post code for the possible cause. For the post code, see “IMS Connect Post Codes” on page 229. This error is probably an internal error. Search the problem reporting database to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

**HWSP1426E WAIT ECB FAILED; F=*fn*, C=*pc*, M=*mc***

**Explanation:** An invalid post code was returned to IMS Connect.

In the message text:
- *fn* identifies the function performed.
- *pc* identifies the post code set by IMS Connect.
- *mc* identifies the module issuing the message.

**Module:**
- - HWSSDOTD

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Check the post code for the possible cause. For the post code, see “IMS Connect Post Codes” on page 229.

**HWSP1430E TCP/IP INTERNAL ERROR; F=*fn*, R=*rc*, E=*ec*, M=*mc***

**Explanation:** TCP/IP is unable to perform the specified socket function.

In the message text:
- *fn* identifies the TCP/IP socket function call.
- *rc* identifies the TCP/IP return code.
- *ec* identifies the TCP/IP error code.
- *mc* identifies the module issuing the message.

**Module:**

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** For the possible cause of the specified return code and error code, see

*TCP/IP Application Programming Interface Reference*.

**HWSP1435E SOCKET CLOSED; REQUEST MESSAGE INCOMPLETE; M=*mc***

**Explanation:** The TCP/IP socket closes before all the data has been received.

In the message text:

- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The connection has been terminated by the client code, and IMS Connect has received either no data or partial data from the client. This error can occur if you specified a TCP/IP value of SO_LINGER=Y,VALUE=0 or SO_LINGER=NO. Instead, specify SO_LINGER=Y,VALUE=10. The VALUE parameter should be any value other than 0. See *TCP/IP Application Programming Interface Reference* for more information on SO_LINGER= and VALUE=. The request message is discarded.

**HWSP1440E INVALID LENGTH SPECIFIED IN MESSAGE PREFIX; L=*ll*, M=*mc***

**Explanation:** The length field in the message prefix contains an invalid value. A valid message length value is between 12 and 10,000,000 inclusive, and it must be equal to the exact data being sent.

In the message text:

- *ll* identifies the length specified in the message prefix. This is the length of the entire message including the 12-byte message prefix.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded.

**HWSP1445E UNKNOWN EXIT IDENTIFIER SPECIFIED IN MESSAGE PREFIX; MSGID=*msgid1/msgid2*, M=*mc***

**Explanation:** The MSGID identifier in the message prefix contains an unknown identifier. Exit identifiers are given to IMS Connect in the `INIT` subroutine of the user exit.

In the message text:

- *msgid1* identifies the EBCDIC `MSGID` in the message prefix.
- *msgid2* identifies the ASCII `MSGID` in the message prefix.
- *mc* identifies the module issuing the message.

If *msgid1* and *msgid2* are both unreadable, then one of the following may have occurred:

1. The message was built incorrectly. The IRM_ID of the message is incorrect or missing.
2. The message was sent on a client-defined Secure Socket Layer (SSL) port; however the port was not defined to IMS Connect as an SSL port.

If the *msgid1* or *msgid2* is partially readable, it may mean the message was built incorrectly and the IRM_ID field contains only part of the ID. For example:

- If 4 extra bytes precede IRM_ID, you may receive one of the following messages:

```
HWSP1445E Unknown EXIT identifier specified in message prefix; MSGID=)($%*SAM/+_{}|":^
HWSP1445E Unknown EXIT identifier specified in message prefix; MSGID=^+_{}|":/)($%*SAM
```

- If 4 bytes are missing in front of IRM_ID, you may receive one of the following messages:

```
HWSP1445E Unknown EXIT identifier specified in message prefix; MSGID=PLE*)($%/+_{}|":^
HWSP1445E Unknown EXIT identifier specified in message prefix; MSGID=^+_{}|"/PLE*)($%
```

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded.

**HWSP1450E MESSAGE CONTAINS INVALID LENGTH; SEG_NO=*sn*, APP_LL=*al*, TOTAL MSG LEN=*tl*, EXPECTED MSG LEN=*el*, C=*clientid*, M=*mc***

**Explanation:** The input OTMA message contains an incorrect application data length.

In the message text:

- *sn* identifies the OTMA segment number.
- *al* identifies the application data length in the OTMA segment.
- *tl* identifies the length of the total message specified.
- *el* identifies the length of the expected message.
- *clientid* identifies the client name. It will contain blanks if the client name is not available.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded.

**HWSP1455E MESSAGE CONTAINS INVALID LENGTH; AREA_LL=*ar*, APP_LL=*al*, M=*mc***

**Explanation:** The input OTMA message contains an incorrect application data length.

In the message text:

- *ar* identifies the internal buffer length.
- *al* identifies the application data length in the OTMA segment.

- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded. This is an IMS Connect/IMS internal error. Contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSP1460E MISSING FIC IN OTMA PREFIX; M=*mc***

**Explanation:** The input OTMA message does not contain a first-in-chain (FIC) flag in the first segment.

In the message text:

- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded.

**HWSP1465E MISSING LIC IN OTMA PREFIX; SEG_NO=*sn*, M=*mc***

**Explanation:** The input OTMA message does not contain a last-in-chain (LIC) flag in the last segment.

In the message text:

- *sn* identifies the number of the segment.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded.

**HWSP1470E LOADING EXIT FAILED; EXIT=*msgid*, R=*rc*, M=*mc***

**Explanation:** IMS Connect failed to load the user exit.

In the message text:

- *msgid* identifies the `MSGID` (exit name) in the message prefix.
- *rc* identifies the return code returned by MVS from an MVS load failure.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued for each user exit that fails to load. If at least one user exit loads correctly, IMS Connect continues to run. However, the exits that failed to load will not be available to IMS Connect.

If all user exits fail to load, IMS Connect continues to run but no TCP/IP communication is established (see HWSS0785W on page 210).

**System Programmer Response:** If TCP/IP communication failed to establish because none of the exits returned a valid return code, run `CLOSEHWS` to terminate IMS Connect.

Examine the return code and resolve the problem and then restart IMS Connect to reload the exit or exits.

**HWSP1475E EXIT EXECUTION FAILED; EXIT=*msgid*, F=*fn*, R=*rc*, M=*mc***

**Explanation:** A user exit returns an incorrect return code to IMS Connect when called by IMS Connect to perform an INIT or TERM function.

In the message text:

- *msgid* identifies the `MSGID` (exit name) in the message prefix.
- *fn* identifies the function failed.
- *rc* identifies the return code returned by MVS from an MVS load failure.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued for each user exit that returns an incorrect return code. If at least one exit returns a valid return code, IMS Connect continues to run. However, the exits that failed will not be available to IMS Connect.

If all exits return an invalid return code, IMS Connect continues to run but no TCP/IP connection is established (see HWSS0785W on page 210).

**System Programmer Response:** Pass the return code and function name to the exit owner to resolve the problem.

If TCP/IP communication is not established because none of the exits returned a valid return code, run `CLOSEHWS` to terminate IMS Connect.

**HWSP1480E CONFLICT IDENTIFIERS RETURNED FROM EXIT; EXIT1=*en1*, EXIT2=*en2*, M=*mc***

**Explanation:** Multiple user exits that use the same exit name are defined in the `EXIT` substatement of the `TCPIP` statement in the `HWSCFGxx` configuration member.

In the message text:

- *en1* identifies the first exit name.
- *en2* identifies the second exit name.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run, but none of the TCP/IP communication facilities will work properly.

**System Programmer Response:** Have the owner of EXIT1 and EXIT2 resolve the naming problem, correct the exit names in the `EXIT` substatement in `HWSCFGxx`, and then shut down and restart IMS Connect.

**HWSP1485E PASSING TO TCP/IP ASYNC FAILED; F=*fn*, R=*rc*, E=*ec*, M=*mc***

**Explanation:** TCP/IP rejects the request for asynchronous function processing.

In the message text:

- *fn* identifies the TCP/IP socket function call.
- *rc* identifies the TCP/IP return code.
- *ec* identifies the TCP/IP error code.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** For the possible cause of the specified return and error codes, see *TCP/IP Application Programming Interface Reference*.

**HWSP1490E INVALID OTMA SEQUENCE NUMBER; Seg=*gn*, SEQ=*qn*, C=*cn*, M=*mc***

**Explanation:** A request message coming from a client or generated by a user exit contains an invalid sequence number in the OTMA prefix. The sequence number must match the segment number.

In the message text:

- *gn* identifies the segment number.
- *qn* identifies the sequence number.
- *cn* identifies the client name. It will contain blanks if the client name is not available.
- *mc* identifies the module issuing the message.

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is discarded.

**HWSP1495E PROTOCOL VIOLATION; R=*rc*, C=*cn*, M=*mc***

**Explanation:** IMS Connect received the input message while waiting for the response ACK/NAK.

In the message text:

- *rc* identifies the return code.
- *cn* identifies the client name. It will contain blanks if the client name is not available.
- *mc* identifies the module issuing the message.

**Module:**

- SDRC — HWSSDRCV

**System Action:** This message is issued and IMS Connect sends the NAK to IMS.

**System Programmer Response:** None. The request message is rejected.

**HWSP1500E SECURITY VIOLATION; R=*rc*, C=*cn*, RACFRC=*rrc,* RACFS=*rrs*, M=*mc***

**Explanation:** An attempt to RACF user identification and verification for the request message coming from a client or generated by a user exit routine contains the password and user ID in the OTMA prefix user data section.

In the message text:

- *rc* identifies the SAF return code.
- *cn* identifies the client name. It will contain blanks if the client name is not available.
- *rrc* identifies the RACF return code.
- *rrs* identifies the RACF reason code.
- *mc* identifies the module issuing the message.

Return codes (decimal): See the RACROUTE REQUEST=VERIFY macro reference for MVS for R=*rc*, RACFRC=*rrc*, and RACFS=*rrs* values.

**Module:**

- SDRC — HWSSDRCV

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is rejected.

**HSWP1503E SECURITY VIOLATION, NO RACROUTE CALL; R=*rc*, C=*clientid*, M=*SDRC***

**Explanation:** IMS Connect rejected the security input data of the user ID and/or the password. RACF=Y had been specified, however, no user ID and/or password was passed to IMS Connect by the user-written exit.

In the message text:

- *rc* identifies the return code from IMS Connect.
  - 255 - No OTMA security header; IMS Connect security checking cannot be done.
  - 252 - Invalid security header length. The security header length is less than X'6A', security parms are missing.
  - 248 - No password (see note 1).
  - 244 - No user ID (see note 2).
  - 242 - Invalid character detected in user ID, groupname, or password field.
  - 240 - No password and no user ID (see note 3).
- *clientid* identifies the client ID.
- *SDRC* identifies the module issuing the error message.

**Notes:**

1. There is no password in IRM, but there is a user ID in IRM. Or, there is no password or user ID in IRM; however, there is a default user ID in the IMS Connect configuration file.

2. There is no user ID in IRM and there is no default user ID in the IMS Connect configuration file.
3. There is no password or user ID in IRM and there is no default user ID in the IMS Connect configuration file.

**Module:**

- SDRC — HWSSDRCV

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** None. The request message is rejected.

---

**HWSP1505E NEGATIVE SEGMENT LEN; SEG LEN=*1111*, R=*rc*, M=*mc***

**Explanation:** One of the data segments contains an invalid segment length; the length is negative.

In the message text:

- *1111* identifies the length value in the message segment.
- *rc* identifies the XCF return code.
- *mc* identifies the module issuing the message.

**Module:**

- DRCV — HWSSDRCV

**System Action:** This message is issued when a negative segment length is received from the client. The connection is closed.

**System Programmer Response:** None. The request message is rejected.

---

**HWSR0653I PROTECTED CONVERSATION PROCESSING WITH RRS/MVS ENABLED M=*mc***

**Explanation:** An attempt to communicate and restart with RRS is successful.

**Module:** RRSI - HWSRRSI0

**System Action:** The message is issued and IMS Connect continues to run.

---

**HWSR0698W PROTECTED CONVERSATION PROCESSING NOT ENABLED FUNC=*fn*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to communicate with RRS is unsuccessful.

In the message text:

- *fn* identifies the RRS call that was issued.
- *rc* identifies the RRS return code.
- *sc* identifies the RRS reason code.
- *mc* identifies the module issuing the message.

**Module:** RRSI - HWSRRSI0

**System Action:** The message is issued and IMS Connect continues to run.

**System Programmer Response:** Ensure that RRS was brought up correctly. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump, if available.

---

**HWSR0800E FUNCTION WORK ELEMENT PROCESSING FAILURE, FUNC=*func*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function work element (FWE) cannot be processed. The FWE requests work between components and within components. This structure contains the function and parameters that a service requires for processing.

In the message text:

- *func* identifies the function requested.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 85 for an explanation of the service and return code.

*Table 85. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVFUNC | The function requested in the FWE is incorrect. | 4 | This is a processing error. |

**Module:**

- HWSRCDR0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

---

**HWSR0810E STORAGE ALLOCATE FAILED FOR RECORDER DCB; R=*rc*, S=*sc,* M=*mc***

**Explanation:** Storage allocation failed for recorder.

In the message text:

- *rc* identifies the return code.

- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 86 for an explanation of service and return codes.

*Table 86. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETDCB | BPEGETM, the system service used to acquire the CTOKEN. | 4 | An incorrect or unsupported subpool is specified. |
| | | 8 | A zero length is requested. |
| | | 12 | Unable to obtain the requested storage (MVS GETMAIN failed). |

**Module:**
- HWSRCDR0

**System Action:** The system continues to operate, however, no logging of input or output messages will occur.

**System Programmer Response:** More storage is required for the execution of the IMS Connect address space.

**HWSR0820E DCB OPEN FAILED FOR RECORDER DATA SET; R=*rc*, S=*sc*, M=*mc***

**Explanation:** DCB open failed for the recorder DCB.

In the message text:
- *rc* identifies the return code returned from the OPEN request.
- *sc* identifies the service code DCBOPEN.
- *mc* identifies the module issuing the message.

**Module:**
- HWSRCDR0

**System Action:** The message is issued and the recorder data set is set to closed.

**System Programmer Response:** In the system programmer's MVS console, see the error message on the line that directly precedes this error message to determine the appropriate action.

**HWSR0880I RECORDER OPENED; M=*mc***

**Explanation:** A recorder function has been opened successfully.

In the message text:
- *mc* identifies the module issuing the message.

**Module:**
- HWSRCDR0

**System Action:** The recorder data set is now open and logging of input and output message text has begun.

**System Programmer Response:** None.

**HWSR0881I RECORDER ALREADY OPENED; M=*mc***

**Explanation:** A recorder open command was issued; however, the recorder trace is already open.

In the message text:
- *mc* identifies the module issuing the message.

**Module:**
- HWSCHWS0

**System Action:** The recorder open request is ignored.

**HWSR0890I RECORDER CLOSED; M=*mc***

**Explanation:** A recorder function has been closed successfully.

In the message text:
- *mc* identifies the module issuing the message.

**Module:**
- HWSRCDR0

**System Action:** The recorder data set is now closed and logging of input and output message text has ended.

**System Programmer Response:** None.

**HWSR0891I RECORDER ALREADY CLOSED; M=*mc***

**Explanation:** A recorder close command was issued; however, the recorder trace is already closed.

In the message text:
- *mc* identifies the module issuing the message.

**Module:**
- HWSCHWS0

**System Action:** The recorder close request is ignored.

**HWSS0700E FUNCTION WORK ELEMENT PROCESSING FAILURE; FUNC=*fn*, R=*rc*, S=*sc*, M=*mc***

**Explanation:** The function work element (FWE) cannot be processed. The FWE requests work between components and within components. This structure contains the function and parameters that a service requires for processing.

In the message text:

- *fn* identifies the function requested.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 87 for an explanation of the service and return code.

*Table 87. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| INVFUNC | The function requested in the FWE is incorrect. | 4 | This is a processing error. |

**Module:**

- SOCC — HWSSOCC0
- SCVC — HWSSCVC0

**System Action:** This message is issued and, if possible, the requestor of the function is notified. Otherwise, the FWE is freed. In all cases, IMS Connect continues to run.

**System Programmer Response:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0712W UNABLE TO START SCHEDULER CONTROLLER FOR PORT=*portid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the scheduler controller structure, or the scheduler controller thread cannot be scheduled. This controller processes the connection of TCP/IP or Local clients.

In the message text:

- *portid* identifies the TCP/IP or Local port.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 88 for an explanation of service and return codes.

*Table 88. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU) for the scheduler controller. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| NOLOCAL | Open the local port. | 4 | Local portid is not specified. |
| OPENERR | Establish local communication mechanism. | 4 | Initialization of local mechanism fails. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the scheduler controller thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |

Table 88. Service and Return Code Explanation (continued)

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread fails. |

**Module:**

- SOC3 — HWSSOC30

**System Action:** This message is issued and IMS Connect continues to run; however, no communication function is available to the identified TCP/IP of Local port.

**System Programmer Response:** Terminate IMS Connect and ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. Restart IMS Connect. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0714E UNABLE TO START A TCP/IP CLIENT ON PORT=*portid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the conversation controller structure, or the conversation controller thread cannot be scheduled. This controller schedules the communication functions for a TCP/IP client. This error is due to using a region size for IMS Connect that is too small or to a processing or internal system error.

In the message text:

- *portid* identifies the TCP/IP port.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 89 for an explanation of service and return codes.

Table 89. Service and Return Code Explanation

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| DUPESVT | A duplicate client ID (LUNAME) has been specified for this client. | 4 | Two different clients are using the same User ID. |
| GETSVTB | BPECBGET, the system service used to acquire the TCP/IP client table (SVT). This table represents the connected TCP/IP client. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| 8 | Storage is unavailable to satisfy the request. | | |
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU) for the conversation controller. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |

*Table 89. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 8 | Storage is unavailable to satisfy the request. |
| INCLOSE | IMS Connect is in close process. No new connection with IMS Connect is possible. | 12 | This is a processing error. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the scheduler controller thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. This is a system error. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. This is an internal system error. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. This is an internal system error. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. This is an internal system error. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread failed. This is an internal system error. |

**Module:**

- SSC1 — HWSSSC10

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Take one of the following actions:

- If the problem is due to an internal system error and the problem recurs after stopping and restarting IMS Connect, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.
- If the problem is due to a storage shortage, either:

– Allow IMS Connect to continue running with the currently connected TCP/IP clients.
– Terminate and then restart IMS Connect, ensuring that the IMS Connect region size is large enough to accommodate an increase in TCP/IP client connections.

If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0727W TERMINATE FAILED FOR TCP/IP CLIENT=*portid_clientid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to terminate the named client is unsuccessful.

In the message text:

- *portid* identifies the port.
- *clientid* identifies the TCP/IP client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 90 for an explanation of the service and return code.

*Table 90. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| NFNDSVT | The TCP/IP client table (SVT) using the portid and the clientid as the search value cannot be located. This table represents a TCP/IP client connection with IMS Connect. | 4 | This is a processing error. |

**Module:**

- SCCL— HWSSCCL0

**System Action:** This is probably an internal error. Search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0730W COMMAND=*command* FOR PORT=*portid* REJECTED, CLIENT(S) IN PROGRESS; M=*mc***

**Explanation:** An attempt to terminate the port with a command cannot be processed because IMS Connect clients are currently scheduled for this port.

In the message text:

- *command* identifies the datastore.
- *portid* identifies the port.
- *mc* identifies the module issuing the message.

**Module:**

- SSTP— HWSSSTP0

**System Action:** Reenter the command after all active clients for the port have become inactive. Use the `VIEWPORT` command to determine the activity on the port.

**HWSS0742W MESSAGE FAILURE, RECEIVED FROM ORIGIN=*portid_clientid* TO DESTID=*did*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to forward a message received from TCP/IP client *clientid* which is communicating through port *portid* to the required datastore destination.

In the message text:

- *portid* identifies the TCP/IP port.
- *clientid* identifies the TCP/IP client.
- *did* identifies the datastore.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 91 for an explanation of service and return codes.

*Table 91. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| DSCLOSE | All datastores are becoming inactive. This could result from a `CLOSEHWS` command that is shutting down IMS Connect. | 12 | This is a processing error. |

*Table 91. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| DUPECLNT | Duplicate Client ID has been detected. | 8 | Client ID should be unique. |
| GETFWEB | BPECBGET, the system service used to acquire an FWE for queuing of messages. The FWE is used as the queuing structure for a message. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| INVLDSTA | Invalid state has been detected. | 8 | IMS is expecting an ACK, NACK, or deallocate, rather than an input message. |
| INVLDTOK | Invalid server token has been detected. | 8 | Use the correct server token for the conversation iteration. Or, a second client is starting a conversation and is using a duplicate ID while the first client is in a conversation. |
| NFNDCOMP | The component that handles the requested function cannot be found. An IMS Connect component issues an interface call for another component's service and the requested component cannot be located. | 4 | This is a processing error. |
| NFNDDST | The datastore table cannot be found. This table maintains the activity of a datastore. | 4 | This is a processing error. |
| NFNDFUNC | The requested function cannot be found. An IMS Connect component issues an interface call for another component's service and the requested service cannot be located. | 8 | This is a processing error. |
| SHUTDOWN | A CLOSEHWS command has been issued. IMS Connect termination is in process. | 8 | Termination in process. |
| STP/CLSE | Datastore / IMSplex in stop or close process. | 4 | This is a processing error. |

**Module:**

- SRE4 — HWSSRE40

**System Action:** This message is issued and IMS Connect continues to run. The message in progress is released.

**System Programmer Response:** The response can vary depending on the service code.

For service codes DSCLOSE, NFNDDST, and SHUTDOWN, the datastore is no longer active or connected to IMS Connect. Investigate why the datastore was terminated, or if a CLOSEDS command was issued.

For service codes DUPLNT and INVLDTOK, a second client connects to IMS Connect with the same Client ID currently identified to IMS Connect. The client might have disconnected and reconnected with the same Client ID; however, IMS Connect is not aware of the disconnect because the client is in a CONN state waiting for a response from IMS.

For service code INVLDST, the client failed to send an ACK/NAK response when the synch level is defined as CONFIRM; or the client left the conversation early without issuing a DEALLOCATE request to IMS Connect.

**HWSS0746W UNABLE TO NOTIFY ORIGIN=*portid_clientid*OF MESSAGE FAILURE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** IMS Connect is unable to notify the named TCP/IP client about an error that has occurred while processing a request message or a response that IMS Connect has received.

In the message text:

- *portid* identifies the TCP/IP port.
- *clientid* identifies the TCP/IP client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 92 for an explanation of service and return codes.

*Table 92. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETFWEB | BPECBGET, the system service used to acquire an FWE for queuing of messages. The FWE is used as the queuing structure and the message is anchored off the FWE. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETC01K | BPECBGET, the system service used to acquire storage to build the error message. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| NFNDSVT | The TCP/IP client table cannot be found. This table maintains the activity of a connected TCP/IP client. | 4 | This is a processing error. |

**Module:**

- SRE4 — HWSSRE40

**System Action:** This message is issued and IMS Connect continues to run. The request or response message being processed is discarded.

**System Programmer Response:** This error can occur when not enough storage is available to complete the process. If the problem persists, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0761I TCPIP COMMUNICATION WITH CLIENT=*portid_clientid* STOPPED; M=*mc***

**Explanation:** The communication for the named TCP/IP client stops.

In the message text:
- *portid* identifies the TCP/IP port.
- *clientid* identifies the TCP/IP client.
- *mc* identifies the module issuing the message.

**Module:**
- SCCM — HWSSCCM0

**System Action:** This message is issued when a `STOPCLNT` command has taken effect.

**HWSS0762I LOCAL COMMUNICATION WITH CLIENT=*cname* STOPPED; M=*mc***

**Explanation:** Local communication for the named client is stopped.

In the message text:
- *cname* identifies the client.
- *mc* identifies the module issuing the message.

**Module:**
- PCCM - HWSPCCM0

**System Action:** This message is issued when a `STOPCLNT` command takes effect for a client using a Local Option connection. IMS connect continues to run.

**HWSS0763W LOCAL COMMUNICATIONS WITH CLIENT=*cname* CONNECTION FAILURE; R=*rc* S=*sc* M=*mc***

**Explanation:** An IMS Connect client could not connect to IMS Connect. Refer to the service code (*sc*) for more information.

In the message text:
- *cname* identifies the client.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can be codes that either specifically identify the error or are returned by called services that failed the request.
- *mc* identifies the module issuing the message.

**Module:**
- PSCH - HWSPSCH0

**System Action:** This message is issued, and IMS Connect continues to run. The response in progress is released.

**System Programmer Response:** This is probably a storage error. Check that the region size for IMS Connect is large enough to complete the process. If the error reoccurs, search the problem-reporting databases to find a correction to the problem. If a correction does not exist, contact the IMS Support Center. Provide the JCL, SYSLOG, and dump, if available.

**HWSS0770I LISTENING ON PORT=*portid* TERMINATED; M=*mc***

**Explanation:** The communication for the named port has terminated.

In the message text:
- *portid* identifies the TCP/IP port.
- *mc* identifies the module issuing the message.

**Module:**
- SCCH — HWSSSCH0

**System Action:** This message is issued when listening has terminated on a port.

**HWSS0771W LISTENING ON PORT=*portid* FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to start listening on the named port is unsuccessful.

In the message text:
- *portid* identifies the TCP/IP port.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 93 for an explanation of service and return codes.

*Table 93. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| SOCKFAIL | TCP/IP SOCKET function failed. | -1 | Return code from TCP/IP. See message HWSP1415E for TCP/IP failure. |
| COMMAND | Connection not completed. | 4 | A `STOPCLNT`, `STOPPORT`, or `CLOSEHWS` terminated a connection that had not completed. |
| GETMFAIL | Connection not completed. | 4 | IMS Connect internal `GETMAIN` failed. |

*Table 93. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| MAXSOC | Reached maximum socket number. | 4 | IMS Connect will not accept any input until a socket becomes available. |
| ENVHANDL | Connection not completed. | 4 | The HWS configuration file does not specify SSLENVAR. |

**Module:**

- SSCH — HWSSSCH0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Ensure that the named ports are available to IMS Connect for communications. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0775W UNABLE TO START PORT=*portid*; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to open the named port is unsuccessful.

In the message text:

- *portid* identifies the TCP/IP port.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 94 for an explanation of the service and return code.

*Table 94. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| NFNDSCT | The port entry table (SCT) using the portid as the search value cannot be located. This table represents a port while connected with IMS Connect. | 4 | This is a processing error. |

**Module:**

- SOCM — HWSSOCM0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Ensure that the port name in the `OPENPORT` command is correct. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0780I TCPIP COMMUNICATION ON HOSTNAME=*hostname* OPENED; M=*mc***

**Explanation:** The communication facility for TCP/IP is available.

In the message text:

- *hostname* identifies the TCP/IP hostname.
- *mc* identifies the module issuing the message.

**Module:**

- SOC1 — HWSSOC10

**System Action:** This message is issued during IMS Connect startup and whenever communication is established with the TCP/IP communication facility.

**HWSS0781I TCPIP COMMUNICATION FUNCTION FAILED; M=*mc***

**Explanation:** The communication facility for TCP/IP has become inactive.

In the message text:

- *mc* identifies the module issuing the message.

**Module:**

- SOCL — HWSSOCL0

**System Action:** This message is issued when IMS Connect communication with the TCP/IP communication facility is decoupled.

**HWSS0785W OPEN TCPIP COMMUNICATION ON HOSTNAME=*hostname* FAILED; R=*rc*, S=*sc*, M=*mc***

**Explanation:** An attempt to start communication with TCP/IP was unsuccessful.

In the message text:

- *hostname* identifies the TCP/IP hostname.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

**Module:**

- SOC1 — HWSSOC10

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Ensure that the TCP/IP hostname was specified correctly in the HWSCFGxx member or that the MVS TCPIP communication facility is active. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSS0790I LISTENING ON PORT=*portid* STARTED; M=*mc***

**Explanation:** Communication has started for the named TCP/IP port.

In the message text:

- *portid* identifies the TCP/IP port.
- *mc* identifies the module issuing the message.

**Module:**

- SOC2 — HWSSOC20

**System Action:** This message is issued when listening has started on a TCP/IP port.

**HWSSSL00E UNABLE TO *action*, RC=*rc*: *error***

**Explanation:** An error has occurred in SSL.

In the message text:

- *action* identifies the name of the action that failed.
- *rc* identifies the return code.
- *error* identifies the error message specified by gsk_strerror( ).

**Module:**

- HWSSSL00

**System Action:** This message is issued. The SSL socket is closed. IMS Connect continues to run fully. If the message occurs during the SSL environment initialization (when the ports are setup to listen), the port may be closed.

**System Programmer Response:** If this is an initialization error, the SSL input file needs to be examined and fixed according to the error message received.

**HWSSSL00I SSL DEBUG MESSAGE**

**Explanation:** The message corresponds to an SSL debugging message. The message is only enabled if the DEBUG_SSL variable is turned on. The message text pertains to the SSL encryption/transfer process or the SSL initialization process.

**Module:**

- HWSSSL00

**System Action:** None

**System Programmer Response:** None

**HWSX0901E UNABLE TO ALLOCATE ENVIRONMENT SYSTEM TABLE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the environment system table (EST). The EST anchors all of the common service routines, control tables, and control blocks used by the IMS Connect components.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 95 for an explanation of service and return codes.

*Table 95. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| BPEGETM | BPEGETM, the system service used to obtain the storage. | 4 | An incorrect or unsupported subpool is specified. |
| | | 8 | A zero length is requested. |

*Table 95. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 12 | Unable to obtain the requested storage (MVS GETMAIN failed). |

**Module:**
- XTRS — HWSXTRS0

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0902E UNABLE TO ALLOCATE INTERFACE STRUCTURE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the interface execution structure. This structure contains the linkage to the functions supported by each component within IMS Connect.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 96 for an explanation of service and return codes.

*Table 96. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETINTF | BPEGETM, the system service used to obtain the interface control block structure. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:**
- ITBL — HWSITBL0

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0903E UNABLE TO ALLOCATE EXECUTION TABLE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the execution table (E_table). This structure contains the component-related data required for each component to run within the IMS Connect environment.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 97 for an explanation of service and return codes.

*Table 97. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETETBL | BPEGETM, the system service used to obtain the execution table. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |

*Table 97. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:**
- XHD0 — HWSXHD00
- XSH0 — HWSXSH00
- XCM0 — HWSXCM00

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0904E UNABLE TO ALLOCATE COMPONENT INTERFACE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** A component cannot register its interface for the functions it supports. This message follows message HWSX0902E, and indicates that storage cannot be allocated for the component interface structure.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 98 for an explanation of service and return codes.

*Table 98. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| REGINTFR | HWSINTFR is the IMS Connect service used to register the component's interface. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:**
- XHD1 — HWSXHD10
- XSH1 — HWSXSH10
- XCM1 — HWSXCM10

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0905E UNABLE TO ALLOCATE MASTER SERVER; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the master server control structure, or the master server thread cannot be scheduled. This server services all requests directed to the IMS Connect environment that are not directed to a specific component.

In the message text:
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 99 for an explanation of service and return codes.

*Table 99. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETTWUB | BPECBGET, the system service used to acquire the thread workunit (TWU). | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |

*Table 99. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 8 | Storage is unavailable to satisfy the request. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread fails. |

**Module:**

- XTRS — HWSXTRS0

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0907E UNABLE TO START OPEN/CLOSE CONTROLLER; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the open/close controller structure, or the open/close controller thread cannot be scheduled. This controller manages the linkage with the communication feature that IMS Connect uses to communicate with datastores and IMS Connect clients.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 100 for an explanation of service and return codes.

*Table 100. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU) for the open/close controller. | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |

*Table 100. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 8 | Storage is unavailable to satisfy the request. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the open/close controller thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread fails. |

**Module:**

- XHD3 — HWSXHD30
- XSH3 — HWSXSH30

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

---

**HWSX0909E ERROR IN PROCESSING CONFIG MEMBER *name*; M=*mc***
**HWSCFG IS NOT SPECIFIED IN THE STARTUP PARMS.**
**UNABLE TO GET STORAGE; R=*rc*, S=*sc***
**ERROR READING MEMBER; R=*rc*, S=*sc***
**ERROR PARSING MEMBER; R=*rc*, S=*sc***
**INAVLID PARAMETER(S) DETECTED; R=*rc*, S=*sc***
**UNABLE TO ALLOCATE SCT; R=*rc*, S=*sc***
**UNABLE TO ALLOCATE DCT; R=*rc*, S=*sc***
**DUPLICATE PORT ID; R=*rc*, S=*sc***

**Explanation:** During the processing of the CONFIG member specifications, an error is detected, such as incorrect specification or allocation of storage for the execution control structure.

In the message text:

- *name* identifies the name of the CONFIG member.
- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 101 on page 215 for an explanation of service and return codes.

Table 101. Service and Return Code Explanation

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| BPEPARSE | The system service used to parse the parameters. | 4 | The parser definition grammar passed on PADER is not a correct BPEPADEF grammar. |
| | | 8 | The control block storage passed on CBSTG is not large enough to contain the control blocks that needed to be built to contain the parsed input data. |
| | | 12 | The CBSTG address passed to the parsing service is 0. |
| | | 16 | The input data address passed to the parsing service is 0. |
| | | 20 | An internal error occurs in the parsing service. |
| | The system service used to parse the parameters. | 64 | An invalid keyword is detected in the input data. |
| | | 68 | An unknown positional parameter is encountered in the input. |
| | | 72 | A keyword parameter is specified with an equal sign followed by a sublist of values (KEYWORD= xxx,yyy[,...]). A sublist must be specified in parentheses; an equal sign is optional when used with a sublist but required if a keyword has only a single value. |
| | | 76 | The input ended before all of a sublist or keyword has been parsed. |
| | | 80 | A keyword is encountered (KEYWORD(...) or KEYWORD=...) when a value is expected. |
| | | 84 | An input number being parsed is out of the range allowed for its output field length. |
| | | 88 | A parameter value defined as decimal contains nondecimal digits. |
| | | 92 | A parameter value defined as hex contains nonhex digits. |
| | | 96 | A parameter value defined as a key value parameter has an unknown key value. |

Table 101. Service and Return Code Explanation (continued)

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 100 | A keyword parameter appears multiple times and is not defined as being repeatable. |
| | | 104 | A parameter defined with REQUIRED= YES on BPEPADEF is not found in the input data (omitted). |
| | | 252 | The parameter list version generated by BPEPARSE is not supported by the parse service module - macro/module level mismatch. |
| GETSCTB | BPECBGET, the system service used to acquire the server communication table (SCT). | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| GETDCTB | BPECBGET, the system service used to acquire the datastore communication table (DCT). | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |

**Module:**

- XCFG — HWSXCFG0

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the parameters in the CONFIG member are specified correctly and, if it is a storage problem, ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0910E UNABLE TO START COMMAND CONTROLLER; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the command controller control structure, or the command controller thread cannot be scheduled. This server services all requests directed to the IMS Connect environment that are not directed to a specific component.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 102 for an explanation of service and return codes.

Table 102. Service and Return Code Explanation

| Service code | Short explanation | Return code (decimal) | Meaning |
|---|---|---|---|
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU). | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |

*Table 102. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code (decimal) | Meaning |
|---|---|---|---|
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread fails. |

**Module:**

- XCM3 — HWSXCM30

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0911E UNABLE TO START COMMAND VERB CONTROLLER; R=*rc*, S=*sc*, M=*mc***

**Explanation:** Storage cannot be allocated for the command controller control structure, or the command controller thread cannot be scheduled. This server services all requests directed to the IMS Connect environment that are not directed to a specific component.

In the message text:

- *rc* identifies the return code.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 103 for an explanation of service and return codes.

*Table 103. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| GETTWUB | BPECBGET, the system service used to acquire the thread work unit (TWU). | 4 | An incorrect CBTE address is passed to the CB get routine. This is an internal system error. |
| | | 8 | Storage is unavailable to satisfy the request. |
| SCHEDTWU | BPETHDCR, the system service used to schedule the thread. | 4 | An incorrect dispatcher work area is passed to the create thread routine. |

*Table 103. Service and Return Code Explanation (continued)*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| | | 8 | An incorrect TCB index value is passed on the TCBIDX parameter. |
| | | 12 | A zero routine address is passed on the ROUTINE= parameter. |
| | | 16 | An incorrect TCB table entry address is passed into the thread create routine. The BPETHDCR macro determines the TCBT address based on whether the parameter TCBTYPE, TCBIDX, or TCBDWA is specified. Ensure that this parameter is correctly coded. |
| | | 20 | Unable to get storage for a thread control block (THCB) for the thread. |
| | | 24 | Unable to get stack storage for the thread. |
| | | 28 | The initial post of the thread fails. |

**Module:**

- XCM3 — HWSXCM30

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Ensure that the region size in the JCL statement is large enough to accommodate the IMS Connect region. If the error recurs, search the problem-reporting databases to find a correction for the problem. If none exists, contact the IBM Support Center. Provide the JCL, SYSLOG, and dump if available.

**HWSX0912E HWS STARTED IN KEY*ky*— KEY 7 IS REQUIRED**

**Explanation:** IMS Connect is executed in supervisor state and key 7.

In the message text:

- *ky* identifies the key.

**Module:**

- HWS — HWSHWS00

**System Action:** Authorize to the APF the resident library (IMS.RESLIB) in which the IMS Connect modules reside.

**HWSX0912W HWSUINIT RETURNS WARNING CODE; R=*rc*, S=*sc*, M=*mc***

**Explanation:** HWSUINIT, the user initialization exit, issues a warning return code. The meaning of that return code is defined by the user initialization exit itself. IMS Connect is not affected by this warning code and continues its initialization processing.

In the message text:

- *rc* identifies the return code that HWSUINIT sets.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 104 on page 219 for an explanation of the service and return code.

*Table 104. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| UINITFAIL | HWSUINIT returns with a warning code. The warning code is defined by the exit. | 1–7 | The meaning of the HWSUINIT-returned warning code is defined by the user initialization exit itself. IMS Connect is not affected by this warning, and continues its initialization processing. |

**Module:**

- XITF — HWSXITF0

**System Action:** This message is issued and IMS Connect continues its initialization processing.

**System Programmer Response:** Because your installation defines the warning code, you must determine the corrective action to take, and whether to restart IMS Connect.

**HWSX0913E HWSUINIT RETURN CODE >=8, IMS CONNECT SHUTDOWN; R=*rc*, S=*sc*, M=*mc***

**Explanation:** HWSUINIT, the user initialization exit, issues a return code of 8 or higher. IMS Connect terminates initialization processing and shuts down the address space.

In the message text:

- *rc* identifies the return code that HWSUINIT sets.
- *sc* identifies the service code. Service codes can contain either codes that more specifically identify the error, or codes returned by called services that failed the request.
- *mc* identifies the module issuing the message.

See Table 105 for an explanation of the service and return code.

*Table 105. Service and Return Code Explanation*

| Service code | Short explanation | Return code | Meaning |
|---|---|---|---|
| UINTFAIL | HWSUINIT returns with an error code of eight or higher to force IMS Connect to terminate. | 8 or higher | An error return code of 8 or higher notifies IMS Connect that HWSUINIT, the user initialization exit routine, has encountered an error. IMS Connect initialization stops and IMS Connect terminates. |

**Module:**

- XITF — HWSXITF0

**System Action:** This message is issued and IMS Connect terminates.

**System Programmer Response:** Because your installation defines the error code, you must determine the corrective action to take, and whether to restart IMS Connect.

| **HWSX0930I HWSTECL0 NOT INITIALIZED, R15=*nn*, R0=*mm*, M=*xxx***

| **Explanation:** IMS Connect loads the module, HWSTECL0, and calls it for event recording initialization. HWSTECL0 returns with a return and reason code indicating initialization is unsuccessful.

| In the message text:

- | *nn* identifies the return code that HWSTECL0 set.
- | *mm* identifies the reason code associated with any non-zero return codes passed.
- | *xxx* identifies the module issuing the message

| See Table 106 on page 220 for an explanation of the service and return codes.

*Table 106. Service and Return Code Explanation*

| Service code | Register Number | Return code | Meaning |
|---|---|---|---|
| HWSTECL0 | R0 | | Reason code associated with any non-zero return codes passed. |
| | R15 | 0 | Initialization was successful. Check the EICB to see if trace or event recording is active. |
| | | 8 | Initialization was not successful. See reason code for additional information. |

**Module:**
- HWSINIT0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** Check with the provider for HWSTECL0 for possible causes of non-zero return code and corresponding reason codes.

**HWSX0931I HWSTECL0 INIT SUCCESSFUL, R15=*nn*, R0=*nn*, M=*xxx***

**Explanation:** IMS Connect loads the module, HWSTECL0, and calls it for event recording initialization. HWSTECL0 returns with a return and reason code indicating initialization is successful.

See Table 106 for an explanation of the service and return codes.

In the message text:
- *nn* identifies the return code that HWSTECL0 set.
- *mm* identifies the reason code associated with any non-zero return codes passed.
- *xxx* identifies the module issuing the message

**Module:**
- HWSINIT0

**System Action:** This message is issued and IMS Connect continues to run.

**System Programmer Response:** No action is required.

# Chapter 17. IMS Connect Return and Reason Codes

This chapter describes the return and reason codes for the user message exits HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 and contains Diagnosis, Modification, or Tuning Information.

**In this chapter:**

## HWSSMPL0, HWSSMPL1, HWSCSLO0, and HWSCSLO1

The following return and reason codes, in Table 107 and Table 108 on page 222, are sent by HWSSMPL0 and HWSSMPL1 to the client in the RSM fields RSM_RETCOD/RSM_RSNCOD.

- **Return codes**:

*Table 107. Return Codes for HWSSMPL0 and HWSSMPL1*

| Hex Value | Description |
|---|---|
| 04 | Exit request error message sent to client before socket termination |
| 08 | Error detected by IMS Connect |
| 0C | Error returned by IMS OTMA |
| 10 | Error returned by IMS OTMA when an OTMA sense code is returned in the ″Reason Code″ field of the RSM. See the *IMS Open Transaction Manager Access Guide* for your installation’s version of IMS for sense code descriptions. |
| 14 | Currently reserved |
| 18 | SCI error detected, see *IMS Common Service Layer Guide and Reference* for reason codes. |
| 1C | OM error detected, see *IMS Common Service Layer Guide and Reference* for reason codes. |
| 20 | IRM_TIMER value has expired. The reason code value is the value of the IRM_TIMER and the socket is disconnected by IMS Connect. |
| 24 | A default IRM_TIMER value has expired. Either the IRM_TIMER value specified was X'00' or an invalid value. The reason code value is the value specified in the IRM_TIMER field and the socket is disconnected by IMS Connect. |

*Table 107. Return Codes for HWSSMPL0 and HWSSMPL1 (continued)*

| Hex Value | Description |
|---|---|
| 28 | IRM_TIMER value has expired. The reason code value is the value of the IRM_TIMER. The connection is not disconnected. The socket remains connected. |
| 2C | Cancel Timer has completed successfully. |

- **Reason codes**:

*Table 108. Reason Codes for HWWSMPL0 and HWSSMPL1*

| OMUSR Reason Code Passed to Exit | Decimal Value in RSM | Description |
|---|---|---|
| N/A | 4 | Input data exceeds buffer size. |
| N/A | 5 | Negative length value. |
| N/A | 6 | IRM length invalid. |
| N/A | 7 | Total message length invalid. |
| N/A | 8 | OTMA NAK with no sense code or RC. |
| N/A | 9 | Contents of buffer invalid. |
| N/A | 10 | Output data exceeds buffer size. |
| N/A | 11 | Invalid unicode definition. |
| N/A | 12 | Invalid message, no data. |
| N/A | 16 | Do not know who client is. |
| N/A | 20 | OTMA segment length error. |
| N/A | 24 | FIC missing. |
| N/A | 28 | LIC missing. |
| N/A | 32 | Sequence number error. |
| N/A | 34 | Unable to locate context token. |
| N/A | 36 | Protocol error. |
| N/A | 40 | Security violation. |
| N/A | 44 | Message incomplete. |
| N/A | 48 | Incorrect message length. |
| NOSECHDR | 51 | Security failure — no OTMA security header. |
| INVESECHL | 52 | Security failure — no security data in OTMA security header. |
| SECFNOPW | 53 | Security failure — no password in OTMA user data header. |
| SECFNUID | 54 | Security failure — no user ID in OTMA security header. |
| SECFNPUI | 55 | Security failure — no password in OTMA user data and no user ID in OTMA security header. |
| DUPECLNT | 56 | Duplicate Client ID used; the client ID is currently in use. |

*Table 108. Reason Codes for HWWSMPL0 and HWSSMPL1 (continued)*

| OMUSR Reason Code Passed to Exit | Decimal Value in RSM | Description |
|---|---|---|
| INVLDTOK | 57 | Invalid token is being used — internal error. |
| INVLDSTA | 58 | Invalid client status — internal error. |
| CANTIMER | 59 | Cancel Timer completed successfully. |
| NFNDCOMP | 70 | Component not found. |
| NFNDFUNC | 71 | Function not found. |
| NFNDDST | 72 | Datastore not found. |
| DSCLOSE | 73 | IMS Connect in shutdown. |
| STP/CLSE | 74 | Datastore/IMSplex in stop or close process. |
| DSCERR | 75 | Datastore communication error. |
| STOPCMD | 76 | Datastore/IMSplex was stopped by command. |
| COMMERR | 77 | Datastore/IMSplex communication error to pending client. |
| SECFAIL | 78 | Security failure. RACF call failed, IMS Connect call failed. See IMS Connect error message on system console. |
| PROTOERR | 79 | IMS Connect protocol error. See IMS Connect error message on system console. |
| NOTACTV | 80 | The IMSplex connection is not active. The `STOPIP` command was issued or the SCI address space is not active. |
| INVLDCM1 | 93 | Invalid commit mode of 1 specified on the RESUME TPIPE request. |
| REQUEST | 94 | Request. |
| CONVER | 95 | Conversation. |
| REQ_CON | 96 | Request and conversation. |
| DEAL_CTD | 97 | Deallocate confirmed. |
| DEAL_ABT | 98 | Deallocate abort. |
|  | 99 | Default reason code. |

## HWSIMSO0 and HWSIMSO1

The following return and reason codes, in Table 109 and Table 110 on page 224, are sent by HWSIMSO0 and HWSIMSO1 to the client in the RSM fields RSM_RETCOD/RSM_RSM_RSNCOD.

- **Return codes**:

*Table 109. Return Codes for HWSIMSO0 and HWSIMSO1*

| Hex Value | Description |
|---|---|
| 04 | Exit request error message sent to client before socket termination |

*Table 109. Return Codes for HWSIMSO0 and HWSIMSO1 (continued)*

| Hex Value | Description |
|---|---|
| 08 | Error detected by IMS Connect |
| 0C | Error returned by IMS/OTMA |
| 10 | Error returned by IMS OTMA when an OTMA sense code is returned in the ″Reason Code″ field of the RSM. See the *IMS Open Transaction Manager Access Guide* for your installation’s version of IMS for sense code descriptions. |
| 14 | Currently reserved. |
| 18 | SCI error detected. See *IMS Common Service Layer Guide and Reference* for REASON codes. |
| 1C | OM error detected. See *IMS Common Service Layer Guide and Reference* for REASON codes. |
| 20 | The IRM_TIMER value has expired. The reason code value is the value of the IRM_TIMER and the socket is disconnected by IMS Connect. |
| 24 | A default IRM_TIMER value has expired. Either the IRM_TIMER value specified was X'00' or an invalid value. The reason code value is the value of the IRM_TIMER and the socket is disconnected by IMS Connect. |
| 28 | IRM_TIMER value has expired. The reason code value is the value of the IRM_TIMER. The connection is not disconnected. The socket remains connected. |
| 2C | Cancel Timer has completed successfully. |

- **Reason codes**:

*Table 110. Reason Codes for HWSIMSO0 and HWSIMSO1*

| OMUSR Reason Code Passed to Exit | Decimal Value in RSM | Description |
|---|---|---|
| N/A | 4 | Input data exceeds buffer size. |
| N/A | 5 | Negative length value. |
| N/A | 6 | IRM length invalid. |
| N/A | 7 | Total message length invalid. |
| N/A | 8 | OTMA NAK with no sense code or RC. |
| N/A | 9 | Contents of buffer invalid. |
| N/A | 10 | Output data exceeds buffer size. |
| N/A | 11 | Invalid unicode definition. |
| N/A | 12 | Invalid message, no data. |
| N/A | 16 | Do not know who client is. |
| N/A | 20 | OTMA segment length error. |
| N/A | 24 | FIC missing. |

*Table 110. Reason Codes for HWSIMSO0 and HWSIMSO1 (continued)*

| OMUSR Reason Code Passed to Exit | Decimal Value in RSM | Description |
|---|---|---|
| N/A | 28 | LIC missing. |
| N/A | 32 | Sequence number error. |
| N/A | 34 | Unable to locate context token. |
| N/A | 36 | Protocol error. |
| N/A | 40 | Security violation. |
| N/A | 44 | Message incomplete. |
| N/A | 48 | Incorrect message length. |
| NOSECHDR | 51 | Security failure — no OTMA security header. |
| INVESECHL | 52 | Security failure — no security data in OTMA security header. |
| SECFNOPW | 53 | Security failure — no password in OTMA user data header. |
| SECFNUID | 54 | Security failure — no user ID in OTMA security header. |
| SECFNPUI | 55 | Security failure — no password in OTMA user data and no user ID in OTMA security header. |
| DUPECLNT | 56 | Duplicate Client ID used; the client ID is currently in use. |
| INVLDTOK | 57 | Invalid token is being used — internal error. |
| INVLDSTA | 58 | Invalid client status — internal error. |
| CANTIMER | 59 | Cancel Timer completed successfully. |
| NFNDCOMP | 60 | Component not found. |
| NFNDFUNC | 61 | Function not found. |
| NFNDDST | 62 | Datastore not found. |
| DSCLOSE | 63 | IMS Connect in shutdown. |
| STP/CLSE | 64 | Datastore/IMSplex in stop or close process. |
| DSCERR | 65 | Datastore communication error. |
| STOPCMD | 66 | Datastore/IMSplex was stopped by command. |
| COMMERR | 67 | Datastore/IMSplex communication error to pending client. |
| SECFAIL | 68 | Security failure. RACF call failed, IMS Connect call failed. See IMS Connect error message on system console. |
| PROTOERR | 69 | IMS Connect protocol error. See IMS Connect error message on system console. |
| INVLDCM1 | 93 | Invalid commit mode of 1 specified on the RESUME TPIPE request. |

*Table 110. Reason Codes for HWSIMSO0 and HWSIMSO1 (continued)*

| OMUSR Reason Code Passed to Exit | Decimal Value in RSM | Description |
|---|---|---|
| REQUEST | 94 | REQUEST |
| CONVER | 95 | Conversation |
| REQ_CON | 96 | Request and conversation |
| DEAL_CTD | 97 | Deallocate confirmed |
| DEAL_ABT | 98 | Deallocate abort |
| | 99 | Default reason code |

## IMS Connector for Java

The following return and reason codes, in Table 111 and Table 112 on page 227, are sent by IMS Connect to IMS Connector for Java in the OTMA User fields OMUSR_RETCODE and OMUSR_RESCODE.

- **Return codes**:

*Table 111. Return Codes for OTMA*

| OMUSR_RETCODE received by IMS Connector for Java (Hex value) | Description |
|---|---|
| 04 | Exit request error message sent to client before socket termination |
| 08 | Error detected by IMS Connect |
| 0C | Error returned by IMS/OTMA |
| 10 | Not valid for HWSJAVA0 (OTMA RETURN code) |
| 14 | Reserved |
| 18 | Not valid for HWSJAVA0 (SCI RETURN code) |
| 1C | Not valid for HWSJAVA0 (OM RETURN code) |
| 20 | The IRM_TIMER has expired. The reason code value is the value of the IRM_TIMER and the socket is disconnected by IMS Connect. |
| 24 | A default IRM_TIMER value has expired. Either the IRM_TIMER value specified was X'00' or an invalid value. The reason code is the value of the IRM_TIMER and the socket is disconnected by IMS Connect. |
| 28 | IRM_TIMER value has expired. The reason code value is the value of the IRM_TIMER. The connection is not disconnected. The socket remains connected. |
| 2C | Cancel Timer has completed successfully. |

- **Reason codes**:

*Table 112. Reason Codes for OTMA*

| OMUSR_RESCODE received by IMS Connector for Java | Description |
|---|---|
| NOSECHDR | Security failure; no OTMA security header. |
| INVSECHL | Security failure; no security data in the OTMA security header. |
| SECFNOPW | Security failure; no password in the OTMA user data header. |
| SECFNUID | Security failure; no user ID in the OTMA user security header. |
| SECFNPUI | Security failure; no password in the OTMA user data header, and no user ID in the OTMA user security header. |
| DUPECLNT | Duplicate client ID was used; the client ID is currently in use. |
| INVLDTOK | Invalid token is being used; internal error. |
| INVLDSTA | Invalid client status; internal error. |
| CANTIMER | Cancel Timer completed successfully. |
| NFNDCOMP | Component not found. |
| NFNDFUNC | Function not found. |
| NFNDDST | Datastore not found. |
| DSCLOSE | IMS Connect in shutdown. |
| STP/CLSE | Datastore/IMSplex in stop or close process. |
| DSCERR | Datastore communication error. |
| STOPCMD | Datastore/IMSplex was stopped by a command. |
| COMMERR | Datastore/IMSplex communication error to pending client. |
| SECFAIL | Security failure; a RACF call failed; an IMS Connect call failed. See the IMS Connect error message on the system console. |
| PROTOERR | An IMS Connect protocol error occurred. See the IMS Connect error message on the system console. |
| INVLDCM1 | An invalid command mode of 1 was specified on the RESUME TPIPE request. |
| REQUEST | Request. |
| CONVER | Conversation. |
| REQ_CON | Request and conversation. |
| DEAL_CTD | Deallocate confirmed. |
| DEAL_ABT | Deallocate abort. |
|  | Default reason code. |
| NFNDUOR | Unit of recovery not found. |

## Extended Local Return and Reason Codes

The following return and reason codes, in Table 113 and Table 114, are sent by IMS Connect to IMS Connector for Java, and are passed back to the client application in the exception.

- **Return codes**:

*Table 113. Extended Local Return Codes*

| Hex Value | Description |
|---|---|
| 04 | Exit request error message sent to client before socket termination |
| 08 | Error detected by IMS Connect |
| C | Error returned by IMS/OTMA |
| 20 | The IRM_TIMER has expired. The reason code value is the value of the IRM_TIMER. |
| 24 | A default IRM_TIMER value has expired. Either the IRM_TIMER value specified was X'00' or an invalid value. The reason code value is the value of the IRM_TIMER. |

- **Reason codes**:

*Table 114. Extended Local Reason Codes*

| Reason code | Description |
|---|---|
| BPESVCER | SVC was incorrectly set up. |
| CLNTSTOP | Client is stopped. This might occur if the client is stopped after an IMS timeout when MPP is unavailable to process the transaction. |
| CTXSWCHF | RRS context switch failed. |
| ESTAEERR | ESTATE setup error was detected. |
| HWSFAIL | IMS Connect failed during a call. |
| HWSNOACT | IMS Connect is currently inactive. |
| HWSSHUTP | IMS Connect is shutting down. |
| INACTIVE | The Local port is not currently active. |
| INTFABND | The client interface to IMS Connect abnormally ended during the call. |
| INVLDCID | An invalid client ID was specified. |
| NAMTKNER | An invalid IMS Connect name was specified. |
| NFNDSVT | The connection token control block was not found. This might indicate that the last 4 bytes of the connection token have been corrupted. |
| RACFFAIL | The SAF check against the client failed. The client address space is not authorized to access HWS.ICON_NAME in the facility class. |
| SBFLBAD | An invalid length for the send buffer was detected. |

## IMS Connect Post Codes

IMS Connect post codes in Table 115 identify the IMS Connect Module that issued the post or the meaning of the post code. For each post code, the first byte is blank, and the following three bytes are alphabetic data. For example, in the post code CMD, the code is 'bCMD' where 'b' is blank.

**Post Codes**:

*Table 115. IMS Connect Post Codes*

| Value (decimal) | Module OR Meaning |
|---|---|
| CMD | HWSCMDC0 |
| CXQ | HWXCXQH0 |
| CXR | HWSCXRP0 |
| DCV | HWSDREC0 |
| DOC | HWSDOCC0 |
| DOP | HWSDOPN0 |
| DO3 | HWSDOC30 |
| DRE | HWSDREC0 |
| DSC | HWSDSCH0 |
| DSE | HWSDSCE0 |
| DSL | HWSDSCL0 |
| DST | HWSDSTM0 |
| DS2 | HWSDSC20 |
| DSE | HWSDSC30 |
| DS5 | HWSDSC50 |
| DXC | HWSDDXCN |
| DXM | HWSDXMT0 |
| EQC | HWSEQCL0 |
| EQS | HWSEQS00 |
| ETR | HWSETRM0 |
| OCL | HWSSOCL0 |
| OCM | HWSSOCM0 |
| PCD | HWSPSVT0 |
| PCI | HWSPCINF |
| PCR | HWSPSVT0 |
| PCS | HWSPSVT0 |
| PCV | HWSPCVC0 |
| PST | Good post value |
| RCD | HWSRCDR0 |
| REC | HWSSREC0 |
| SCV | HWSSCVC0 |
| SOC | HWSSOCL0 |
| SOL | HWSSOCL0 |

*Table 115. IMS Connect Post Codes (continued)*

| **Value (decimal)** | **Module OR Meaning** |
|---|---|
| SOP | HWSSOPN0 |
| SST | HWSSSTP0 |
| STP | HWSSSTP0 |
| STR | HWSSTRM0 |
| SVT | HWSSVTM0 |
| SXT | HWSSXTE0 |
| VTD | NWSSVTD0 |
| XMT | HWSSXMT0 |
| $TI | BPE timer call post |

# Part 4. Appendixes

# Appendix A. Recorder Log Record Mapping

This appendix illustrates the recorder log record mapping and contains Diagnosis, Modification, or Tuning Information.

```
**********************************************************************
*              COMMON SECTION      32_BYTES
**********************************************************************
USTAT_NEXT     DS  F               NEXT POINTER
USTAT_EYE      DS  CL4'ITOC'       EYECATCHER
USTAT_CALLID   DS  CL2             CALLER ID
*                                  CHARS "RC" = RECEIVE
*                                  CHARS "SN" = SEND
*                                  CHARS "ER" = READ ERROR
USTAT_SMFHDR   DS  0C              SMF HEADER
SMFITOCLEN     DS  CL2             SMF LENGTH
SMFITOCSEG     DS  CL2             INTERNAL WORK
SMFITOCFLG     DS  X               INTERNAL FLAG
SMFITOCRTY     DS  X               RECORD TYPE
SMFITOCTME     DS  CL4             TIME OF TRACE
SMFITOCDTE     DS  CL4             SEQUENCE NUMBER
SMFITOCSID     DS  CL4             RESERVED
               DS  CL4             RESERVED
**********************************************************************
*              UOW PROGRESSION TIME STAMP SECTION
**********************************************************************
SMFITOCCID     DS  CL8             CLIENT NAME
USTAT_TSMREC   DS  D               TIME HWSW MSG RECEIVED
USTAT_TSMNQ    DS  D               TIME HWSW MSG ENQUEUED
USTAT_TDMDQ    DS  D               TIME 1ST DST MSG DEQUEUED
USTAT_TCLRDQ   DS  D               TIME DST CLR DENQUEUED
USTAT_TERROR   DS  D               TIME ERROR OCCURRED
USTAT_NMSGX    DS  H               NUMBER OF MSGS TRANSMITTED
USTAT_NMSGR    DS  H               NUMBER OF MSGS RECEIVED
               DS  CL8             RESERVED
USTAT_SMFITOCL EQU *-USTAT_SMFHDR  LENGTH OF SMF
**********************************************************************
*              INPUT MSG
**********************************************************************
USTAT_IN_EYE   DS  CL4'*IPB'       EYECATCHER
*                                  *IPB IS THE INPUT TO THE EXIT
*                                       FOR EITHER RECEIVE OR SEND
*                                       USTAT_CALLID = RC - RECEIVE
*                                                      SN - SEND
*                                                      ER - READ ERROR
*  for ITOCRC and *IPB
*     (USTAT_CALLID = "RC")
*
*     THE LOGGED DATA STARTING AT OFFSET X'60' IS AS FOLLOWS:
*      (INPUT TO EXIT FROM CLIENT)
*          llll
*          IRM
*          llzzTRANCODEDATA
*          X'00040000'
**********************************************************************
*  for ITOCRC and *IPB
*     (USTAT_CALLID = "SN")
*
*     THE LOGGED DATA STARTING AT OFFSET X'60' IS AS FOLLOWS:
*      (INPUT TO EXIT FROM IMS APPLICATION)
*       OTMA CONTROL HEADER followed by
*       OTMA STATE DATA HEADER (if present) followed by
*       OTMA SECURITY DATA HEADER (if present) followed by
*       OTMA USER DATA HEADER (if present) followed by
*       DATA TO BE SENT
*          llzzTRANCODEDATA
*
USTAT_MSG_I    DS  CL202           MSG

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

*              OUTPUT MSG

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

USTAT_OUT_EYE  DS  CL4'*OPB'       EYECATCHER

*                                  *OPB IS THE OUTPUT FROM THE EXIT
*                                       FOR EITHER RECEIVE OR SEND
*                                       USTAT_CALLID = RC - RECEIVE
*                                                      SN - SEND
*                                                      ER - READ ERROR
*  for ITOCRC and *OPB
*     (USTAT_CALLID = "RC")
*
*     THE LOGGED DATA STARTING AT OFFSET X'300' IS AS FOLLOWS:
*      (OUTPUT FROM USER EXIT OF CLIENT INPUT DATA)
*       OTMA CONTROL HEADER followed by
*       OTMA STATE DATA HEADER (if present) followed by
*       OTMA SECURITY DATA HEADER (if present) followed by
*       OTMA USER DATA HEADER (if present) followed by
*       APPLICATION DATA TO BE SENT
*          llzzTRANCODEDATA
**********************************************************************
*  FOR ITOCRC AND *OPB
*     (USTAT_CALLID = "SN")
*
*     THE LOGGED DATA STARTING AT OFFSET X'300' IS AS FOLLOWS:
*      (OUTPUT FROM USER EXIT OF APPLICATION OUTPUT DATA)
*       OTMA CONTROL HEADER followed by
*       OTMA STATE DATA HEADER (if present) followed by
*       OTMA SECURITY DATA HEADER (if present) followed by
*       OTMA USER DATA HEADER (if present) followed by
*       DATA TO BE SENT (ONE OF THE FOLLOWING STRUCTURES
*          'RMM'LLZZDATA.................'CSM'
*          LLZZDATA.................'CSM'
*          'RSM'
*
*           RMM is the *REQMOD* structure
*           CSM is the *CSMOKY* structure
*           RSM is the *REQSTS* structure
*
USTAT_END_EYE  DS  CL4'*END'       EYECATCHER
```

# Appendix B. OTMA Headers

The following tables, (Table 116, Table 117 on page 240, Table 121 on page 245, Table 122 on page 246, Table 123 on page 246, Table 124 on page 247, and Table 125 on page 247) lists the fields of the OTMA headers, and the requirements for each field as they should be set or integrated by the user exits. The notes for each table are defined at the end of this appendix under Notes, which follows Table 125 on page 247.

*Table 116. HWS0MCTL DSECT - OTMA Control Header (Control Data Common Section for All Messages)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMCTLALV | 1 | 0 | | ARCHITECTURE LEVEL<br><br>Set to X'01' arch. level 1.<br>Set for all messages. | 1 |
| OMCTLMGT | 1 | 1 | OMCTLDTA X'80' | MESSAGE TYPE=Data<br><br>Set for conversational transactions but not on first input.<br><br>If EXPREA FLAG1 is set to EXPREA_ CONVERS then set OMCTLMGT to OMCTLDTA.<br><br>EXPREA FLAG1 is not set to EXPREA_ CONVERS on the first input for conversation. | 1 |
| | | | OMCTLTXN X'40' | MESSAGE TYPE=Transaction<br><br>Set for first transaction input. That is, first input for conversation or nonconversation, EXPREA_FLAG1 is not set to EXPREA_CONVERS. | 1 |

*Table 116. HWS0MCTL DSECT - OTMA Control Header (Control Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMCTLRSP X’20’ | MESSAGE TYPE=Response<br><br>Set for:<br>• ACK response to msg sent to client<br>• NAK response to msg sent to client<br><br>Required for:<br>• Commit Mode 0 (synch level=CONFIRM)<br>• Commit Mode 1 (synch level=CONFIRM) | 1 |
| | | | OMCTLCMD X’10’ | MESSAGE TYPE-Command<br><br>Set for - RESUME TPIPE | 1 |
| | | | OMCTLCMT X’08’ | MESSAGE TYPE = Commit Confirmation<br><br>Set for SEND ONLY or DEALLOCATE. SEND ONLY or DEALLOCATE is indicated in IRM from client. | 1 |
| OMCTLRSI | 1 | 2 | OMCTLACK X’80’ | RESPONSE INDICATOR<br><br>RESPONSE = ACK<br><br>Set for ACK. ACK is indicated in IRM. | |
| | | | OMCTLNAK X’40’ | RESPONSE = NAK<br><br>Set for NAK. NAK is indicated in IRM. | 1 |
| | | | OMCTLRRQ X’20’ | RESPONSE = Response requested<br><br>If set, then conversational trans and the IMSEA_RSNCODE must be set to 96 (X’60’) to signal client application that conversation continues. | 1 |

*Table 116. HWS0MCTL DSECT - OTMA Control Header (Control Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMCTLERQ X'10' | RESPONSE=Extended response requested<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMCTLCCI | 1 | 3 | OMCTLCTD X'80' | COMMIT CONFIRMATION INDICATOR<br><br>Confirm=Committed<br><br>If set, then the IMS application has terminated the conversation, and the IMSEA_RSNCODE must be set to 97 (X'61') to signal client application that the IMS application terminated successfully. | |
| | | | OMCTLABT X'40' | Confirm=Aborted NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMCTLYP | 1 | 4 | OMCTLBID X'04' | COMMAND TYPE<br><br>COMMAND=Client Bid<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMCTLAVL X'08' | COMMAND=Server Available<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMCLTRSN X'0C' | Command=Resynch<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | X'10' | Reserved for future use. Neither tested nor set by exit. | 4 |
| | | | OMCTLSPA X'14' | Command=Suspend I/P for all tpipes.<br><br>Neither tested nor set by exit. | 4 |
| | | | OMCTLRSA X'18' | Command=Resume I/P for all tpipes.<br><br>Neither tested nor set by exit. | 4 |

*Table 116. HWS0MCTL DSECT - OTMA Control Header (Control Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMCTLSPN X'1C' | Command=Suspend I/P for named tpipe.<br><br>Neither tested nor set by exit. | 4 |
| | | | OMCTLRSM X'20' | Command=Resume I/P for named tpipe.<br><br>Neither tested nor set by exit. | 4 |
| | | | OMCTLRTP X'24' | Command=Resume O/P for named tpipe without options.<br><br>Set for RESUME TPIPE without options. | 1 |
| | | | OMCTLRID X'28' | Command=Resume single tpipe with options.<br><br>Set for RESUME TPIPE with options. | 1 |
| OMCTLPFG | 1 | 5 | OMCTLLPG X'80' | PROCESSING FLAG<br><br>Load Program<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMCTLSYP X'40' | Synchronized tpipe.<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMCTLASY X'20' | Asynchronous/ unsolicited queued messages.<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMCTLERR X'10' | There is an error message with the NAK. NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMCTLQUE X'08' | Asynchronous message is in IMS Hold Queue.<br><br>If set, set CSM_FLG1 to CSM_AMSG if sending CSM, orset RSMFLG1 to RSM_AMSG if sending RSM. | 4 |

*Table 116. HWS0MCTL DSECT - OTMA Control Header (Control Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMCTLOMEX'01' | SCI not present error message. | |
| OMCTLTNM | 8 | 6 | | Tpipe name. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMCTLCHN | 1 | E | OMCTLFIC X’80’ | CHAIN STATE FLAG<br><br>First in chain. Set for first message segment in chain. | 1 |
| | | | OMCTLMIC X’40’ | Middle in chain. Set for not first and/or not last message segment in chain. | 1 |
| | | | OMCTLLIC X’20’ | Last in chain. Set for last message segment in chain. | 1 |
| | | | OMCTLCAN X’10’ | Cancel this message. Neither tested nor set by exit. | 4 |
| OMCTLPFL | 1 | F | OMCTLSTD X’80’ | PREFIX FLAG<br><br>State Data is present. Set if State Data Header present in OTMA Headers being built. | 1 |
| | | | OMCTLSEC X’40’ | Security data is present. Set if Security Data Header present in OTMA Headers being built. | 1 |
| | | | OMCTLUSR X’20’ | User data is present. Set if User Data Header present in OTMA Headers being built. | 1 |
| | | | OMCTLAPP X’10’ | Application data is present. Set if Application Data Header present in OTMA Headers being built. | 1 |
| OMCTLSSN | 4 | 10 | | SEND SEQUENCE NUMBER. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMCTLSNS | 4 | 14 | | SENSE CODE. See OMCTLSNC and OMCTLRSC, which follow. | |

Table 116. HWS0MCTL DSECT - OTMA Control Header (Control Data Common Section for All Messages) (continued)

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | ORG OMCTLSNS | | |
| OMCTLSNC | 2 | 14 | | SENSE CODE<br><br>If nonzero value, then build a NAK RSM to send to the client application, pass the sense code in the RSM as the reason code, and set the return code to X'0C'. | 1 |
| OMCTLRSC | 2 | 16 | | REASON CODE<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMCTLRSQ | 4 | 18 | | RECOVERABLE MESSAGE SEQUENCE NUMBER<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMCTLSEQ | 2 | 1C | | SEGMENT SEQUENCE NUMBER<br><br>Set to 1 in first OTMA Control Header and count maintained in user work area.<br><br>Increment by 1 for each subsequent OTMA Control Header within a single message being sent to IMS. | 1 |
| | 1 | 1E | | RESERVED. | 3 |
| | 1 | 1F | | RESERVED. | 3 |

Table 117. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for Server Available and Client Bid Command Format)

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDRLEN | 2 | 0 | | STATE DATA LENGTH<br>Set to State Data length. | 1 |
| OMHDRORG | | 2 | State data for 'Server Available' and 'Client Bid' commands. | | |

*Table 117. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for Server Available and Client Bid Command Format) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDRONM | 16 | 2 | | MEMBER NAME OF ORIGINATING SERVER. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDROMT | 8 | 12 | | MEMBER TOKEN OF COMMAND ORIGINATOR. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRDMT | 8 | 1A | | MEMBER TOKEN OF COMMAND DESTINATION. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRUEN | 8 | 22 | | UNRESOLVED DESTINATION EXIT NAME. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRMBS | 2 | 2A | | XCF TRANSMISSION MAX BLOCKSIZE. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRRQE | 1 | 2C | OMHDRCMQ X’80’ | CREATE HOLD MESSAGE QUEUE. NEITHER TESTED OR SET BY EXIT. | 4 |
| | 1 | 2D | | RESERVED. NEITHER TESTED OR SET BY EXIT. | 3 |
| OMHDRUAV | 4 | 2E | | SAF USER ID TABLE AGING VALUE. NEITHER TESTED OR SET BY EXIT. | 4 |

*Table 117. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for Server Available and Client Bid Command Format) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDRHTS | 4 | 32 | | MESSAGE RE-ASSEMBLY HASH TABLE SIZE. NEITHER TESTED OR SET BY EXIT. | 4 |
| | | 2 | ORG OMHDRORG | | |

*Table 118. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for resume output for single named TPIPE for the asynchronous option of NO OPTION selection)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDCRSM_COUNT | 2 | 2 | | NUMBER OF TPIPES IN THE ARRAY. Set to number of tpipes to retrieve output from a RESUME TPIPE request. Only valid value is one (1). | 1 |
| OMHDCRSM_TPIPEN | n | 4 | | TPIPE ARRAY. Set to the name of the tpipe to retrieve output from a RESUME TPIPE request. Only one name is valid. | 1 |
| | | 2 | ORG OMHDRORG | | |

*Table 119. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for resume output for single named TPIPE for options of NOAUTO, SINGLE, SINGLE with WAIT, and AUTO)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDRRHQ | 1 | 2 | OPTIONS | | |
| | | | OMHDRRHQ_NOAUTO X'80' | Exhaust all current messages in IMS queue, then hold any new message in IMS queue, until next RESUME TPIPE. Active option in HWSIMSO0, HWSIMSO1, HWSSMPL0, HWSSMPL1 | 1 |

*Table 119. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for resume output for single named TPIPE for options of NOAUTO, SINGLE, SINGLE with WAIT, and AUTO) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMHDRRHQ_AUTO X'40' | Exhaust all current messages in IMS queue, and wait for next message. This option requires that IRM_TIMER be set to X'E9' on ACK to IMS Connect from client, to wait for next output from IMS Connect. | 1 |
| | | | OMHDRRQ_ONE X'20 | Send only one message, and require a new RESUME TPIPE Receive sequence to get any subsequent messages. | 1 |
| | 1 | 3 | | Reserved for IMS Connect. | |
| OMHDCRHQ_TPIPEN | 8 | 4 | | Tpipe name. Set to the name of the tpipe to retrieve output from a RESUME TPIPE request. Only one name is valid. | 1 |
| | | 2 | ORG OMHDRORG | | |

*Table 120. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for transaction messages)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDRIST | 1 | 2 | OMHDRCNV X'80' | IMS STATE FLAG Conversational State. | 2 |
| | | | OMHDRRSP X'40' | Response Mode NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMHDRMHQ X'20' | Message from Hold Queue. | |
| OMHDRSYN | 1 | 3 | X'80' | SYNCHRONIZATION FLAG. Reserved. | 3 |
| | | | OMHDRCM0 X'40' | Commit Mode 0<br>Set if default for exit or the IRM requests Commit Mode 0, field IRM_F2 is set to ″IRM_CMODE0.″ | 1 |

*Table 120. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for transaction messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMHDRCM1 X'20' | Commit Mode 1<br>Set if default for exit or the IRM requests Commit Mode 1, field IRM_F2 is set to ″IRM_CMODE1.″ | 1 |
| | | | OMHDRNTX X'10' | Notify of transfer.<br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRSLV | 1 | 4 | OMHDRSL0 X'00' | SYNCH LEVEL<br>Synchlevel=0 (None)<br>Set if default for exit or the IRM requests synch level none, field IRM_F3 is not set to ″IRM_CONFIRM.″<br>Synchlevel=0 is only valid for Commit Mode 1. | 1 |
| | | | OMHDRSL1 X'01' | Synchlevel=1 (Confirm)<br>Set if default for exit or the IRM requests synch level confirm, field IRM_F3 is set to ″IRM_CONFIRM.″<br>Synchlevel=1 is valid for Commit Modes 0 and 1. | 1 |
| | | | OMHDRSL2 X'02' | Synchlevel=2 (Syncpt)<br>NEITHER TESTED NOR SET BY EXIT. | |
| OMHDRCFL | 1 | 5 | OMHDRSOM X'80' | Set for SENDONLY.<br>SENDONLY is indicated in the IRM from the client. | 1 |
| OMHDRMAP | 8 | 6 | | MAP NAME<br>If client application requested that the MODname be returned, then the exit must build an RRM in front of the data being returned to the client. The MODname (OMHDRMAP) would be moved to the RRM to field RRM_MODNAME by the exit. | 1 |

*Table 120. HWS0MHDR DSECT - OTMA State Data Header (State Data Common Section for transaction messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMHDRTOK | 16 | E | | SERVER TOKEN. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRCOR | 16 | 1E | | CORRELATOR. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRCID | 16 | 2E | | CONTEXT ID. NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMHDRLTM | 8 | 3E | | OVERRIDE LTERM NAME. Set from IRM LTERM field ″IRM_LTERM.″ | 1 |
| OMHDRLIU | 2 | 46 | | LENGTH OF IMS HEADER USER DATA. Set to the user header data length that follows. The length of this field is not included in the total length. | 5 |
| OMHDRIUD | n | 48 | | IMS HEADER USER DATA. Variable length, set by the user. | 5 |

*Table 121. HWS0MSEC DSECT - OTMA Security Data Header (Security Data Common Section for All Messages)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMSECLEN | 2 | 0 | | SECURITY DATA LENGTH | |
| OMSECFLG | 1 | 2 | OMSECNON C’N’ | SECURITY FLAG<br><br>No RACF checking.<br><br>Set to ’N’ if no OTMA RACF calls are to be made. | 1 |
| | | | OMSECCHK C’C’ | Check for Tran and Cmd.<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMSECFUL C’F’ | Check for Tran, Cmd, and MPR<br><br>Set to ’F’ is OTMA is to issue RACF call. | 1 |

Table 121. HWS0MSEC DSECT - OTMA Security Data Header (Security Data Common Section for All Messages) (continued)

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMSECFLN | 1 | 3 | | LENGTH OF FOLLOWING FIELDS<br><br>Set to length of USERID and GROUPID section.<br>• Set to X'0A' if only USERID.<br>• Set to X'14' if USERID and GROUPID.<br>• Set to X'00' if neither USERID or GROUPID present. | 1 |

Table 122. HWSECUDS DSECT - OTMA USERID Definition (Security Data USERID Section for All Messages)

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMSECULN | 1 | 0 | | LENGTH OF USERID FIELDS<br><br>Set to length of USERID fields. The length includes this field. Set to X'09' if USERID present. | 1 |
| OMSECUTY | 1 | 1 | OMSECUXX X'02' | FIELD TYPE USERID type.<br>Set to X'02' to identify USERID present. | 1 |
| OMSECUID | 8 | 2 | | USERID<br><br>Set to USERID from IRM field IRM_RACF_USERID. | 1 |

Table 123. HWSECGDS DSECT - OTMA GROUPID Definition (Security Data GROUPID Section for All Messages)

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMSECGLN | 1 | 0 | | LENGTH OF GROUPID FIELDS<br><br>Set to length of GROUPID fields. The length includes this field. Set to X'09' if GROUPID present. | 1 |

*Table 123. HWSECGDS DSECT - OTMA GROUPID Definition (Security Data GROUPID Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMSECGTY | 1 | 1 | OMSECGXX X'02' | FIELD TYPE GROUPID type. Set to X'03' to identify GROUPID present. | 1 |
| OMSECGRP | 8 | 2 | | RACF GROUPID Set to GROUPID from IRM field IRM_RACF_GROUPID or from default GROUPID from IMS Connect configuration file. | 1 |

*Table 124. HWSECFDS DSECT - OTMA RACF UTOKEN Definition (Security Data UTOKEN Section for All Messages)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMSECRLN | 1 | 0 | | LENGTH OF UTOKEN FIELDS<br><br>Set to length of UTOKEN fields. The length includes this field. Set to X'51' if user security exit issued RACF call. | 1 |
| OMSECRTY | 1 | 1 | OMSECRXX X'02' | FIELD TYPE UTOKEN type.<br><br>Set to X'00' to identify UTOKEN present. | 1 |
| OMSECPRF | 80 | 2 | | UTOKEN<br><br>Set to UTOKEN from user security exit. | 1 |

*Table 125. HWSOMUSR DSECT - User Data Header (User Data Common Section for All Messages)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMUSRLN | 2 | 0 | | USER DATA LENGTH<br><br>Set to length of User Data Header. | 1 |
| | 2 | 2 | | ALIGN FULLWORD. | |

*Table 125. HWSOMUSR DSECT - User Data Header (User Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMUSR_DESTID | 8 | 4 | | DESTINATION ID<br><br>Set to destination ID (datastore) from IRM field IRM_IMSDESTID. | 1 |
| OMUSR_ORIGID | 8 | C | | ORIGIN ID. Neither tested nor set by exit. | 4 |
| OMUSR_PORTID | 8 | 14 | | PORTID.<br>Neither tested nor set by exit. | 4 |
| OMUSR_LTOKEN | 8 | 1C | | LOGON TOKEN.<br>Neither tested nor set by exit. | 4 |
| OMUSR_RETCODE | 4 | 24 | | COMMUNICATION RETURN CODE. Neither tested nor set by exit. | 4 |
| OMUSR_RESCODE | 8 | 28 | | COMMUNICATION REASON CODE. Neither tested nor set by exit. | 4 |
| OMUSR_RTOKEN | 4 | 30 | | RESPONSE TOKEN - A(SVT) COMMUNICATION RETURN CODE. Neither tested nor set by exit. | 4 |
| OMUSR_PASSTICK | 8 | 34 | | RACFPASSWORD/ PASSTICKET.<br><br>Set to password from IRM field IRM_RACF_PW.<br><br>This field must be cleared before passing message back to IMS Connect. | 1 |
| OMUSR_FLAG1 | 1 | 3C | OMUSER_TRAN X'00' | FLAG 1<br><br>Transaction Socket. Set if transaction socket specified in IRM field IRM_SOCT has been set to IRM_SOCT_TRAN. | 4 |

*Table 125. HWSOMUSR DSECT - User Data Header (User Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| | | | OMUSER_PSOCKET X'10' | Persistent Socket<br><br>Set if persistent socket specified in IRM field IRM_SOCT has been set to IRM_SOCT_PER. | 4 |
| | | | OMUSER_NPSOCKET X'40' | Non-persistent Socket<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMUSR_FLAG2 | 1 | 3D | OMUSR_PWDTEXT X'01' | FLAG 2<br><br>PASSTCKT field is text password.<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMUSR_PWDBIN X'02' | PASSTCKT field is binary password.<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMUSR_TRSTUSR X'80' | Trusted user can be set by exit. | |
| OMUSR_FLAG3 | 1 | 3E | OMUSR_HDRCM0 X'40' | ORIGINAL SYNCHRONIZATION.<br><br>Commit Mode 0.<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| | | | OMUSR_HDRCM1 X'20' | Commit Mode 1 (Send Commit)<br><br>NEITHER TESTED NOR SET BY EXIT. | 4 |
| OMUSR_TIMER | 1 | 3F | OMUSR_ZERO X'E9' - X'nn'<br>**Note:** See Table 8 on page 53 for a range of values. | Wait for Read following ACK or RECEIVE for RESUME TPIPE.<br><br>See "IRM_TIMER Usage" on page 52 for this value. | 1 |
| OMUSR_USTAT | 4 | 40 | | USTAT ADDRESS. NEITHER TESTED NOR SET BY EXIT. | 4 |

*Table 125. HWSOMUSR DSECT - User Data Header (User Data Common Section for All Messages) (continued)*

| Field | Len. | Hex Offset | Field Value | Description and Settings | Note |
|---|---|---|---|---|---|
| OMUSR_APPL_NM | 8 | 44 | | PassTicket APPLname set to blanks or IRM value. | 1 |
| OMUSR_RESV3 | 4 | 4C | | RESERVED FOR IMS CONNECT USAGE. NEITHER TESTED NOR SET BY EXIT. | 3 |
| OMUSR_RESV4 | 4 | 50 | | RESERVED FOR IMS CONNECT USAGE. NEITHER TESTED NOR SET BY EXIT. | 3 |
| OMUSR_RESV5 | 4 | 54 | | RESERVED FOR IMS CONNECT USAGE. NEITHER TESTED NOR SET BY EXIT. | 3 |
| OMUSR_RESV6 | 4 | 58 | | RESERVED FOR IMS CONNECT USAGE. NEITHER TESTED NOR SET BY EXIT. | 3 |
| OMUSRDTA | n | 5C | | User defined area. | 5 |

- NOTE **1**: Set by READ routine of user-written exit.
- NOTE **2**: Set by IMS Connect.
- NOTE **3**: Reserved fields.
- NOTE **4**: Set by IMS Connect and not analyzed by user exit.
- NOTE **5**: User-defined area.

# Appendix C. HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 Security Actions

This appendix describes the security actions that the sample user message exits HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 take in different circumstances.

Table 126, Table 127, and Table 128 define the action that HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 take when they do not call the security exit (IMSLSECX).

*Table 126. USERID Results If Security Exit Not Called*

| | USERID field present in IRM | IRM USERID field blank/null | RACF parms results passed in OTMA Security Header |
|---|---|---|---|
| USERID | Yes | Yes | Default RACFID |
| USERID | Yes | No | IRM USERID |
| USERID | No | N/A | Default RACFID |

*Table 127. GROUPID Results If Security Exit Not Called*

| | GROUPID field present in IRM | IRM USERID field blank/null | RACF parms results passed in OTMA Security Header |
|---|---|---|---|
| GROUPID | Yes | Yes | Blanks/nulls |
| GROUPID | Yes | No | IRM GROUPID |
| GROUPID | No | N/A | Blanks/nulls |

*Table 128. Password Results If Security Exit Not Called*

| | Password field present in IRM | IRM PASSWORD field blank/null | RACF parms results passed in OTMA Security Header |
|---|---|---|---|
| PASSWORD | Yes | Yes | Blanks/nulls |
| PASSWORD | Yes | No | IRM PASSWORD |
| PASSWORD | No | N/A | Blanks/nulls |

Table 129, Table 130 on page 252, and Table 131 on page 252 define the action that HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 take when they call the security exit (IMSLSECX).

*Table 129. USERID Results If Security Exit Called; Returns Blank or Non-blank USERID*

| | USERID field present in IRM | IRM USERID field blank/null | Security exit return USERID | RACF parms results passed in OTMA Security Header |
|---|---|---|---|---|
| USERID | Yes | Yes | No | Default RACF USERID |
| USERID | Yes | Yes | Yes | Security exit returned USERID |
| USERID | Yes | No | No | USERID passed in IRM |

*Table 129. USERID Results If Security Exit Called; Returns Blank or Non-blank USERID (continued)*

| | **USERID field present in IRM** | **IRM USERID field blank/null** | **Security exit return USERID** | **RACF parms results passed in OTMA Security Header** |
|---|---|---|---|---|
| USERID | Yes | No | Yes | Security exit returned USERID |
| USERID | No | N/A | No | Default RACF USERID |
| USERID | No | N/A | Yes | Security exit returned USERID |

*Table 130. GROUPID Results If Security Exit Called; Returns Non-blank USERID*

| | **GROUPID field present in IRM** | **IRM GROUPID field blank/null** | **Security exit return GROUPID** | **RACF parms results passed in OTMA Security Header** |
|---|---|---|---|---|
| GROUPID | Yes | Yes | No | Blank GROUPID |
| GROUPID | Yes | Yes | Yes | Security exit returned GROUPID |
| GROUPID | Yes | No | No | Blank GROUPID |
| GROUPID | Yes | No | Yes | Security exit returned GROUPID |
| GROUPID | No | N/A | No | Blank GROUPID |
| GROUPID | No | N/A | Yes | Security exit returned GROUPID |

*Table 131. GROUPID Results If Security Exit Called; Returns Blank USERID*

| | **GROUPID field present in IRM** | **IRM GROUPID field blank/null** | **Security exit return GROUPID** | **RACF parms results passed in OTMA Security Header** |
|---|---|---|---|---|
| GROUPID | Yes | Yes | No | Blank GROUPID |
| GROUPID | Yes | Yes | Yes | Blank GROUPID |
| GROUPID | Yes | No | No | IRM GROUPID |
| GROUPID | Yes | No | Yes | IRM GROUPID |
| GROUPID | No | N/A | No | Blanks |
| GROUPID | No | N/A | Yes | Blanks |

**Important:** If the security exit returns a blank USERID, then the GROUPID that is returned by the exit is not used.

Table 132 on page 253 defines the action that HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1 take regardless of whether the security exit (IMSLSECX) is called. The password is based on the IRM, not on the security exit.

*Table 132. Password Results Regardless of Whether Security Exit Called*

| | **PASSWORD field present in IRM** | **PASSWORD field blank/null** | **Security exit return PASSWORD** | **RACF parms results passed in OTMA Security Header** |
|---|---|---|---|---|
| PASSWORD | Yes | Yes | N/A | Blanks/nulls |
| PASSWORD | Yes | No | N/A | IRM PASSWORD |
| PASSWORD | No | N/A | N/A | Blanks/nulls |

IMS Connect decides what error actions to take depending on the RACF parameter setting in the IMS Connect configuration file, as well as the specific circumstances that cause the error.

- Table 133 describes the error actions that IMS Connect takes if RACF=Y, based on required RACROUTE call parameters.
- Table 134 on page 254 describes the error actions that IMS Connect takes if RACF=Y, either based on OTMA header data, or should the RACROUTE call fail.
- Table 135 on page 255 describes the error actions that IMS Connect takes if RACF=N, based on required RACROUTE call parameters.
- Table 136 on page 256 describes the error actions that IMS Connect takes if RACF=N, either based on OTMA header data, or should the RACROUTE call fail.

*Table 133. IMS Connect Error Actions Taken Based on RACROUTE Call Parameters (RACF=Y)*

| **USERID** | **PASSWORD** | **GROUPID** | **Action Taken** |
|---|---|---|---|
| Non-blanks | Non-blanks | Non-blanks | RACROUTE call issued |
| Non-blanks | Blanks | Blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X'04'<br>• Set OMUSR_RESCODE='SECFNOPW'<br>• Password cleared in OTMA header<br>• * Security failed, no password * |
| Non-blanks | Blanks | Non-blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X'04'<br>• Set OMUSR_RESCODE='SECFNOPW'<br>• Password cleared in OTMA header<br>• * Security failed, no password * |
| Blanks | Non-blanks | Blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X'04'<br>• Set OMUSR_RESCODE='SECFNUID'<br>• Password cleared in OTMA header<br>• * Security failed, no password * |

*Table 133. IMS Connect Error Actions Taken Based on RACROUTE Call Parameters (RACF=Y) (continued)*

| USERID | PASSWORD | GROUPID | Action Taken |
|---|---|---|---|
| Blanks | Non-blanks | Non-blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X’04’<br>• Set OMUSR_RESCODE=’SECFNUID’<br>• Password cleared in OTMA header<br>• * Security failed, no password * |
| Blanks | Blanks | Non-blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X’04’<br>• Set OMUSR_RESCODE=’SECFNPUI’<br>• Password cleared in OTMA header<br>• * Security failed, no password * |
| Blanks | Blanks | Blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X’04’<br>• Set OMUSR_RESCODE=’SECFNPUI’<br>• Password cleared in OTMA header<br>• * Security failed, no password * |
| Non-blanks | Blanks | Non-blanks | • Error message HWSP1503 issued<br>• Input rejected<br>• RACROUTE call not issued<br>• Set OMUSR_RETCODE=X’04’<br>• Set OMUSR_RESCODE=’SECFNOPW’<br>• Password cleared in OTMA header<br>• * Security failed, no password * |

*Table 134. IMS Connect Error Actions Taken for RACROUTE Call Failure or OTMA Header Data (RACF=Y)*

| RACROUTE Call Failure or OTMA Header Data | Action Taken |
|---|---|
| No security header | • Error message HWSP1503 issued<br>• Input rejected<br>• Set OMUSR_RETCODE=X’04’<br>• Set OMUSR_RESCODE=’NOSECHDR’<br>• Password cleared in OTMA header<br>• No RACF call made |

Table 134. IMS Connect Error Actions Taken for RACROUTE Call Failure or OTMA Header Data (RACF=Y) (continued)

| RACROUTE Call Failure or OTMA Header Data | Action Taken |
|---|---|
| Security header < X'6A' | • Error message HWSP1503 issued<br>• Input rejected<br>• Set OMUSR_RETCODE=X'04'<br>• Set OMUSR_RESCODE='INVSECHL'<br>• Password cleared in OTMA header<br>• No RACF call made |
| Conversation continued | • No error message issued<br>• Input accepted<br>• Password cleared in OTMA header<br>• No RACF call made |
| Response message | • No error message issued<br>• Input accepted<br>• Password cleared in OTMA header<br>• No RACF call made |
| UTOKEN present | • No error message issued<br>• Input accepted<br>• Password cleared in OTMA header<br>• No RACF call made |
| RACROUTE call failed | • Error message HWSP1500 issued<br>• Input rejected<br>• Set OMUSR_RETCODE=X'04'<br>• Set OMUSR_RESCODE='SECFAIL'<br>• Password cleared in OTMA header<br>• RACF return/reason codes in HWSP1500 message |
| All others | See Table 133 on page 253 |

Table 135. IMS Connect Error Actions Taken Based on RACROUTE Call Parameters (RACF=N)

| USERID | PASSWORD | GROUPID | Action Taken |
|---|---|---|---|
| Non-blanks | Non-blanks | Non-blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |
| Non-blanks | Blanks | Blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |
| Non-blanks | Blanks | Non-blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |

*Table 135. IMS Connect Error Actions Taken Based on RACROUTE Call Parameters (RACF=N) (continued)*

| USERID | PASSWORD | GROUPID | Action Taken |
|---|---|---|---|
| Blanks | Non-blanks | Blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |
| Blanks | Non-blanks | Non-blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |
| Blanks | Blanks | Non-blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |
| Blanks | Blanks | Blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |
| Non-blanks | Blanks | Non-blanks | • Password cleared<br>• Bypass RACROUTE call<br>• Pass these parms to OTMA |

*Table 136. IMS Connect Error Actions Taken for RACROUTE Call Failure or OTMA Header Data (RACF=N)*

| RACROUTE Call Failure or OTMA Header Data | Action Taken |
|---|---|
| No security header | • Password cleared<br>• Bypass RACROUTE call<br>• Pass OTMA headers and data to IMS OTMA |
| Security header < X’6A’ | • Password cleared<br>• Bypass RACROUTE call<br>• Pass OTMA headers and data to IMS OTMA |
| Conversation continued | • Password cleared<br>• Bypass RACROUTE call<br>• Pass OTMA headers and data to IMS OTMA |
| Response message | • No error message issued<br>• Input accepted<br>• Password cleared in OTMA header<br>• No RACF call made |
| UTOKEN present | • Password cleared<br>• Bypass RACROUTE call<br>• Pass OTMA headers and data to IMS OTMA |
| RACROUTE call failed | • Password cleared<br>• Bypass RACROUTE call<br>• Pass OTMA headers and data to IMS OTMA |

*Table 136. IMS Connect Error Actions Taken for RACROUTE Call Failure or OTMA Header Data (RACF=N) (continued)*

| RACROUTE Call Failure or OTMA Header Data | Action Taken |
|---|---|
| All others | • Password cleared<br>• Bypass RACROUTE call<br>• Pass OTMA headers and data to IMS OTMA |

# Appendix D. IMS Connect JCL

This appendix provides sample JCL examples to assist you when link-editing and compiling these exits:

- HWSSMPL0
- HWSSMPL1
- HWSJAVA0
- HWSDRU0
- HWSUINIT

See "Customizing IMS Connect" on page 29 for more information about how to customize these four exits.

In this appendix:

## HWSSMPL0 Sample JCL

For the HWSSMPL0 user message exit.

```
//HWSSMPL JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//SMPL01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=SYS1.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//           DD DSN=IMSHWS.SHWSMAC,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSSMPL0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMSBLD.IMSCON22.APAR.MAINT.SHWSSRC(HWSSMPL0),DISP=SHR
//SMPL02 EXEC PGM=IEWL,
//        PARM='SIZE=(180K,28K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSBLD.USERTEMP.HWSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSSMPL0)
 ENTRY   HWSSMPL0
 MODE RMODE(24),AMODE(31)
 NAME HWSSMPL0(R)
//
```

## HWSSMPL1 Sample JCL

For the HWSSMPL1 user message exit.

```
//HWSSMPL JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//SMPL01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=SYS1.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//           DD DSN=IMSHWS.SHWSMAC,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSSMPL1)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMSBLD.IMSCON22.APAR.MAINT.SHWSSRC(HWSSMPL1),DISP=SHR
//SMPL02 EXEC PGM=IEWL,
//        PARM='SIZE=(180K,28K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSBLD.USERTEMP.HWSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSSMPL1)
 ENTRY   HWSSMPL1
 MODE RMODE(24),AMODE(31)
 NAME HWSSMPL1(R)
//
```

## HWSJAVA0 Sample JCL

For the HWSJAVA0 user message exit.

```
//HWSJAVA JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//JAVA01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=SYS1.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=IMSHWS.SHWSMAC,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSJAVA0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMSBLD.IMSCON22.APAR.MAINT.SHWSSRC(HWSJAVA0),DISP=SHR
//JAVA02 EXEC PGM=IEWL,
//        PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSHWS.USERTEMP.HWSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSJAVA0)
 ENTRY   HWSJAVA0
 NAME HWSJAVA0(R)
//
```

## HWSYDRU0 Sample JCL

For the HWSYDRU0 sample OTMA DRU exit.

```
//HWSYDRU JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//YDRU01 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE),SYSPARM(HWSYDRU0)'
//SYSLIB DD DSN=SYS1.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=IMSHWS.SHWSMAC,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSYDRU0)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMSBLD.IMSCON22.APAR.MAINT.SHWSSRC(HWSYDRU0),DISP=SHR
//YDRU02 EXEC PGM=IEWL,
//        PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSBLD.USERTEMP.CRESLIB,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSYDRU0)
 ENTRY   HWSYDRU0
 NAME HWSYDRU0(R)
//
```

## HWSUINIT Sample JCL

For the HWSUINIT user initialization exit.

```
//HWSINIT JOB (ACTINF01),'PGMRNAME',
//             CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//UINIT1 EXEC PGM=ASMA90,REGION=32M,
//       PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE),SYSPARM(HWSUINIT)'
//SYSLIB DD DSN=SYS1.SDFSMAC,DISP=SHR
//       DD DSN=SYS1.MODGEN,DISP=SHR
//       DD DSN=IMSHWS.SHWSMAC,DISP=SHR
//           DD DSN=SYS1.MACLIB,DISP=SHR

//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//            DSN=&&TEXT(HWSUINIT)
//SYSPRINT DD SYSOUT=*,
//         DCB=(BLKSIZE=605),
//         SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1   DD UNIT=SYSDA,DISP=(,DELETE),
//         DCB=BLKSIZE=13024,
//         SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMSBLD.IMSCON22.APAR.MAINT.SHWSSRC(HWSUINIT),DISP=SHR
//UINIT2 EXEC PGM=IEWL,
//        PARM='SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD DSN=IMSBLD.USERTEMP.HWSRESL,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN   DD *
 INCLUDE   TEXT(HWSUINIT)
 ENTRY   HWSUINIT
 NAME HWSUINIT(R)
//
```

# Appendix E. Unicode Considerations

This appendix describes how IMS Connect handles Unicode data and describes in what circumstances this data is translated. IMS Connect support for Unicode allows Unicode data to be sent to and from an IMS Connect client application. This support requires that the client application and the IMS host application both support:

- Unicode data
- The same Unicode encoding schema (either UTF8, UTF16, or UCS-2) as the structure and content of the message being sent and received

IMS Connect supports ASCII and EBCDIC data streams both to and from the client application. If the client application sends ASCII data to IMS Connect, the ASCII is translated to EBCDIC. The subsequent output from IMS Connect to the client application is translated back to ASCII from EBCDIC. If the client application sends EBCDIC data to IMS Connect, no translation is required. With Unicode support, an IMS client application can also send and receive Unicode data to and from IMS Connect, specifically UTF8, UTF16, or UCS-2 data streams.

IMS Connect supports language groups 1, 2, and 3.

The client application uses the IMS Request Message (IRM) to:

- Tell IMS Connect if the data it is sending is Unicode and if the IMS transaction code is being sent as Unicode. IMS Connect transforms the transaction code if it is being sent as Unicode and then sends the transformed code and unicode data to IMS.
- Tell IMS Connect the Unicode encoding schema that is being used (UTF8, UTF16, or UCS-2).

The transaction code can be sent as Unicode, ASCII, or EBCDIC; however, it must be a valid IMS transaction code of up to 8 bytes that occupies an 8-byte field. In the field, the code must be left justified and, if it is shorter than 8 bytes, padded with blanks. If a blank follows the 8-byte transaction code field, it is considered to be part of the Unicode data.

In this appendix:

## Message Translation

All IMS error messages (for example, DFS555) are sent as either ASCII or EBCDIC. The client application uses the IRM_MSGID field of the IRM to tell IMS Connect which type to send. IMS Connect does not transform messages to Unicode. For example, if IRM_MSGID is EBCDIC, the IMS error message (DFSnnnn) is sent as EBCDIC; if IRM_MSGID is ASCII, the IMS error message (DFSnnnn) is translated from EBCDIC to ASCII.

IRM_MSGID also identifies the code type of the OTMA header.

An IMS client application can send the IMS transaction code as ASCII, EBCDIC, or Unicode. When the IMS client application sends the transaction code as Unicode, the IMS Connect user message exit (HWSSMPL0, HWSSMPL1, HWSIMSO0, and HWSIMSO1) translates the transaction code from Unicode TO EBCDIC. When the client application sends the transaction code as ASCII and the remaining data as Unicode, only the transaction code is translated to EBCDIC. A valid 8-byte IMS transaction code can be constructed from the following characters and must begin with an alphabetic character:

- A through Z (uppercase only)
- 0 through 9
- Special characters #, $, @

An IMS host application that supports Unicode must define an 8-byte field in the input message definition to contain the transaction code. If you pad the 8-byte field with a blank, it is sent as an EBCDIC blank.

If the client application sends Unicode data, the output message is not transformed and is treated as Unicode. For RESUME TPIPE requests, the client application must specify in the IRM if the output should be treated as Unicode or not. During message switching, the IMS host application must ensure that the output message is formatted correctly (using a specific Unicode schema or EBCDIC) for its destination.

## Input Message Format Sent by the Client

Table 137 contrasts the message structure for input messages sent by the client. The overall message structure is the same as the structure defined in “User Exit Message Description and Structures” on page 69. Table 137 defines the valid ASCII, EBCDIC, and UNICODE formats.

*Table 137. Input Message Structure - message sent by client*

| EBCDIC IRM | ASCII IRM | If OTMA headers are passed by client | Transaction Code | Data |
|---|---|---|---|---|
| Y | N/A | EBCDIC | EBCDIC | UNICODE |
| Y | N/A | EBCDIC | UNICODE | UNICODE |
| N/A | Y | ASCII | ASCII | UNICODE |
| N/A | Y | ASCII | UNICODE | UNICODE |

## Output Message Format Received by the Client

Table 138 defines the valid output message elements when the client sends UNICODE data. The overall message structure is the same as the structure defined in “User Exit Message Description and Structures” on page 69.

*Table 138. Output Message Structure - message received by client*

| If input message was EBCDIC IRM | If input message was ASCII IRM | RMM | RSM | Output CSM | Output data |
|---|---|---|---|---|---|
| Y | N/A | EBCDIC | EBCDIC | EBCDIC | UNICODE |
| N/A | Y | ASCII | ASCII | ASCII | UNICODE |

# Appendix F. Suggested TCP/IP Settings

The following TCP/IP settings are described to assist you with your TCP/IP installation. You can choose different TCP/IP values to maximize your environment settings. Here are some suggested values you can use:

**TCPNODELAY=ENABLE**

- Data is transmitted by TCP/IP per client SEND.
- TCP/IP waits one millisecond per transmission.
- Multiple client TCP/IP SENDS can result in multiple TCP/IP transmissions.

**TCPNODELAY=DISABLE**

- Data is collected by TCP/IP from client TCP/IP SENDS, before transmission.
- TCP/IP waits until the buffer is full before transmission.
- Multiple client SENDS results in 1 to *n* TCP/IP transmissions to IMS Connect.

**SO_LINGER=Y, VALUE=0**

- Immediate return to client code.
- A client request to close the socket can bypass data sent with a previous client TCP/IP SEND request, but may result in the loss of the client SEND data.

**SO_LINGER=N**

- Immediate return to client code.
- A client request to close the socket can bypass data sent with a previous client TCP/IP SEND request, but may result in the loss of the client SEND data.

**SO_LINGER=Y, VALUE=10**

- Return to client code when an ACK is received from the host, or wait for 10 seconds before sending close.
- Socket close will not bypass data sent.

**DELAYACK**

DELAYACK is used to minimize non-data transmissions from the host. If DELAYACK is used, the MVS TCP/IP waits 200 milliseconds before sending an ACK to the remote server TCP/IP. However, if the ACK is appended to the data being sent from IMS Connect, there is no delay.

If your client application performs a single SEND followed by a READ, DELAYACK is recommended.

DELAYACK can be set on the TCP/IP ″Port Statement″ or on the ″Gateway Statement.″

**NODELAYACK**

NODELAYACK is used to allow non-data transmissions from the host to flow without data. If NODELAYACK is used, the MVS TCP/IP immediately sends an ACK to the remote server TCP/IP. The ACK is not appended to the data being sent from IMS Connect.

If the client code sends one SEND followed by a READ to the host with a NODELAYACK setting, an ACK is sent separately.

If the client code sends two or more SENDs followed by a READ to the host, the host TCP/IP will send an ACK immediately to the data received. This will allow the next SEND of data from the client to flow.

NODELAYACK is recommended if your client application sends more than one SEND followed by a READ.

NODELAYACK can be set on the TCP/IP ″Port Statement″ or on the ″Gateway Statement.″

# Appendix G. HWSTECL0 User Exit

For performance or basic data analysis, you may want to record specific data events. For example, you may wish to record events such as:

- TCP/IP read/write
- RACF calls
- OTMA send/receive
- User exit calls
- Session errors
- Two-phase commit events

IMS Connect can be customized to facilitate event recording by passing event data to the load module, HWSTECL0. This module stores all trace and event notifications through a recording routine and can be used by any event recording function. IMS Connect provides a sample HWSTECL0 user exit for you to customize.

In this appendix:

## Modifying HWSTECL0 User Exit

Although IMS Connect provides a sample HWSTECL0 user exit, you must modify the HWSTECL0 user exit, using standard user-exit development guidelines, if you want to receive event data from IMS Connect. The source code for the HWSTECL0 user exit is located in the AHWSSRC source library.

After you have customized the sample HWSTECL0 user exit, you must install it into your IMS Connect resource library (SHWSRESL). To install HWSTECL0 into the resource library, you must compile and bind (link-edit) the user exit before you execute IMS Connect to create the load module, HWSTECL0. IMS Connect will load your HWSTECL0 module from the resource library and call it during initialization and termination.

The following steps describe how to customize, modify, and re-install the HWSTECL0 exit.

1. Insert your changes to the source code provided in the AHWSSRC source library.
2. Assemble the exit. The exit and its associated macro files are members of the partitioned data set into which you receive the AHWSSRC data set. See “DSECTS for Event Recording” on page 300 for a list of the macro files associated with the HWSTECL0 exit.
3. Bind (link-edit) the output from the assembled job to create a load module named HWSTECL0.

4. Bind (link-edit) HWSTECL0 into the IMS Connect resource library, SHWSRESL. IMS Connect loads the module from the resource library during initialization.

## HWSTECL0 Initialization

When IMS Connect initializes, IMS Connect automatically loads the HWSTECL0 module and calls the module for event recording initialization. If event and trace recording is detected and is active, module HWSTECL0 sets the Event Interface Control Block (EICB) fields, which is used to control event recording, to the appropriate values needed for event and trace recording. For more information about the EICB fields, see “Event Interface Control Block (EICB)” on page 297. The address of the EICB is pointed to by HWSTECL0 register on Register 1 entry. Note, event initialization will only occur if the caller is executing under the JOBSTEP TCB, the caller is in primary TCB mode, and the call occurs before any task that records events is created. Table 139 describes the registers at entry to HWSTECL0.

*Table 139. Registers at entry to HWSTECL0*

| Register Number | Contents and meaning |
|---|---|
| R1 | Address of the Event Interface Control Block (EICB) that is to be completed by HWSTECL0 when trace or even recording is active. |
| R13 | Address of save area that is a set of pre-chained save areas. HWSTECL0 must preserve the integrity of the save area set. |
| R14 | Caller’s return address. |
| R15 | Entry point of module HWSTECL0. |

The EICB area is allocated by IMS Connect and passed to HWSTECL0 at the initialization request. The DESECT name is HWSECIB. If trace or event recording is active, HWSTECL0 completes the EICB and returns it to the caller. The contents of the control block that are returned from HWSTECL0 are shown in Table 140.

*Table 140. Contents of Event Interface Control Block (EICB) Pointed to by HWSTECL0*

| Element | Length | Usage and Meaning |
|---|---|---|
| EYECATCHER | 4 | Value of EICB identifying this block in working storage. Set by caller. |
| FLAGS | 1 | Interface control flags: 1. Event recording is enabled. |
| EVENT_TOKEN | 4 | Address of the token used by the event recording routine. The token must be passed to the event recording routine when an event-recording request is made. |
| EVENT_ADDRESS | 4 | Entry address of event recording routine. |
|  | 4 | Reserved space. |
|  | 4 | Reserved space. |
| MESSAGE_LEN | 2 | Length of the message returned from HWSTECL0 module. |
| MESSAGE_AREA | 120 | An area that can be used by HWSTECL0 to return an informational or error message to IMS Connect. |

If trace or event recording is not active, HWSTECL0 does not complete the EICB and instead returns with a return and reason code indicating that trace or event recording, or both is not active. Table 141 describes the registers at return from HWSTECL0. **Note:** Module HWSTECL0 always returns a return code of 0. The EICB flags must be inspected to determine if event or trace recording is active.

*Table 141. Registers at return from HWSTECL0*

| Register Number | Contents and meaning |
|---|---|
| R0 | Reason code associated with any non-zero return codes passed. |
| R15 | Return code<br>• 0 = Initialization was successful. Check the EICB to see if trace or event recording is active.<br>• 8 = Initialization was not successful. See reason code for additional information. |

## Invoking HWSTECL0 for Event Recording

When IMS Connect records an event, IMS Connect calls the event recording routine address, EVENT_ADDRESS, indicated in the EICB. For each event that is recorded, the event recording routine passes the Event Record Parameter List (ERPL), which is used to define the event type and event data. The ERPL defines which event data to capture. The ERPL records an IMS Connect event and associated data to an event-recording log. See "Event Recording Parameter List (ERPL)" on page 296 for more information about the ERPL control block.

When event recording has been initialized, the EICB contains the entry address for event recording and calls the event recording routine. The routine points to the ERPL address and records the event. To record an event, the caller requesting event recording must be in primary TCB mode and the caller must return the event recording token which is provided in the EICB by HWSTECL0. Table 142 describes the registers at event recording entry.

*Table 142. Registers at Event Recording Entry*

| Register Number | Contents and meaning |
|---|---|
| R1 | Address of the Event Recording Parameter List (ERPL). |
| R13 | Address of one save area. The event recording routine must preserve the integrity of the save area. |
| R14 | Caller's return address. |
| R15 | Entry point of even recording taken from EICB after initialization of the even recording interface. |

Table 143 shows the registers at return from EICB, the event recording interface.

*Table 143. Registers at Return from Event Recording*

| Register Number | Contents and meaning |
|---|---|
| R0 | Reason code associated with any non-zero return codes passed. |
| R1 | When R1 is not equal to zero, it contains the address of a message providing additional information about initialization of trace and event recording. |

Table 143. Registers at Return from Event Recording (continued)

| Register Number | Contents and meaning |
|---|---|
| R15 | Return code<br>• 0 = Event recording was successful.<br>• 4 = Event recording is not active -- event was not recorded.<br>• 16 = Event recording was not successful. See reason code for additional information. An error message is present if R1 is not zero. |

## Error Message Format

If an error message is returned by the event-recording routine, the format of the error message is described in Table 144. Note, the use of error messages is optional and currently is not supported.

Table 144. Error Message Format

| Value | Contents and meaning |
|---|---|
| 2 byte message length | The true length of the error message not including the message length field. |
| Error message | An error message returned by the recording exit. |

## Event Types

There are two types of events, single and multiple.

**single event**
An event that is not related to any other events.

**multiple event**
Events that are closely related to each other within a process such as a transaction.

Each event is assigned a numerical value called an event number. Each event also has an associated key value of either EVNT or SVTOKEN. These two key values indicate whether or not an event is associated with a multiple event process. Table 145 describes the key values and the length of the event key.

Table 145. Keys Associated with Events

| Key value | Length | Usage and meaning |
|---|---|---|
| EVNT | 8 | This is a constant value used to indicate the event is not associated with a multi-event process. The constant is left-justified and padded right with blanks. |
| SVTOKEN | 8 | SVT Token. A token representing the SVT control block for the remote client name associated with the transaction or multi-event process. The token is the STCK time of when the SVT was created. |

The following table identifies the events that are categorized as a single event type. Table 146 on page 271 lists the possible single events that may be recorded.

*Table 146. Single Process Events*

| Event Number | Event Key | Event Description |
|---|---|---|
| 1 | EVNT | Connect region initialization. This event record is generated as a result of the call made to module HWSTECL0 for event recording initialization. This is the first event recorded for an IMS Connect execution. |
| 2 | EVNT | Connect region has completed termination. This event is the last event recorded for an IMS Connect execution. This event causes the event recording process to terminate. |
| 3 | EVNT | A support task (TCB) has been created. If the task records events, this event must be the first event recorded by the task. It should be recorded as soon as possible after the task begins processing. |
| 4 | EVNT | A support task (TCB) is terminating. If the task records events, this event must be the last event recorded by the task. It should be recorded as close as possible to the task returning to MVS. |
| 5 | EVNT | Begin INIT API. |
| 6 | EVNT | End INIT API |
| 7 | EVNT | Begin Bind socket. |
| 8 | EVNT | End Bind socket. |
| 9 | EVNT | Listen on socket. |
| 10 | EVNT | Begin Accept socket. |
| Note | | Events 12 and 13 are defined in the section on multi-event types. |
| 14 | EVNT | Begin init of message exits. This event serves to initialize the task for message exit processing. |
| 16 | EVNT | IMS datastore becomes available. The event is recorded during the following processes. It represents a successful client bid process.<br>1. During IMS Connect initialization - once for each available datastore.<br>2. After IMS Connect initialization - any time a datastore joins the XCF group and client bid is completed. |
| 17 | EVNT | IMS datastore becomes unavailable. This event represents a datastore that has become unavailable for transactions. It could be due to a stop datastore command or the datastore member leaving the XCF group. The event is recorded for either occurrence. |
| 18 | EVNT | An IMS TMEMBER joins the XCF group. |
| 19 | EVNT | An IMS TMEMBER leaves the XCF group. |
| 20 | EVNT | Begin SCI registration. |
| 21 | EVNT | End SCI registration. |
| 22 | EVNT | Begin SCI De-registration. |
| 23 | EVNT | End SCI De-registration. |

Table 146. Single Process Events (continued)

| Event Number | Event Key | Event Description |
|---|---|---|
| 24 | EVNT | Recorded trace DCB has been opened. This event recorded after the recorder trace DCB has been successfully opened. |
| 25 | EVNT | Recorded trace DCB pre-close. This event is recorded when the recorder trace DCB is about to be closed. This event is recorded while the recorder trace DCB is still open. |
| 26 | EVNT | User message exit return from INIT. This event is recorded just after the user message exit returns. |
| 27 | EVNT | User message exit return from TERM. This event is recorded just after the user message exit returns. |
| 28 | EVNT | Begin Secure Environment Open. This is issued at the start of SSL environment creation. |
| 29 | EVNT | End Secure Environment Open. This is issued at the end of SSL environment creation. |
| 32 | EVNT | Begin Secure Environment Close. This is issued at the start of SSL close. |
| 33 | EVNT | End Secure Environment Close. This is issued at the end of SSL initialization. |
| 34 | EVNT | Begin Local Port Setup. This event is recorded when a local port is present. |
| 35 | EVNT | End Local Port Setup. This event is recorded when a local port is present. |
| 36 | EVNT | Begin RRS Connect. This event is recorded when RRS connect processing is started. |
| 37 | EVNT | End RRS Connect. This event is recorded when RRS connect processing is completed. |
| 38 | EVNT | List In-doubt Context. This event records the receipt of an in-doubt context during RRS connect processing. |
| 39 | EVNT | Begin RRS Disconnect. This event is recorded when RRS disconnect processing is started. |
| 40 | EVNT | End RRS Disconnect. This event is recorded when RRS disconnect processing is completed. |

The following table identifies the events that are categorized as a multiple event type. Table 147 lists the possible multiple events that may be recorded.

Table 147. Multi-process Events

| Event Number | Event Key | Event description |
|---|---|---|
| 12 | SVT Token | Begin close socket. |
| 13 | SVT Token | End close socket. |
| 60 | SVT Token | Prepare for socket read. This is the start-of-frame event for a multi-event process. It is the first event associated with an SVT Token. |
| 61 | SVT Token | User message exit entered for READ, XMIT, or EXER. This event is recorded just prior to calling the user message exit. |

*Table 147. Multi-process Events (continued)*

| Event Number | Event Key | Event description |
|---|---|---|
| 62 | SVT Token | User message exit return for READ, XMIT, or EXER. This event is recorded just after the user message exit returns. |
| 63 | SVT Token | Begin SAF security request. |
| 64 | SVT Token | End SAF security request. |
| 65 | SVT Token | Message sent to OTMA. This entry is made after the message has been sent to OTMA. |
| 66 | SVT Token | Message received from OTMA. This entry is made when a message has been received from OTMA. It is recorded after all parts of the message have been assembled. |
| 67 | SVT Token | Message sent to SCI. This entry is made after the message has been sent to SCI. |
| 68 | SVT Token | Message received from SCI. This entry is made as soon as a message has been received from SCI. |
| 69 | SVT Token | OTMA time-out. This event signals that a time-out occurred for an OTMA request. |
| 70 | SVT Token | De-allocate request. This event is generated when IMS Connect honors a request from the remote client to disconnect the session. |
| 71 | SVT Token | Session error. This event is called when an unrecoverable error has been encountered and the session is being aborted. Fro this error condition, this should probably be the last event before the trigger event is recorded. |
| 72 | SVT Token | Trigger event. This is the end-of-frame event recorded by IMS Connect when a multi-event process has completed. |
| 73 | SVT Token | Read socket. |
| 74 | SVT Token | Write socket. |
| 75 | SVT Token | Local client connect. This event is issued at receipt of the client connect call (logon). |
| 76 | SVT Token | Local message send. This event is issued when IMS Connect receives a local client message. The send orientation is to the local client. |
| 77 | SVT Token | Local message receive. This event is issued when IMS Connect sends a local client message. The receive orientation is to the local client. |
| 78 | SVT Token | Local message send-then-receive. This event is issued when IMS Connect receives a local client message. The local client waits until the output message is ready and IMS Connect sends the message back to the local client. The send and receive orientation is to the local client. |
| 79 | SVT Token | Local disconnect. This event is issued when IMS Connect disconnects from a local client (logoff). |
| 80 | SVT Token | Begin create context. This event records the request to RRS to create a context for a transaction requesting two-phase commit support. |

*Table 147. Multi-process Events (continued)*

| Event Number | Event Key | Event description |
|---|---|---|
| 81 | SVT Token | End create context. This event records the end of creation of context for a transaction requesting two-phase commit support. |
| 82 | SVT Token | Begin RRS prepare. This event records sending the prepare-to-commit request to RRS. |
| 83 | SVT Token | End RRS prepare. This event records receiving the response to the prepare-to-commit request. |
| 84 | SVT Token | Begin RRS commit/abort. This event records sending the commit/abort request to RRS. |
| 85 | SVT Token | End RRS commit/abort. This event records receiving a response to the commit/abort request. |
| 86 | SVT Token | Begin secure environment select. This is issued at the end of SSL select. |
| 87 | SVT Token | End secure environment select. This is issued at the end of SSL select. |

## Event Record Formats

The following tables list the format for all event records. Each table identifies each possible event in the ERPL (Event Record Parameter List) that can be recorded to the HWSTECL0 module and provides the format for each event.

Table 148 identifies the parameter list content associated with the IMS Connect region initialization event.

*Table 148. Connect Region Initialization Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 1 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 4 | 2 |
| VAR_DATA | Start of variable data area. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_VVRR | IMS Connect Version and Release data. | 2 |

Table 149 identifies the parameter list contents associated with the Connect region termination.

*Table 149. Connect Region Termination Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 2 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |

*Table 149. Connect Region Termination Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| VAR_DATA | Start of variable data area. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_COMPCODE | Completion code associated with region termination. | 4 |

Table 150 identifies the parameter list contents associated with the Support Task Created event.

*Table 150. Support Task Created Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 3 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |
| VAR_DATA | Start of variable data area. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_FLAG | Flag field indicating TCB type:<br>1. port<br>2. local<br>3. recorder | 2 |
| VAR_PORT | Port number if port task. | 2 |

Table 151 identifies the parameter list contents associated with the Support Task Terminating event.

*Table 151. Support Task Terminating Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 4 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_FLAG | Flag field indicating TCB type:<br>1. port<br>2. local<br>3. recorder | 2 |
| VAR_PORT | Port number if port task. | 2 |

Table 152 on page 276 identifies the parameter list contents associated with the event, Begin Initialize API.

Table 152. Begin Initialize API Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 5 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 153 identifies the parameter list contents associated with the event, End Initialize API.

Table 153. End Initialize API Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 6 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 10 | 2 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 154 identifies the parameter list contents associated with the Begin Bind Socket event. If this is a secure socket (SSL), the TCPIB (TCP/IP Information Block) contains a flag indicating the operation is executing against an SSL port.

Table 154. Begin Bind Socket Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 7 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB | 4 |

Table 155 identifies the parameter list contents associated with the End Bind Socket event. If this is a secure socket (SSL), the TCPIB (TCP/IP Information Block) contains a flag indicating the operation is executing against an SSL port.

Table 155. End Bind Socket Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 8 | 4 |

*Table 155. End Bind Socket Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 10 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 156 identifies the parameter list contents associated with the Listen on Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operations is executing against an SSL port.

*Table 156. Listen on Socket Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 9 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB | 4 |

Table 157 identifies the parameter list contents associated with the Begin Accept Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operation is executing against an SSL port.

*Table 157. Begin Accept Socket Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 10 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 158 identifies the parameter list contents associated with the End Accept Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operation is executing against an SSL port.

*Table 158. End Accept Socket Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 11 | 4 |

Table 158. End Accept Socket Event (continued)

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 10 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 159 identifies the parameter list content associated with the Begin Initialization of Message Exits event.

Table 159. Begin Initialization of Message Exits

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 14 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 160 identifies the parameter list contents associated with the Datastore Available event.

Table 160. Datastore Available Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 16 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 161 identifies the parameter list contents associated with the Datastore Unavailable event.

Table 161. Datastore Unavailable Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 17 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 161. Datastore Unavailable Event (continued)

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 162 identifies the parameter list contents associated with the TMEMBER Joins XCF Group event.

Table 162. TMEMBER Joins XCF Group Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 18 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 163 identifies the parameter list contents associated with the TMEMBER Leaves XCF Group event.

Table 163. TMEMBER Leaves XCF Group Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 19 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 164 identifies the parameter list contents associated with the Begin SCI Registration event.

Table 164. Begin SCI Registration Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 20 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 165 on page 280 identifies the parameter list contents associated with the End SCI Registration event.

Table 165. End SCI Registration Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 21 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 10 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 166 identifies the parameter list contents associated with the Begin SCI De-registration event.

Table 166. Begin SCI De-registration Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 22 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 167 identifies the parameter list contents associated with the End SCI De-registration event.

Table 167. End SCI De-registration Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 23 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 10 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 168 on page 281 identifies the parameter list contents associated with the Recorder Trace DCB Opened event.

Table 168. Recorder Trace DCB Opened Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 24 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the recorder trace DCB. | 4 |

Table 169 identifies the parameter list contents associated with the Recorder Trace DCB Pre-close event.

Table 169. Recorder Trace DCB Pre-close Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 25 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 170 identifies the parameter list contents associated with the Message Exit INIT Call event.

Table 170. Message Exit INIT Call Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 26 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 18 | 2 |
| EVENT_DATA_ADDR | Address of the exit parameter list. | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |
| VAR_EXIT_NAME | Name of the user message exit. | 8 |

Table 171 identifies the parameter list contents associated with the Message Exit TERM Call event.

Table 171. Message Exit TERM Call Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 27 | 4 |

*Table 171. Message Exit TERM Call Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 18 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |
| VAR_EXIT_NAME | Name of the user message exit. | 8 |

Table 172 identifies the parameter list contents associated with the Begin Secure Environment Open event.

*Table 172. Begin Secure Environment Open Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 28 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 173 identifies the parameter list contents associated with the End Secure Environment Open event.

*Table 173. End Secure Environment Open Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 29 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 10 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 174 on page 283 identifies the parameter list contents associated with the Begin Secure Environment Close event.

*Table 174. Begin Secure Environment Close Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 32 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 175 identifies the parameter list contents associated with the End Secure Environment Close event.

*Table 175. End Secure Environment Close Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 33 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 176 identifies the parameter list contents associated with the Begin Local Port Setup event.

*Table 176. Begin Local Port Setup Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 34 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 177 identifies the parameter list contents associated with the End Local Port Setup event.

*Table 177. End Local Port Setup Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 35 | 4 |

*Table 177. End Local Port Setup Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 14 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 178 identifies the parameter list contents associated with the Begin RRS Connect event.

*Table 178. Begin RRS Connect Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 36 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 179 identifies the parameter list contents associated with the End RRS Connect event.

*Table 179. End RRS Connect Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 37 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |

Table 180 identifies the parameter list contents associated with the List In-doubt Context event.

*Table 180. List In-doubt Context Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 38 | 4 |
| EVENT_KEY | EVNT | 8 |

*Table 180. List In-doubt Context Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 162 | 2 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_URTOKEN | The UR_INTEREST_TOKEN returned by RRS. | 16 |
| VAR_XID | The XID associated with this transaction | 140 |

Table 181 identifies the parameter list contents associated with the Begin RRS Disconnect event.

*Table 181. Begin RRS Disconnect Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 39 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 182 identifies the parameter list contents associated with the End RRS Disconnect event.

*Table 182. End RRS Disconnect Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 40 | 4 |
| EVENT_KEY | EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |

Table 183 identifies the parameter list contents associated with the Begin Close Socket event. If this is a secure socket (SSL), the TCPIB (TCP/IP Information Block) contains a flag indicating that the operations is executing against an SSL port.

*Table 183. Begin Close Socket Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 12 | 4 |

Table 183. Begin Close Socket Event (continued)

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 184 identifies the parameter list contents associated with the End Close Socket event. If this is a secure socket, the TCPIB contains a flag indicating the operations is executing against an SSL port.

Table 184. End Close Socket Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 13 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 10 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

Table 185 identifies the parameter list contents associated with the Prepare Socket Read event.

Table 185. Prepare Socket Read Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 60 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 186 identifies the parameter list contents associated with the Message Exit Called for READ, XMIT, or EXER event.

Table 186. Message Exit Called for READ, XMIT, or EXER Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 61 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 2 | 2 |
| VAR_DATA_LL | 10 | 2 |

*Table 186. Message Exit Called for READ, XMIT, or EXER Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_DATA_ADDR | Address of the parameter list at entry (R1). | 4 |
| EVENT_DATA_ADDR2 | If READ or EXER, address of the IRM header. If XMIT, address of the OTMA header. | 4 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_EXIT_NAME | Exit name. | 8 |

Table 187 identifies the parameter list contents associated with the Message Exit Return for READ, XMIT, or EXER event.

*Table 187. Message Exit Return for READ, XMIT, or EXER Event.*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 62 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 2 | 2 |
| VAR_DATA_LL | 18 | 2 |
| EVENT_DATA_ADDR | Address of the parameter list at entry (R1). | 4 |
| EVENT_DATA_ADDR2 | If XMIT or EXER, address of the remote client message. If READ, address of the OTMA header. | 4 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |
| VAR_EXIT_NAME | Exit name. | 8 |

Table 188 identifies the parameter list contents associated with the Begin SAF Request event.

*Table 188. Begin SAF Request Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 63 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the SAFIB. | 4 |

Table 189 on page 288 identifies the parameter list contents associated with the End SAF Request event.

Table 189. End SAF Request Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 64 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the SAFIB. | 4 |

Table 190 identifies the parameter list contents associated with the Message Sent to OTMA event.

Table 190. Message Sent to OTMA Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 65 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 191 identifies the parameter list contents associated with the Message Received from OTMA event.

Table 191. Message Received From OTMA Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 66 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 192 identifies the parameter list contents associated with a Message Sent to SCI event.

Table 192. Message Sent to SCI Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 67 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 193 identifies the parameter list contents associated with a Message Received from SCI event.

*Table 193. Message Received From SCI Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 68 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the DSIB. | 4 |

Table 194 identifies the parameter list contents associated with an OTMA Time-out event.

*Table 194. OTMA Time-out Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 69 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_TO_VALUE | Time-out value. | 4 |

Table 195 identifies the parameter list contents associated with a De-allocate Session event.

*Table 195. De-allocate Session Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 70 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 6 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_DEALC_RSN | Reason for session de-allocation. Note: Can be a flag or constant type of reason. | 4 |

Table 196 on page 290 identifies the parameter list contents associated with a Session Error event.

Table 196. Session Error Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 71 | 4 |
| EVENT_KEY | SVT Token or EVNT | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 154 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_FLAG | Flag fields indicating record content. 1. Message is present in the record. 2. Out-of-frame error. | 2 |
| VAR_MESSAGE | If a message is generated for the error, it is contained in this field. | 134 |
| VAR_SESS_RSN | Reason for the session de-allocation. Note: The session reason is a character expression of the error type. | 8 |
| VAR_SESS_TOKEN | The SVTTOKEN associated with the message when the session error occurs out-of-frame and the SVTTOKEN for the message cannot be located by IMS Connect. Note: This field is valid only when the key of the event is EVNT. If the key is an SVTTOKEN value, this field is zero. In some cases, where asynchronous output is created by a non-IMS Connect source, the field may contain values that do not resemble a normal IMS Connect SVTTOKEN. | 8 |

Table 197 identifies the parameter list contents associated with a Trigger event.

Table 197. Trigger Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 72 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 10 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_TRIG_TYPE | Constant identifying triggers type. Values can be TRAN or TPIPE or anything else that is needed. | 8 |

Table 198 on page 291 identifies the parameter list contents associated with a Read Socket event.

*Table 198. Read Socket Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 73 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 199 identifies the parameter list contents associated with the Write Socket event.

*Table 199. Write Socket Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 74 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 200 identifies the parameter list contents associated with the Local Client Connect event.

*Table 200. Local Client Connect Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 75 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 201 identifies the parameter list contents associated with the Local Message Send event. This event is completed following the event recording of the SRB scheduling and may not precisely mark the actual completion of the operation.

*Table 201. Local Message Send Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 76 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 202 identifies the parameter list contents associated with the Local Message Receive event. This event is completed following the event recording of the SRB scheduling and may not precisely mark the actual completion of the operation.

*Table 202. Local Message Receive*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 77 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 203 identifies the parameter list contents associated with the Local Message Send/Receive event. This event is completed following the copy of the SRB scheduling and may not precisely mark the actual completion of the operation.

*Table 203. Local Message Send/Receive Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 78 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 204 identifies the parameter list contents associated with the Local Client Disconnect event.

*Table 204. Local Client Disconnect Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 79 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 0 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |

Table 205 identifies the parameter list contents associated with the Begin Create Context event.

*Table 205. Begin Create Context Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 80 | 4 |

Table 205. Begin Create Context Event (continued)

| Parameter list item | Content | Length in bytes |
|---|---|---|
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 0 | 2 |

Table 206 identifies the parameter list contents associated with the End Create Context event.

Table 206. End Create Context Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 81 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 162 | 2 |
| VAR_DATA | Start of variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | RRS return code. | 4 |
| VAR_URTOKEN | UR Interest token returned from RRS. | 16 |
| VAR_XID | The remote client XID associated with the transaction. | 140 |

Table 207 identifies the parameter list contents associated with the Begin RRS Prepare event.

Table 207. Begin RRS Prepare Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 82 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 18 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_URTOKEN | URTOKEN associated with the request. | 16 |

Table 208 identifies the parameter list contents associated with the End RRS Prepare event.

Table 208. End RRS Prepare Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 83 | 4 |
| EVENT_KEY | SVT Token | 8 |

*Table 208. End RRS Prepare Event (continued)*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 24 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_FLAG | Result flags: 1. At least 1 participant replied abort. Note: The results flag is set if any participant has requested the context be aborted. | 2 |
| VAR_URTOKEN | URTOKEN associated with the request. | 16 |

Table 209 identifies the parameter list contents associated with the Begin RRS Commit/Abort event.

*Table 209. Begin RRS Commit/Abort Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 84 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 20 | 2 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_FLAG | Result flags: 1. request to abort 2. request to commit Note: The results flag is set if any participant has requested the context be aborted. | 2 |
| VAR_URTOKEN | URTOKEN associated with the request. | 16 |

Table 210 identifies the parameter list contents associated with the End RRS Commit/Abort event.

*Table 210. End RRS Commit/Abort Event*

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 85 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 0 | 2 |
| VAR_DATA_LL | 24 | 2 |
| VAR_DATA | Start of the variable data. | 0 |

Table 210. End RRS Commit/Abort Event (continued)

| Parameter list item | Content | Length in bytes |
|---|---|---|
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_FLAG | Result flags: 1. request to abort 2. request to commit 3. could not find the URTOKEN Note: The results flag is set if any participant has requested the context be aborted. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_URTOKEN | URTOKEN associated with the request. | 16 |

Table 211 identifies the parameter list contents associated with the Begin Secure Environment Select event.

Table 211. Begin Secure Environment Select Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 86 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 4 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |
| VAR_FLAG | Result flags: 1. Select for Read. 2. Select for Write | 2 |

Table 212 identifies the parameter list contents associated with the End Secure Environment Select event.

Table 212. End Secure Environment Select Event

| Parameter list item | Content | Length in bytes |
|---|---|---|
| TOKEN | Token address | 4 |
| EVENT_NUMBER | 87 | 4 |
| EVENT_KEY | SVT Token | 8 |
| DATA_ADDR_COUNT | 1 | 2 |
| VAR_DATA_LL | 12 | 2 |
| EVENT_DATA_ADDR | Address of the TCPIB. | 4 |
| VAR_DATA | Start of the variable data. | 0 |
| VAR_APAR | APAR sequence number for the control block. | 2 |

Table 212. End Secure Environment Select Event (continued)

| Parameter list item | Content | Length in bytes |
|---|---|---|
| VAR_FLAG | Result flags: 1. Select for Read. 2. Select for Writer. | 2 |
| VAR_RC | Return code. | 4 |
| VAR_RSN | Reason code. | 4 |

## Control Blocks and DSECTS for Event Recording

The following tables list the DSECTS and control blocks that are referenced by various events. Each table describes the parameter list contents of control blocks that are used to record an IMS Connect event and associated data.

## Event Recording Parameter List (ERPL)

This control block is used to record an IMS Connect event and associated data to an event-recording log. The parameter list contains mandatory and optional fields. The content and usage of the list arguments is dependent on the event being recorded. The DSECT name is HWSERPL. The contents of the ERPL which are pointed to by HWSTECL0 are shown in Table 213.

Table 213. Event Recording Parameter List (ERPL) Pointed to by HWSTECL0

| Element | Length | Usage and Meaning |
|---|---|---|
| TOKEN | 4 | Address of the token for event recording. This is the token returned in the EICB when event recording was initialized. Required. |
| EVENT_NUMBER | 2 | The number associated with the event being recorded. Required. |
| EVENT_KEY | 8 | The event key that is associated with the event being recorded. Required. |
| DATA_ADDR_COUNT | 2 | Count of the number of EVENT_DATA_ADDR entries in the parameter list. A count of 0 indicates that no entries are present. Required, but can be 0. |
| VAR_DATA_LL | 2 | Length of the variable data element. The variable data length does not include this length field. A length of 0 indicates that no variable data is present. Required, but can be 0. |
| EVENT_DATA_ADDR | 4 | The address of a data element that begins with a two-byte length field. The parameter list can contain any number of element addresses. The number of element addresses is contained in DATA_ADDR_COUNT. Optional. |
| VAR_DATA | VAR | A variable length field containing event dependent data. The length of the data element is defined by VAR_DATA_LL. Only one variable data element can be present in the parameter list. Optional. |

## Event Interface Control Block (EICB)

This control block links IMS Connect and the trace and event recording module, HWSTECL0. The block is formatted by IMS Connect and passed to HWSTECL0 with the initialization request. The DSECT name is HWSEICB.

The contents of the EICB are shown in Table 214.

*Table 214. EICB Parameter List Contents*

| Element | Length | Usage and Meaning |
|---|---|---|
| EYECATCHER | 4 | Value of EICB identifying this block in working storage. Set by caller. |
| FLAGS | 1 | Interface control flags:<br>1. Event recording is enabled. |
| EVENT_TOKEN | 4 | Address of the token used by the event recording routine. The token must be passed to the event recording routine when an event-recording request is made. |
| EVENT_ADDRESS | 4 | Entry address of event recording routine. |
|  | 4 | Reserved space. |
|  | 4 | Reserved space. |
| MESSAGE_LEN | 2 | Length of the message returned from HWSTELC0 module. |
| MESSAGE_AREA | 120 | An area that can be used by HWSTECL0 to return an informational or error message to IMS Connect. |

## TCP/IP Information Block (TCPIB)

This control block is used to pass information about TCP/IP events to the event recording routine. The block contains a length field that allows the recording routine to capture the block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record based on the length field. The DSECT is HWSTCPIB.

The contents of the TCPIB are shown in Table 215.

*Table 215. TCP/IP Information Block (TCPIB) Contents*

| Element | Length | Usage and Meaning |
|---|---|---|
| LENGTH | 2 | Length of the TCPIP block, including the length of the field. |
| BLOCK_ID | 1 | Block ID = X'01' identifying the block as a TCPIB. |
| VERSION | 2 | The version and release of IMS Connect in the VVRR format. |
| APAR_COUNT | 2 | A sequential count field starting at one and incrementing by one for any APAR changing the format or content of the control block. The number resets to one at each new release. |
| PORT_NUMBER | 2 | The port number associated with the TCP/IP event being recorded. |
| LOCAL_PC_NUM | 4 | The PC number used by the local connection. |

Table 215. TCP/IP Information Block (TCPIB) Contents (continued)

| Element | Length | Usage and Meaning |
|---|---|---|
| SOCKET_NUM | 2 | The socket number associated with the request. This field is redefined to LOCAL_PC_NUM. A 2-byte reserved field follows this field to account for the 4-byte length of LOCAL_PC_NUM. |
| SOCKET_FLAG | 1 | A flag byte identifying information about the socket. 1. Listen socket. 2. Session socket. |
| PORT_FLAG | 1 | A flag byte identifying information about the port. 1. SSL port. 2. Local port. |
| LENGTH_ISSUED | 4 | The length values associated with the read or write command. |
| LENGTH_EXECUTED | 4 | The length value actually executed by the read or write command. |
| LOCAL_SND_LEN | 4 | Overlays LENGTH_ISSUED. For the local interface, the lengths for a local send operation. |
| LOCAL_RCV_LEN | 4 | Overlays LENGTH_EXECUTED. For the local interface, the lengths for a local receive operation. |
| EVENT_DATA | 4 | Data or flag bits or both associated with the event. This data can be unique for each event recording the TCPIB. |
| RETURN_CODE | 4 | Return code associated with the request. |
| TCPIP REASON_CODE | 4 | Reason code received from TCP/IP. |
| LOCAL REASON_CODE | 8 | Reason code received from local interface. |

## Datastore Information Block (DSIB)

This block is used to pass information about Datastore-related events to the event recording routine. The DSIB is also used with the SYSPLEX interface. The block contains a length field that allows the recording routine to capture the block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record. The DSECT name is HWSDSIB.

The contents of the DSIB are shown in Table 216.

Table 216. Datastore Information Block (DSIB) Contents

| Element | Length | Usage and Meaning |
|---|---|---|
| LENGTH | 2 | Length of the DSIB block, including the length of the field. |
| BLOCK_ID | 1 | Block ID = X'02' identifying the block as a DSIB. |
| DS_FLAG | 1 | A flag byte providing information about the DSTOR_NAME field: 1. Name is datastore. 2. Name is SCI. 3. Name is MEMBER. 4. Name is TMEMBER. |
| VERSION | 2 | The version and release of IMS Connect in the VVRR format. |

Table 216. Datastore Information Block (DSIB) Contents (continued)

| Element | Length | Usage and Meaning |
|---|---|---|
| APAR_COUNT | 2 | A sequential count field starting at one and incrementing by one for any APAR changing the format or content of the control block. The number resets to one at each new release. |
| DSTOR_NAME | 16 | Name associated with the datastore. For the SYSPLEX (SCI) interface, this is the SYSPLEX name. The field can also be the name of a MEMBER or TMEMBER. |
| DATA_LEN | 4 | Length associated with a send or receive operation. |
| DATA_ADDR | 4 | Data address if any associated with the event. Currently, only OTMA sends and receives operations. |
| RETURN_CD | 4 | The return code associated with the operation. |
| REASON_CD | 4 | The reason code associated with the operation. |
| TPIPE_NAME | 8 | The TPIPE name associated with the data transfer. |
| CUR_SVTTOKEN | 8 | The SVT Token associated with the request, if any. |

## Security Information Block (SAFIB)

This block is used to pass information about security-related events to the event-recording routine. The block contains a length field that allows the recording routine to capture block information regardless of the content or length. When the block is recorded, the entire block is moved to the event record. The DSECT name is HWSSAFIB.

The contents of SAFIB are shown in Table 217.

Table 217. Security Information Block (SAFIB) Contents

| Element | Length | Usage and Meaning |
|---|---|---|
| LENGTH | 2 | Length of the SAFIB block, including the length of the field. |
| BLOCK_ID | 1 | Block ID = X'03' identifying the block as a SAFIB. |
| VERSION | 2 | The version and release of IMS Connect in the VVRR format. |
| APAR_COUNT | 2 | A sequential count field starting at one and incrementing by one for any APAR changing the format or content of the control block. The number resets to one at each new release. |
| REQUEST_TYPE | 1 | Flag indicating the type of request: 1. Type is VERIFY. 2. Type is FASTAUTH. 3. Type is DELETE. 4. Type is LIST. |
| USERID | 8 | USERID or PASSTICKET associated with the request. |
| CLASS_NAME | 8 | Name of the SAF class associated with the request. |
| RETURN_CODE | 4 | Return code received. |
| REASON_CODE | 4 | Reason code received from the SAF interface. |

## Variable Data Block (VDB)

This block is used to present variable data to the event-recording interface. The block is contained within the event parameter list. The block does not contain a length field. The length of this block is specified in the even parameter lists. This allows the block information to be captured regardless of the content or length. When the block is recorded, the entire block is moved to the event record. The DSECT name is HWSVDBxx, where xx equals the event number.

The contents of the VDB are shown in Table 218.

*Table 218. Variable Data Block (VDB) Contents*

| Element | Length | Usage and Meaning |
|---|---|---|
| VAR_DATA | variable | A set of fields defined as variable data for each event that contains variable data. Each event can have individually defined variable data. |

## DSECTS for Event Recording

The following table lists all the macros that are shipped with IMS Connect to help customize with event recording:

*Table 219. Event Recording Macros Shipped with IMS Connect*

| Macro | Function |
|---|---|
| HWSDSIB | DATASTORE INFORMATION BLOCK |
| HWSEICB | EVENT INITIALIZATION BLOCK |
| HWSERPL | EVENT RECORDING PARAMETER LIST |
| HWSSAFIB | SAF INTERFACE BLOCK |
| HWSTCPIB | TCPIP EVENT INFORMATION BLOCK |
| HWSVDB01 | EVENT 01 VARIABLE DATA BLOCK |
| HWSVDB02 | EVENT 02 VARIABLE DATA BLOCK |
| HWSVDB03 | EVENT 03 VARIABLE DATA BLOCK |
| HWSVDB04 | EVENT 04 VARIABLE DATA BLOCK |
| HWSVDB06 | EVENT 06 VARIABLE DATA BLOCK |
| HWSVDB08 | EVENT 08 VARIABLE DATA BLOCK |
| HWSVDB11 | EVENT 11 VARIABLE DATA BLOCK |
| HWSVDB13 | EVENT 13 VARIABLE DATA BLOCK |
| HWSVDB21 | EVENT 21 VARIABLE DATA BLOCK |
| HWSVDB23 | EVENT 23 VARIABLE DATA BLOCK |
| HWSVDB26 | EVENT 26 VARIABLE DATA BLOCK |
| HWSVDB27 | EVENT 27 VARIABLE DATA BLOCK |
| HWSVDB29 | EVENT 29 VARIABLE DATA BLOCK |
| HWSVDB33 | EVENT 33 VARIABLE DATA BLOCK |
| HWSVDB35 | EVENT 35 VARIABLE DATA BLOCK |
| HWSVDB37 | EVENT 37 VARIABLE DATA BLOCK |
| HWSVDB38 | EVENT 38 VARIABLE DATA BLOCK |
| HWSVDB40 | EVENT 40 VARIABLE DATA BLOCK |
| HWSVDB61 | EVENT 61 VARIABLE DATA BLOCK |

*Table 219. Event Recording Macros Shipped with IMS Connect (continued)*

| Macro | Function |
|---|---|
| HWSVDB62 | EVENT 62 VARIABLE DATA BLOCK |
| HWSVDB69 | EVENT 69 VARIABLE DATA BLOCK |
| HWSVDB70 | EVENT 70 VARIABLE DATA BLOCK |
| HWSVDB71 | EVENT 71 VARIABLE DATA BLOCK |
| HWSVDB72 | EVENT 72 VARIABLE DATA BLOCK |
| HWSVDB81 | EVENT 81 VARIABLE DATA BLOCK |
| HWSVDB82 | EVENT 82 VARIABLE DATA BLOCK |
| HWSVDB83 | EVENT 83 VARIABLE DATA BLOCK |
| HWSVDB84 | EVENT 84 VARIABLE DATA BLOCK |
| HWSVDB85 | EVENT 85 VARIABLE DATA BLOCK |
| HWSVDB86 | EVENT 86 VARIABLE DATA BLOCK |
| HWSVDB87 | EVENT 87 VARIABLE DATA BLOCK |

## Terminating HWSTECL0

To end event recording, IMS Connect calls the event recording routine address in the EICB. The routine is passed to the ERPL, which defines the event and event data. The event number which is passed to the event recording routine corresponds to the Connect Region Termination event. See Table 149 on page 274 for the contents of the parameter list.

When the termination processing for event recording has completed, HWSTECL0 must return to the caller otherwise IMS will hang.

Note: The termination call to HWSTECL0 is made even if the event recording flag in the EICB is not on. If the EICB contains a token and event recording address, the termination call is made so that event recording can terminate the event recording environment.

The event recording termination call can only occur when the caller is executing under the JOBSTEP TCB, the caller is in primary TCB mode, and all tasks as potential event records have terminated.

# Notices

## Programming Interface Information

This publication is intended to help the customer perform the following tasks:
- Plan for and design the installation of IMS Connect.
- Install and operate IMS Connect.
- Diagnose and recover from IMS Connect system problems.
- Write an IMS Connect client.
- Use IMS Connect with IMS Connector for Java.

The *IMS Connect Guide and Reference* primarily documents General-use Programming Interfaces and Associated Guidance Information provided by IMS Connect.

General-use programming interfaces allow the customer installation to write programs that obtain the services of IMS Connect.

However, the *IMS Connect Guide and Reference* also documents Product-Sensitive Programming Interfaces and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by IMS Connect.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of IMS and IMS Connect. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

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 and Associated Guidance Information...

Diagnosis, Modification or Tuning Information is provided to help the customer installation diagnose, modify, or tune IMS Connect.

**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: Diagnosis, Modification or Tuning Information....

## Trademarks

# Bibliography

This bibliography includes all the publications cited in this book.

- *External Security Interface (RACROUTE) Macro Reference for MVS*, GC28-1366
- *IMS Version 8 Customization Guide*, SC27-1294
- *IMS Version 8 Diagnosis Guide and Reference*, LY37-3742
- *IMS Version 8 Master Index and Glossary*, SC27-1300
- *IMS Version 8 Open Transaction Manager Access Guide*, SC27-1303
- *IPv6 Network and Application Design Guide*, SC31-8885
- *MVS/ESA Programming: Assembler Services Guide*, GC28-1466
- *MVS/ESA Programming: Authorized Assembler Services Guide*, GC28-1467
- *MVS/ESA SP Authorized Assembler Reference*, GC28-1650
- *MVS/ESA System Commands*, GC28-1442
- *OS/390 MVS Authorized Assembler Services Reference, Volume 3 (LLA-SDU)*, GC28-1766
- *OS/390 MVS Initialization and Tuning Reference*, SC28-1752
- *Program Directory for IBM IMS Connect for OS/390*, GI10-8275
- *TCP/IP Application Programming Interface Reference*, SC31-7187
- *z/OS: Security Server RACF Command Language Reference*, SA22-7687
- *z/OS: Security Server RACF Security Administrators Guide*, SA22-7683
- *z/OS System Secure Sockets Layer Programming*, SC24-5901
- *z/OS UNIX System Services Planning*, GA22-7800

This bibliography lists all of the information in the IMS Version 9 library.

## IMS Version 9 Library

| Title | Acronym | Order number |
|---|---|---|
| *IMS Version 9: Administration Guide: Database Manager* | ADB | SC18-7806 |
| *IMS Version 9: Administration Guide: System* | AS | SC18-7807 |
| *IMS Version 9: Administration Guide: Transaction Manager* | ATM | SC18-7808 |
| *IMS Version 9: Application Programming: Database Manager* | APDB | SC18-7809 |
| *IMS Version 9: Application Programming: Design Guide* | APDG | SC18-7810 |
| *IMS Version 9: Application Programming: EXEC DLI Commands for CICS and IMS* | APCICS | SC18-7811 |
| *IMS Version 9: Application Programming: Transaction Manager* | APTM | SC18-7812 |
| *IMS Version 9: Base Primitive Environment Guide and Reference* | BPE | SC18-7813 |
| *IMS Version 9: Command Reference* | CR | SC18-7814 |
| *IMS Version 9: Common Queue Server Guide and Reference* | CQS | SC18-7815 |
| *IMS Version 9: Common Service Layer Guide and Reference* | CSL | SC18-7816 |
| *IMS Version 9: Customization Guide* | CG | SC18-7817 |
| *IMS Version 9: Database Recovery Control (DBRC) Guide and Reference* | DBRC | SC18-7818 |
| *IMS Version 9: Diagnosis Guide and Reference* | DGR | LY37-3203 |
| *IMS Version 9: Failure Analysis Structure Tables (FAST) for Dump Analysis* | FAST | LY37-3204 |
| *IMS Version 9: IMS Connect Guide and Reference* | CT | SC18-9287 |
| *IMS Version 9: IMS Java Guide and Reference* | JGR | SC18-7821 |
| *IMS Version 9: Installation Volume 1: Installation Verification* | IIV | GC18-7822 |
| *IMS Version 9: Installation Volume 2: System Definition and Tailoring* | ISDT | GC18-7823 |
| *IMS Version 9: Master Index and Glossary* | MIG | SC18-7826 |
| *IMS Version 9: Messages and Codes, Volume 1* | MC1 | GC18-7827 |
| *IMS Version 9: Messages and Codes, Volume 2* | MC2 | GC18-7828 |

| Title | Acronym | Order number |
|---|---|---|
| *IMS Version 9: Open Transaction Manager Access Guide and Reference* | OTMA | SC18-7829 |
| *IMS Version 9: Operations Guide* | OG | SC18-7830 |
| *IMS Version 9: Release Planning Guide* | RPG | GC17-7831 |
| *IMS Version 9: Summary of Operator Commands* | SOC | SC18-7832 |
| *IMS Version 9: Utilities Reference: Database and Transaction Manager* | URDBTM | SC18-7833 |
| *IMS Version 9: Utilities Reference: System* | URS | SC18-7834 |

## Supplementary Publications

| Title | Order number |
|---|---|
| *IMS Connector for Java 2.2.2 and 9.1.0.1 Online Documentation for WebSphere Studio Application Developer Integration Edition 5.1.1* | SC09-7869 |
| IMS Version 9 Fact Sheet | GC18-7697 |
| *IMS Version 9: Licensed Program Specifications* | GC18-7825 |

## Publication Collections

| Title | Format | Order number |
|---|---|---|
| IMS Version 9 Softcopy Library | CD | LK3T-7213 |
| IMS Favorites | CD | LK3T-7144 |
| Licensed Bill of Forms (LBOF): IMS Version 9 Hardcopy and Softcopy Library | Hardcopy and CD | LBOF-7789 |
| Unlicensed Bill of Forms (SBOF): IMS Version 9 Unlicensed Hardcopy Library | Hardcopy | SBOF-7790 |
| OS/390 Collection | CD | SK2T-6700 |
| z/OS Software Products Collection | CD | SK3T-4270 |
| z/OS and Software Products DVD Collection | DVD | SK3T-4271 |

## Accessibility Titles Cited in This Library

| Title | Order number |
|---|---|
| *z/OS V1R1.0 TSO Primer* | SA22-7787 |
| *z/OS V1R5.0 TSO/E User's Guide* | SA22-7794 |
| *z/OS V1R5.0 ISPF User's Guide, Volume 1* | SC34-4822 |

# Index

Program Number:  5655-J38

Printed in USA

SC18-9287-00

Spine information:

IMS    IMS Connect Guide and Reference    Version 9