IMS TCP/IP OTMA Connection User's Guide and Reference

itocug-0002-02

Note

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

ITOC User’s Guide

Contents

Notices

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

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

Trademarks

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

Database 2	IBM
IMS	IMS/ESA
MVS	MVS/SP
OS/390	RACF
VTAM	

Windows and Windows NT are registered trademarks of Microsoft Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

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

Chapter 1. Overview of the IMS TCP/IP OTMA Connection

Introduction

The IMS TCP/IP OTMA Connection (IMS TOC) is a TCP/IP server that enables TCP/IP clients to exchange messages with IMS OTMA. As shown in Figure 1, this server provides communication linkages between TCP/IP clients and IMS (datastores). It supports multiple TCP/IP clients accessing multiple datastore resources. The IMS TCP/IP OTMA Connection runs on an MVS or OS/390 platform. For environmental details, see "Chapter 3. How to install and configure the IMS TOC" on page 7.

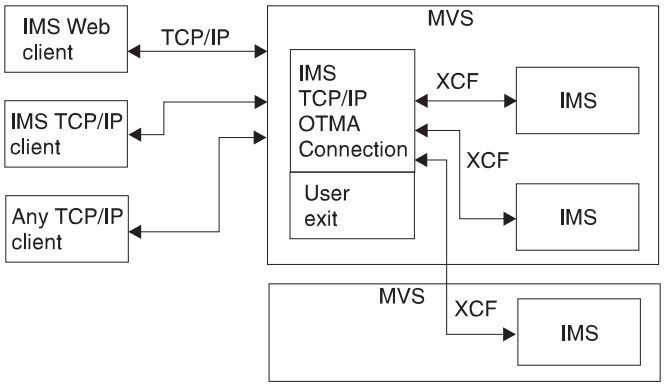


Figure 1. System overview

The IMS TCP/IP OTMA Connection performs router functions between TCP/IP clients and datastores. Request messages received from TCP/IP clients, via TCP/IP connections, are passed to a datastore through XCF sessions. The IMS TCP/IP OTMA Connection receives response messages from the datastore and then passes them back to the originating TCP/IP clients. The IMS TCP/IP OTMA Connection architecture is designed to support IMS Web, but is not limited to working with IMS Web clients.

The IMS TCP/IP OTMA Connection supports TCP/IP clients communicating with socket calls. In order to support any TCP/IP client communicating with a different input data stream format, the IMS TCP/IP OTMA Connection allows user-written programs running in its address space to convert customer message format to OTMA message format before it ships to IMS. The same user-written programs can also convert OTMA message format to customer message format before sending a message back to a client. MVS TCP/IP already has an exit written for general customer use: exit HWSIMSO0. For details, see "Chapter 4. IMS TCP/IP OTMA Connection user exit support" on page 29.

Components

As shown in Figure 2 on page 3, the IMS TCP/IP OTMA Connection consists of three core components:

- TCP/IP communication component (CCC)

  The TCP/IP communication component processes communications between IMS Web servers (and their client workstations) and the IMS TCP/IP OTMA Connection.
- Datastore communication component (DCC)

  The datastore communication component handles communications between the IMS TCP/IP OTMA Connection and the IMS datastores.
- Command component (CMD)

  The command component processes commands received from the MVS console operator. This release supports the following commands:

Command	Description
CLOSEHWS	Terminates the IMS TCP/IP OTMA Connection (HWS).
OPENDS	Starts a communication session between the HWS and the specified datastore (IMS).
OPENPORT	Reestablishes the communication session between the HWS and the TCP/IP network through the specified port address.
SETRACF	Turns on and off the RACF flag.
STOPCLNT	Immediately terminates the communication session between the HWS and the specified client using the specified port address.
STOPDS	Immediately terminates the communication session between the HWS and the specified datastore (IMS).
STOPPORT	Immediately terminates the communication session between the HWS and the TCP/IP network that uses the specified port address.
VIEWDS	Displays the current status of the specified datastore (IMS).
VIEWHWS	Displays the current status of OTMA Connection.
VIEWPORT	Displays the current status of the communication session between the HWS and the specified port.

For details of these commands, see “IMS TCP/IP OTMA Connection commands” in the *IMS TCP/IP OTMA Connection Programmer’s Reference*.

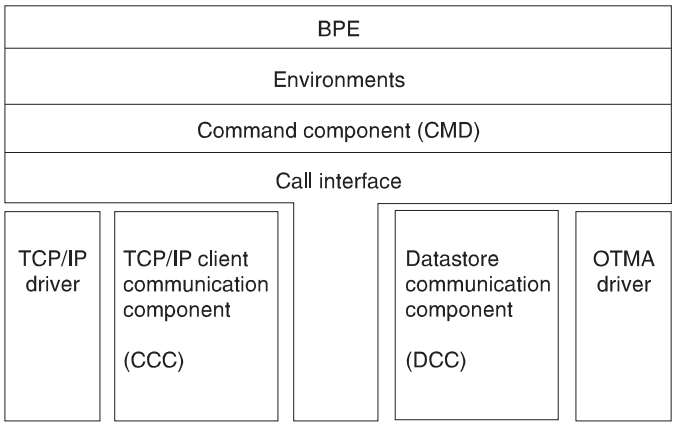


Figure 2. IMS TCP/IP OTMA Connection component layout

In addition to these core components, the IMS TCP/IP OTMA Connection uses a communication driver facility to isolate the core components from the communication software. The TCP/IP driver is used to communicate with IMS Web servers (and their client workstations) using the TCP/IP communications protocol. The IMS OTMA driver is used to communicate with the datastores (IMSs) using the IMS OTMA communications protocol. Communication between components takes place via the call interface service. The call interface provides the encapsulation and isolation of structures between the components. Each IMS TCP/IP OTMA Connection 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 base primitive environment (BPE) provides the following base services for the IMS TCP/IP OTMA Connection:

- Environment
- Storage
- Serialization
- Tracing

What's new in release 2.1.0

The following enhancements have been made for release 2.1.0:

- Added conversational support
  - Added ability for user to retain message continuity from a client
  - Added new message HWSP1495E
- Enhanced security
  - Added new command, `SETRACF`
  - Added new error message, HWSP1500E

- Added a new exit routine, HWSWEB00. This exit is required for IMS Web input messages. This exit allows the customer to select the security exit.
- Updated installation and configuration
- Added a trace dump example in the messages and codes section
- Added a new exit routine that is supported by IMS TOC (HWSIMSO0)
- Removed 32-KB message restriction
- Separated client ID and LTERM override name
- Changed input message format to allow for the removal of the 32-KB message restriction and separation of LTERM and client ID
- Changed output format from the exit routine on a READ to improve performance
- Changed message structure from the client

  Attention: You must regenerate and recompile any projects generated with earlier versions of IMS Web!
- Added support for variable-length OTMA prefixes
- Added support for variable-length IRM headers

Attention:

- Because the message structure from the client has changed (to provide new options, functions, HWEXPRM fields, variable-length OTMA headers, and separation of client ID and LTERM name), be sure and review the new message structure.
- If you modified the HWSSMPL0 exit routine during installation, you need to make those same changes to the new HWSSMPL0 exit routine.

What's new in release 2.1.1

The following enhancement has been made for release 2.1.1:

- Added support for TCP/IP 3.4 (requires TCP/IP APAR PQ13154 to be installed on TCP/IP 3.2 and TCP/IP 3.4)

What's new in release 2.1.2

The following enhancements have been made for release 2.1.2:

- Reorganized the user's guide and corrected some technical information
- Added user exit reason and return codes to messages and codes

Reader comment form

A reader comment form is included with the hardcopy version of this book. If you would like to fill out the online version of the reader comment form, please click here or go to http://www.software.ibm.com/data/rcf/.

Chapter 2. IMS TCP/IP OTMA Connection prerequisites

This section describes the hardware and software prerequisites for IMS TCP/IP OTMA Connection.

Hardware requirements

- Host processor capable of running IMS/ESA Transaction Manager Version 5 or later and IMS/ESA Database Manager Version 5 or later

Software requirements

- MVS/System Product Version 4.2 or later
- IMS/ESA Transaction Manager Version 5 or later
- IMS/ESA Database Manager Version 5 or later
- TCP/IP for MVS Version 3.2 or 3.4 (TCP/IP for MVS Version 3.3 does not support IMS TOC 2.1) and APAR PQ13154
- Resource Access Control Facility (RACF) Version 1.9.2 or equivalent product

Note: This document uses the term RACF when referring to RACF or an equivalent product.

Chapter 3. How to install and configure the IMS TOC

The IMS TCP/IP OTMA Connection (IMS TOC) provides the following components that enable TCP/IP clients to exchange messages with IMS:

IMS TOC

An MVS application program that provides the following services:

- Communication to IMS Web Runtime or TCP/IP clients via TCP/IP connections
- Communication to IMS via OTMA connections

Base primitive environment (BPE)

A system-service component that supports IMS and IMS TOC. This component is supplied only for systems running IMS 5.1, which does not include BPE; systems running IMS 6.1 should use the BPE that is installed with IMS 6.1.

Installing and configuring IMS TOC for IMS 5.1 or IMS 6.1 require the following considerations.

Requirement for IMS 5.1:

For IMS 5.1 environments, HWSJCLIN must be executed to link-edit the IMS TOC load modules (HWSxxxxx) into an IMS TOC RESLIB. Place the IMS TOC modules in a separate IMS TOC RESLIB.

In an IMS 5.1 environment, HWSBPEIN must be executed to link-edit the base primitive environment (BPE) load modules (BPExxxxx) into a separate BPE RESLIB. Placing the IMS TOC and BPE modules in two separate RESLIBs in an IMS 5.1 environment will make it easier to move to IMS 6.1.

When moving from an IMS 5.1 environment to an IMS 6.1 environment, you will need to replace the BPE RESLIB JCL statement with the IMS 6.1 RESLIB that contains the BPE modules.

In an IMS 5.1 environment, BPE maintenance will be part of IMS TOC maintenance.

Requirement for IMS 6.1:

For IMS 6.1 environments, HWSJCLIN must be executed to link-edit the IMS TOC load modules (HWSxxxxx) into an IMS TOC RESLIB. Place the IMS TOC modules in a separate IMS TOC RESLIB.

The HWSBPEIN must NOT be executed.

In an IMS 6.1 environment, IMS TOC uses the base primitive environment (BPE) load modules in the IMS 6.1 RESLIB. Therefore, you must concatenate the IMS TOC RESLIB and the IMS 6.1 RESLIB, which contains the BPE modules, in the startup JCL for your IMS TOC region.

In an IMS 6.1 environment, BPE maintenance will be part of the normal IMS 6.1 maintenance.

In addition to these two components (IMS TOC and BPE), the following are included:

- A sample user exit, HWSSMPL0.
- IMS TOC's associated files, HWSIMSCB, HWSIMSEA, HWSEXPRM, and HWSOMPFX. The associated files can be used with the IMS client for Java™.
- Product installation and release documentation files.

This section describes the following:

Installing the IMS TCP/IP OTMA Connection

Attention: IMS TOC and the IMS client for Java sample program are packaged using Info-ZIP's compression utility to create a self-extracting executable archive (zip) file. This program uses UnZip internally to extract the archived files. Info-ZIP's software (Zip, UnZip and related utilities) is free and can be obtained as source code or executables from various anonymous-ftp sites, including ftp.uu.net:/pub/archiving/zip/* or from the Web at http://www.cdrom.com/pub/infozip/.

To install the IMS TCP/IP OTMA Connection, perform the following actions:

1. IMS TCP/IP OTMA Connection (IMS TOC) 2.1.1 is available from the Internet. Go to the IMS TOC home page (http://www.software.ibm.com/data/ims/about/imstoc/) and select the Downloads icon.
2. From the IMS TOC download page, select a download link for IMS TCP/IP OTMA Connection.
3. Complete the customer registration form and click Submit Survey.
4. After reading the license agreement, click Accept this License Agreement.
5. In the section "IMS TCP/IP OTMA Connection and IMS Client for Java", select the link: Download for IMS TCP/IP OTMA Connection v.r.m. This connection downloads to your workstation the file, HWSMH*vrm*.exe (where *vrm* represents the version, release, and modification of IMS TOC). Using the Save dialog box, save the downloaded file to a temporary directory on a Windows or OS/2 workstation where it will be stored and its contents expanded in the next step.
6. Run the downloaded .exe file, which is a self-extracting, compressed file, by entering `HWSMH`*`vrm`*`.exe` at a Windows or OS/2 command prompt (where again *vrm* represents the version, release, and modification of IMS TOC) to expand the contents of HWSMH*vrm* into the current directory. You can enter `HWSMHvrm.exe -t` to check the integrity of the zip file without actually expanding the files.

   Note: If HWSMH211 fails with the following error message:

   ```
   Usage : OS2 /P /C
   ```

   Try the following:

   ```
   OS/2 /P HWSMH211.exe /C
   ```

Execution of this file generates the sequential files GENLIBB.BIN, BPELOAD.BIN, and LOAD.BIN, along with HTML and text versions of both these installation instructions and a list of changes to the contents of HWSMH*vrm*.exe and two .GIF files that contain graphics used in the HTML version of these installation instructions. Also contained in HWSMH*vrm*.exe is another self-extracting, compressed file, JAVASAMP.exe.

GENLIBB.BIN contains:

- A sample link-edit job, HWSJCLIN, that you use to link-edit the IMS TCP/IP OTMA Connection load modules into a RESLIB
- HWSUMSG, which can be used to map a user-defined message format
- A sample user exit (HWSSMPL0) for use with IMS TOC clients
- Two macros (HWSEXPRM and HWSOMPFX) that can be used by that user exit (HWSSMPL0) or any IMS TCP/IP OTMA Connection user exits that you write
- The IMS Web user exit (HWSWEB00), for use with IMS Web clients, to allow you to issue a security function

BPELOAD.BIN contains the BPE load modules, which must be link-edited and copied into a separate IMS BPE RESLIB.

Recommendation: Use a separate load library for BPE in order to simplify migration from IMS 5.1 to IMS 6.1 when you are using IMS TOC on IMS 5.1.

LOAD.BIN contains the IMS TCP/IP OTMA Connection load modules, which must be link-edited and copied into the RESLIB that you plan to use for IMS TCP/IP OTMA Connection.

The sample user exit (HWSSMPL0) contains code to translate messages into the format required by IMS Version 5 Open Transaction Manager Access (OTMA) and is used in conjunction with two macro files, HWSEXPRM and HWSOMPFX.

JAVASAMP.exe contains the IMS client for Java sample program, and HTML and text versions of both the installation instructions and a list of changes to the sample client for Java. See “Installing the IMS client for Java sample program” in the *IMS TCP/IP OTMA Connection Programmer’s Reference* for installation instructions.

Attention: The directory where JAVASAMP.exe is expanded must be located on an OS/2 or Windows drive that supports the long file names used for the expanded files.

7. If your file transfer tool will not do so automatically, allocate three fixed block record format, 80-byte record length sequential data sets, IMSHWS.SEQ.GENLIBB, IMSHWS.SEQ.LOAD, and IMSHWS.SEQ.BPELOAD, into which the sequential data sets will be transferred (see the next step in this procedure). Note that you may use names other than IMSHWS.SEQ.GENLIBB, IMSHWS.SEQ.LOAD, and IMSHWS.SEQ.BPELOAD for these data sets. If you are using IBM’s Personal Communications Workstation Program for Windows 95 (and NT 4), you may set up the file transfer options to automatically create these data sets using the required parameters.

   In the **Setup/Define Transfer-Types** option under the **Transfer** pulldown, enter a name in the **Transfer-Type Names** entry field, select **Fixed** for the Record

Format, enter 80 for the Logical Record Length, and leave all other fields blank and all other selections unchecked or unselected. Save this Transfer-Type by pressing the **Add** push button and then click **OK**. Be sure to use this Transfer-Type in the next step when sending the sequential files to the host.

Restriction: The format for these data sets must be as follows:

```
Organization  . . . : PS  (physical sequential)
Record format . . . : FB  (fixed block)
Record length . . . : 80  (bytes)
```

Note that it is not necessary to specify a block size.

8. Using PCOMM or FTP from a workstation command prompt, upload the extracted and uncompressed files using binary transfer mode into the sequential data sets allocated in the MVS environment in the previous step. Note that the PCOMM, Send File to Host.., function might not work correctly within ISPF. If you encounter a problem when using PCOMM's Send File to Host.. function while the emulator is in ISPF, exit ISPF so that the emulator is at a TSO READY prompt and try sending the file again.
9. Convert the sequential data sets to partitioned data sets by issuing the following commands from the ISPF 6 (ISPF Command Shell) prompt or from the TSO READY prompt:

```
RECEIVE INDSN(IMSHWS.SEQ.GENLIBB)
DSNAME(IMSHWS.GENLIBB)
RECEIVE INDSN(IMSHWS.SEQ.LOAD)
DSNAME(IMSHWS.LOAD)
RECEIVE INDSN(IMSHWS.SEQ.BPELOAD)
DSNAME (IMSHWS.BPELOAD)
```

Notes:

a. `RECEIVE` is used to restore the data sets because the sequential data sets were created using the `TRANSMIT` command. `TRANSMIT` converted the original partitioned data sets into sequential data sets. `RECEIVE` is used to convert them from sequential data sets back to their original partitioned organization.

b. You are prompted for the DSNAMEs.

c. The input data set names (IMSHWS.SEQ.GENLIBB, IMSHWS.SEQ.LOAD, and IMSHWS.SEQ.BPELOAD in the previous example) are the names of the data sets referred to in steps 7 through 9 of this procedure.

d. MVS prompts you for the restore data set names (IMSHWS.GENLIBB, IMSHWS.LOAD, and IMSHWS.BPELOAD in the previous example, or names of your choice) after you have entered the `RECEIVE INDSN(IMSHWS.SEQ.xxx)`commands. It is not necessary to allocate the restore data sets. If they do not already exist, they are created with the proper formats.

   Attention: If the restore data sets specified are existing data sets, any existing members are overwritten by like-named members of the input data sets. Therefore, if the same receive data sets are used for successive versions of the IMS TCP/IP OTMA Connection, the previous versions of the HWSJCLIN and HWSBPEIN members will be overwritten during the receive of the new GENLIBB data set and any customization of HWSJCLIN and/or HWSBPEIN will be lost!

e. If IMSHWS.GENLIBB is a temporary data set, copy the members of IMSHWS.GENLIBB to the correct data set. (To receive the GENLIBB members directly into your production PROCLIB data set, specify the correct PROCLIB in place of IMSHWS.GENLIBB.)

10. Modify the HWSJCLIN in the GENLIBB data set to correctly link-edit the load modules for the IMS TCP/IP OTMA Connection for your installation. Change the //LOAD DD card to point to the LOADLIB where the IMS TCP/IP OTMA Connection is installed. Then change the //SYSLMOD DD card to point to your RESLIB. Ensure that your RESLIB has enough directory block space (see “Recommended allocations for the required libraries” on page 22). Submit the modified HWSJCLIN, which then builds the RESLIB for you.
11. Modify the HWSBPEIN in the GENLIBB data set to correctly link-edit the load modules for BPE for your installation. Change the //LOAD DD card to point to the LOADLIB where this copy of BPE is installed. Then change the //SYSLMOD DD card to point to your RESLIB. Ensure that your RESLIB has enough directory block space (see “Recommended allocations for the required libraries” on page 22). Submit the modified HWSBPEIN, which then builds the RESLIB for you.
12. Define the IMS TCP/IP OTMA Connection environment for IMS Web, as described in “Defining the IMS TCP/IP OTMA Connection environment”.

Defining the IMS TCP/IP OTMA Connection environment

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

As the following IMS TOC startup JCL statements show, both IMS TOC and BPE have configuration members:

```
//HWS       PROC  RGN=4096K,SOUT=A,
//             BPECFG=BPECFGHT,
//             HWSCFG=HWSCFG00
//*
//*********************************************************************
//*  BRING UP AN IMS TCP/IP OTMA CONNECTION SYSTEM                    *
//*********************************************************************
//STEP1    EXEC PGM=HWSHWS00,REGION=&RGN,TIME=1440,
//              PARM='BPECFG=&BPECFG,HWSCFG=&HWSCFG'
//STEPLIB  DD   DSN=HWS.RESLIB,DISP=SHR
//         DD   DSN=BPE.RESLIB,DISP=SHR
//PROCLIB  DD   DSN=USER.PROCLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
```

Configuring IMS TOC

IMS TOC supports communication between one or more TCP/IP clients and IMS systems. IMS TOC 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 and distribute the client request to the IMSs (datastores).

To configure IMS TOC, perform the following actions:

1. Authorize the Application Program Family (APF).

2. Update the Program Properties Table (PPT) in MVS/ESA. Updating the PPT allows IMS TOC to run in authorized supervisor state and in key 7.
3. Create an IMS TOC configuration member to hold the configuration statements that IMS TOC uses during initialization.

The following sections describe these actions in more detail.

Authorizing IMS TOC and BPE to the APF

The resident library in which the IMS TOC (HWS...) and BPE (BPE...) modules reside must be authorized to the APF. Create and run a JCL job that authorizes this RESLIB to the APF.

Updating the MVS PPT

Because IMS TOC 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 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       */
```

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 TOC configuration member

Specify the environment for IMS TOC as a member in your PROCLIB data set. IMS TOC 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 TOC startup. Specify the member name to use in the HWSCFG= parameter of the IMS TOC startup JCL (see the previous IMS TOC startup JCL example on page 11).

IMS TOC configuration statement parameters: You specify values for some of the parameters that define the way in which IMS TOC is to communicate with TCP/IP and IMS OTMA in the IMS TOC configuration member. The IMS TOC configuration member contains three types of configuration statements: IMS TOC, TCPIP, or DATASTORE. Because the configuration member must be in a dataset 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 TOC.

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

id The HWS name (TPIPE name), which:

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

racf Provide the RACF user identification and verification using the password and user ID provided from the Web or a user exit routine. Set it to yes or no as follows:

- Y
- N (this is the default)

TCPIP Specify only one TCP/IP.

The TCPIP statement keyword parameters are as follows:

hostname
A 1- to 8-alphanumeric character field set to the name of the TCP/IP host.

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.

portid A 1- to 8-character decimal field set to the port number or numbers that will bind to the socket. You can define more than one port as `PORTID=(9999,8888,7777)` to a maximum of 10. Port numbers must be within the range 1 to 65535 and must be selected so as not to conflict with other ports in the TCP/IP domain.

exit A 1 to 8 alphanumeric character field set to the name of the TCP/IP user exit 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 15. These exits support users other than IMS Web Runtime to use OTMA linkage through the IMS TOC to IMS. Do not include HWSWEB00 in the `EXIT=` list. IMS TOC will load the HWSWEB00 exit because it is the only valid exit for IMS Web.

DATASTORE
To access IMS OTMA, specify each datastore with which the IMS TOC communicates via IMS OTMA.

The DATASTORE statement keyword parameters are as follows:

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 Web clients, the ID must match the generated (or user generated/modified) CGI source file. For non-IMS Web clients, the ID must match the datastore ID that is placed in the IRM that is sent to IMS TOC.

group The XCF group name for the IMS OTMA. IMS TOC uses this value to join the appropriate XCF group(s). Because IMS TOC 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

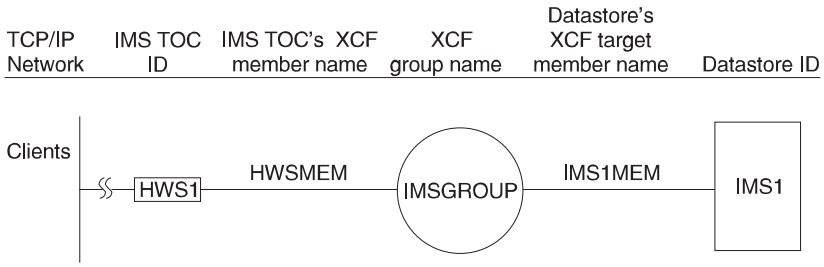
IMS (*GRNAME*) in the IMS startup JCL (for example, ″OTMA=Y,***GRNAME***=&***GROUP***,USERVAR=&MEMBER″,...). Each IMS TOC can join any number of groups

member
The XCF member name that identifies IMS TOC in the XCF group specified by the &GROUP parameter. This name is the XCF name that IMS uses to communicate with IMS TOC in that XCF group. This XCF member name for IMS TOC 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 TOC 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 in different ways, depending on the version of IMS that you are using and what level of service you have applied. In IMS 5.1 (without PQ12917 applied), the IMS startup parameters must include "...,OTMA=Y,GRNAME=&GROUP,***USERVAR***=&***TMEMBER***,...". In IMS 6.1 (without PQ10195 applied), USERVAR is replaced by APPLID ("...,OTMA=Y,GRNAME=&GROUP,***APPLID***=&***TMEMBER***,..."). With PQ12917 applied to IMS 5.1 or PQ10195 applied to IMS 6.1, USERVAR is replaced by OTMANM (″...,OTMA=Y,GRNAME=&GROUP, ***OTMANM***=&***TMEMBER***,...″). Each datastore definition within an IMS TOC configuration member must contain a unique tmember name.

See the following figure for an example of a simple system configuration.

Example 1 (simple) system diagram



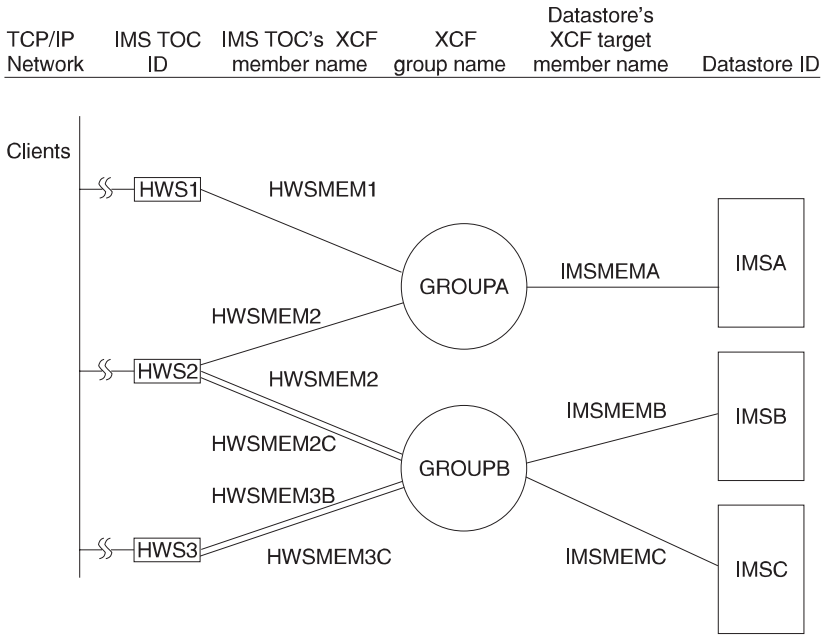
- In the following IMS TOC configuration member, the IMS TOC ID is defined as `HWS1`. This IMS TOC is configured to include the ports defined for TCP/IP communications and the IMS OTMA group and member names for communication with IMS.
- The TCP/IP configuration defines the HOSTNAME as `MVSTCPIP`, the RACFID as `RACFID`, the PORTID as `9999`, and the EXIT as `EZAEXIT`.
- The datastore configuration defines the ID as `IMS1`, the GROUP as `IMSGROUP`, the MEMBER as `HWSMEM`, and the TMEMBER as `IMS1MEM`.

```
************************************************
* IMS TOC EXAMPLE 1 CONFIGURATION FILE
************************************************
```

```
HWS (ID=HWS1,RACF=N)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),EXIT=(EZAEXIT))
DATASTORE  (ID=IMS1,GROUP=IMSGROUP,MEMBER=HWSMEM,TMEMBER=IMS1MEM)
```

See the following figure for a more complex system configuration.

Example 2 (complex) system diagram



- In this example, three IMS TOC's are configured. Each IMS TOC has its own configuration member.
- Each IMS TOC 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 TOC configuration member, the XCF member name for that IMS TOC must be unique in each DATASTORE statement. However, if the datastores are members of different XCF groups, the XCF member names may be the same for different datastores within a single IMS TOC configuration member. For example, observe that the XCF member name for IMS TOC in the IMSA and IMSB DATASTORE statements in the HWS2 configuration member in the configuration example below, HWSMEM2, is the same for both DATASTORE statements because the IMSA and IMSB datastores are members of different XCF groups, GROUPA and GROUPB, respectively. Note that these member names could have been made unique, for example, HWSMEM2A and HWSMEM2B, but it is not necessary to do so. However, the XCF member names for IMS TOC 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)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),EXIT=(EZAEXIT))
DATASTORE          (ID=IMSA,GROUP=GROUPA,MEMBER=HWSMEM1,TMEMBER=IMSMEMA)
************************************************
```

```
************************************************
* HWS EXAMPLE 2 CONFIGURATION MEMBER FOR HWS2
************************************************
HWS (ID=HWS2,RACF=N)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9998),EXIT=(EZAEXIT))
DATASTORE          (ID=IMSA,GROUP=GROUPA,MEMBER=HWSMEM2,TMEMBER=IMSMEMA)
DATASTORE          (ID=IMSB,GROUP=GROUPB,MEMBER=HWSMEM2,TMEMBER=IMSMEMB)
DATASTORE          (ID=IMSC,GROUP=GROUPB,MEMBER=HWSMEM2C,TMEMBER=IMSMEMC)
************************************************

************************************************
* HWS EXAMPLE 2 CONFIGURATION MEMBER FOR HWS3
************************************************
HWS (ID=HWS3,RACF=Y)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9997),EXIT=(EZAEXIT))
DATASTORE          (ID=IMSB,GROUP=GROUPB,MEMBER=HWSMEM3B,TMEMBER=IMSMEMB)
DATASTORE          (ID=IMSB,GROUP=GROUPB,MEMBER=HWSMEM3C,TMEMBER=IMSMEMC)
************************************************
```

Defining IMS TOC security

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

If the datastore (which is IMS) is RACF protected, you have to start IMS TOC as a job with the JOB card specifying a valid *USERID* in order to make the connection from IMS TOC to IMS, or you can use the RACF stored task table. The *USERID=&userid* parameter specified in the JOB card of the IMS TOC job JCL is used as the security vehicle to ensure IMS TOC 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 “IMS/ESA Open Transaction Manager Access Reference,” the section dealing with security for OTMA.

Configuring base primitive environment (BPE)

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

This section describes how to change or modify the configuration data set member and includes some examples.

Changing the BPE configuration member

To change the settings, you can modify a member of the PDS that the PROCLIB DD card specifies. You select the member to use in the BPECFG= parameter of the IMS TOC startup JCL.

The following example shows a BPE configuration member and the statement parameters. In this example, the TCP/IP to IMS trace includes entries for all component events.

```
*********************************************************************
* CONFIGURATION FILE FOR BPE WITH HWS                               *
*********************************************************************
```

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

#
# DEFINITIONS FOR BPE SYSTEM TRACES
#

TRCLEV=(AWE,LOW,BPE)                  /* AWE SERVER TRACE          */
TRCLEV=(CBS,MEDIUM,BPE)               /* CONTROL BLK SRVCS TRACE   */
TRCLEV=(LATC,LOW,BPE)                 /* LATCH TRACE               */
TRCLEV=(DISP,HIGH,BPE,PAGES=12)       /* DISPATCHER TRACE WITH 12  */
                                      /* PAGES (48K BYTES)         */
TRCLEV=(SSRV,HIGH,BPE)                /* GEN SYS SERVICES TRACE    */
TRCLEV=(STG,MEDIUM,BPE)               /* STORAGE TRACE             */

#
# DEFINITIONS FOR HWS TRACES
#

TRCLEV=(CMDT,HIGH,HWS)                /* HWS COMMAND TRACE         */
TRCLEV=(ENVT,HIGH,HWS)                /* HWS ENVIRONMENT TRACE     */
TRCLEV=(HWSW,HIGH,HWS)                /* SERVER TO HWS TRACE       */
TRCLEV=(OTMA,HIGH,HWS)                /* HWS COMM DRIVER TRACE     */
TRCLEV=(HWSI,HIGH,HWS)                /* HWS TO IMS OTMA TRACE     */
TRCLEV=(TCPI,HIGH,HWS)                /* HWS COMM DRIVER TRACE     */
```

As shown in the previous example of a BPE configuration member, you can specify parameters for LANG and TRCLEV.

Each BPE configuration statement begins with `LANG=` or `TRCLEV=`.

LANG You can specify the language to be used for message text. The default for LANG is ENU (U.S. English), which is the only supported language.

TRCLEV

You can specify different levels of tracing detail for BPE and IMS TOC trace tables. BPE provides several internal trace tables to capture diagnostic information for the services that it provides. In addition, IMS TOC has several trace tables for tracing its functions.

For each trace table type that BPE and IMS TOC support, you can specify one `TRCLEV` keyword and the parameters that control tracing.

The parameters for TRCLEV are as follows:

type Specifies the type of trace table for which you are specifying a tracing level. The *type* parameter is a 1- to 4-character string that indicates the BPE or IMS TOC trace table for which you are specifying a trace.

BPE trace tables

AWE Asynchronous Work Element Services trace table, which traces the activity of internal BPE server processes known as AWE servers. When you specify AWE as the *type*, specify BPE as the *product*.

CBS Control Block Services trace table, which traces requests for control block storage managed by BPE. When you specify CBS as the *type*, specify BPE as the *product*.

DISP Dispatcher trace table, which traces dispatcher activity by use of a sub-dispatching service that IMS TOC uses. When you specify DISP as the *type*, specify BPE as the *product*.

LATC Latch services trace table, which traces internal serialization (latching) calls within the IMS TOC address space. When you specify LATC as the *type*, specify BPE as the *product*.

SSRV General system services trace table, which is used to trace general BPE system service events for services, such as data set handling and printing, that do not have their own specific trace table. When you specify SSRV as the *type*, specify BPE as the *product*.

STG Storage service trace table, which traces storage service requests and module LOAD and DELETE requests made through BPE services. When you specify STG as the *type*, specify BPE as the *product*.

IMS TOC trace tables

CMDT IMS TOC command trace table, which traces command activity. When you specify CMDT as the *type*, specify HWS (for IMS TOC) as the *product*.

OTMA IMS TOC communication driver trace table, which traces internal communication protocol activity (XCF calls). When you specify OTMA as the *type*, specify HWS (for IMS TOC) as the *product*.

TCPI IMS TOC communication driver trace table, which traces communication protocol activity (TCP/IP calls). When you specify TCPI as the *type*, specify HWS (for IMS TOC) as the *product*.

ENVT IMS TOC environment trace table, which traces IMS TOC environment events such as startup and shutdown. When you specify ENVT as the *type*, specify HWS (for IMS TOC) as the *product*.

HWSI IMS TOC to OTMA driver trace table, which traces communication activity between IMS TOC and OTMA drivers. When you specify HWSI as the *type*, specify HWS (for IMS TOC) as the *product*.

HWSW
IMS TOC to TCP/IP driver trace table, which traces communication activity and events between TCP/IP drivers and the IMS TOC. When you specify HWSW as the *type*, specify HWS (for IMS TOC) as the *product*.

level Specifies the volume of trace data recorded in the trace table. You can specify the following levels:

ERROR
Only includes trace entries for error conditions. This is the default trace level for all trace table types listed previously.

HIGH High-volume tracing, which includes entries for all component events, such as every module entered in a call path.

MEDIUM
Medium-volume tracing, which includes entries for key component events and some detailed internal component events, such as a component going into an idle state.

LOW Low-volume tracing, which includes entries for key component events, such as the start of a unit of work (UOW). Use this trace level setting to trace normal IMS TOC operation. If you have operations problems, increase the level of tracing.

NONE No tracing is done for the specified table, even if an error condition occurs. Do not use this level of tracing.

product
Indicates whether the trace table is under the control of BPE or IMS TOC:

- Code BPE for all BPE trace tables
- Code HWS for all IMS TOC trace tables

pages Specifies how many 4-KB pages to allocate for the trace table type. If you omit this parameter, BPE uses its own default value (usually 2 or 4 pages). Increase this value if the trace table wraps too quickly to find problems.

Examples of using TRCLEV

The following example shows a setting for the AWE services trace table:

```
TRCLEV=(AWE,LOW,BPE)
```

In this example, the AWE trace includes entries for key component events.

The following example shows a setting for the CBS services trace table.

```
TRCLEV=(CBS,MEDIUM,BPE)
```

In this example, the CBS trace includes entries for key component events and some internal component events.

The following example shows a setting for the LATC trace table:

```
TRCLEV=(LATC,LOW,BPE)
```

In this example, the LATC trace includes entries for key component events.

The following example shows a setting for the dispatcher trace table:

```
TRCLEV=(DISP,HIGH,BPE,PAGES=12)
```

In this example, the dispatcher trace includes entries for all component events. BPE allocates 12 pages (48 KB) for the trace table.

The following example shows a setting for the general system services trace table:

```
TRCLEV=(SSRV,HIGH,BPE)
```

In this example, the general systems service trace includes entries for all component events.

The following example shows a setting for the storage service trace table:

```
TRCLEV=(STG,MEDIUM,BPE)
```

In this example, the storage service trace includes entries for key component events and some internal component events.

The following example shows a setting for the IMS TOC to OTMA driver trace table:

```
TRCLEV=(HWSI,HIGH,HWS)
```

In this example, the IMS TOC to OTMA driver trace includes entries for all component events.

The following example shows a setting for the IMS TOC to TCP/IP driver trace table:

```
TRCLEV=(HWSW,HIGH,HWS)
```

In this example, the IMS TOC to TCP/IP driver trace includes entries for all component events.

The following example shows a setting for the OTMA activity trace table:

```
TRCLEV=(OTMA,HIGH,HWS)
```

In this example, the OTMA activity trace includes entries for all component events.

The following example shows a setting for the TCP/IP activity trace table:

```
TRCLEV=(TCPI,HIGH,HWS)
```

In this example, the TCP/IP activity trace includes entries for all component events.

The IMS TOC trace requests that the BPE place the trace data into are the IMS TOC trace tables. These trace tables are “incore” tables.

The traces are formatted by the standard IMS BPE formatting services. You must link-edit HWSFTRC0 with an alias of HWSFTnnn (where nnn is the version and release level; for example, 2.1.2 would have the alias of HWSFT210). This link-edit is provided in HWSJCLIN and is link-edited into your own defined RESLIB. Either the IMS TOC RESLIB must be included in the IMS/BPE procedure for formatting the dump or you will have to link-edit the IMS TOC trace formatter module (HWSFTRC0) into the IMS/BPE procedure RESLIB (IMS RESLIB). Following is the include, alias, and name statement:

```
INCLUDE LOAD(HWSFTRC0)          HWS FORMATTED TRACE
ALIAS HWSFT210                  HWS 2.1 FORMATTED TRACE ALIAS
NAME HWSFTRC0(R)
```

The link-edit step must use the parms as defined in HWSJCLIN for step 1 as defined below for IMS TOC:

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

The *IMS TCP/IP OTMA Connection Messages and Codes* book contains the format and content of the trace entries.

Guidelines for IMS TOC

This section covers:

Recoverable IMS transactions

This section contains some scenarios when running recoverable transactions in the IMS TOC 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, none of the output is placed (ENQUEUED) in the IMS queue

Only COMMIT mode = 0 is treated as recoverable; COMMIT mode = 1 is not recoverable. With the use of COMMIT mode = 0, IMS TOC 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 usage of the COMMIT mode is the key. 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 TOC, IMS TOC 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 TOC 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 TOC, IMS TOC sends it to the client, and an ACK from the client will result in the successful completion of the application. This scenario works as it should work.

  The input message is processed by IMS and an output message is sent back to IMS TOC, IMS TOC 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 client must ACK the DFS555 message and go to receive mode to receive the deallocate being sent by IMS through IMS TOC. 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 application, and a reentry of the transaction would be necessary to complete the operation.
- With COMMIT mode = 0 and SYNC LEVEL = CONFIRM:

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

  The input message is processed by IMS and an output message is sent back to IMS TOC, IMS TOC 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. However, because asynchronous output is not yet supported, the customer will need to purge the output message by issuing an IMS `PURGE` command. IMS TOC will in the future, on a subsequent refresh or for a new version, provide a mechanism to retrieve the reenqueued data from the NAK issued by the client or provide a mechanism to determine if any output exists for this client.

Recommendation: To run recoverable transactions in the IMS TOC environment, use COMMIT mode = 0 SYNC LEVEL = CONFIRM, and use a single unique CLIENT_ID for each client that uses COMMIT mode = 0 SYNC LEVEL = CONFIRM. Then use and IMS `PURGE` command to purge the output message.

Recommended allocations for the required libraries

Following are the recommended allocations for the required libraries for IMS TOC. You might want to reduce or modify the recommended allocations for your installation.

Library Name	Number of Blocks	Block Size	Trk3380	Trk3390	Directory Blocks
GENLIBB	400	6160	51	43	50
BPE LOADLIB	200	6144	25	21	50
IMS TOC LOADLIB	183	6233	24	20	50
BPE RESLIB	500	18432	200	170	50
IMS TOC RESLIB	500	18432	200	170	50

Tips on using IMS and IMS TOC commands

This section describes tips on using IMS and IMS TOC commands to determine the status of the TCP/IP network and IMS TOC.

In the following examples the datastore definition is:
`DATASTORE=(ID=DSNAME,MEMBER=ITOCNAME,TMEMBER=IMSNAME,GROUP=GRPNAME...)`

1. You can check the status of the IMS TCP/IP OTMA Connection from IMS using the following IMS commands:
   - `/DIS OTMA`
   - `/DIS TMEMBER IMSNAME TPIPE DSNAME`

   /DIS OTMA

   When the IMS TCP/IP OTMA Connection 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*
-ITOCNAME          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
ITOCNAME
IMSNAME            n         n      0
```

   Note: n = count

2. You can also check status using the following IMS TCP/IP OTMA Connection display commands:
   - `VIEWHWS`
   - `VIEWPORT`
   - `VIEWDS`

   For details of these commands, see “IMS TCP/IP OTMA Connection commands” in the *IMS TCP/IP OTMA Connection Programmer’s Reference*.
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 TOC commands (or whenever you are using the IMS TCP/IP OTMA Connection), 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 IMS TCP/IP OTMA Connection Messages and Codes.
5. IMS TCP/IP OTMA Connection requires that all active clients, whether they are IMS Web or non-IMS Web TCP/IP clients, have unique client names. IMS Web Runtime creates unique RUNames which identify each request that an application makes to execute an IMS transaction. If you are using non-IMS Web TCP/IP clients, you must ensure that your clients each use a unique client name (see “Using generated code in non-Web applications” in the *IMS Web Programmer’s Reference* for more information). This is the name that is displayed for a client in the “CLIENT=” or “ORIGIN=” fields in *IMS TCP/IP OTMA Connection Messages and Codes*.
6. **Restriction**: Those IMS applications that issue a `CHNG` call and modify the alternate PCB with a transaction code are not supported. The reason they are not supported is that the processing of the program-to-program switch within IMS is not known to IMS TOC.

Invoking the IMS TCP/IP OTMA Connection

You invoke the IMS TCP/IP OTMA Connection using either an MVS procedure or an MVS job. If you start multiple IMS TCP/IP OTMA Connections (IMS TOCs) with the same configuration, a connection outage can occur.

Recommendation: To avoid starting the same IMS TCP/IP OTMA Connection system more than once, start the IMS TCP/IP OTMA Connection 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 TCP/IP OTMA CONNECTION USING A JOB                  *
//*********************************************************************
//HWS01    EXEC HWS,SOUT=A
```

The following example shows the JCL statements required to define the MVS environment for the IMS TCP/IP OTMA Connection.

```
//HWS       PROC  RGN=4096K,SOUT=A,
//             BPECFG=BPECFGHT,
//             HWSCFG=HWSCFG00
//*
//*********************************************************************
//*  BRING UP AN IMS TCP/IP OTMA CONNECTION SYSTEM                    *
//*********************************************************************
//STEP1    EXEC PGM=HWSHWS00,REGION=&RGN,TIME=1440,
//              PARM='BPECFG=&BPECFG,HWSCFG=&HWSCFG'
//STEPLIB  DD   DSN=HWS.RESLIB,DISP=SHR
//         DD   DSN=BPE.RESLIB,DISP=SHR
//PROCLIB  DD   DSN=USER.PROCLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
```

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 TOC 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 BPE specifications.

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

Installing the HWSSMPL0 sample user message exit

The purpose of the sample user exit is to provide users with the flexibility to use their own message formats to fit their specific business needs. The sample program, HWSSMPL0, returns the MOD name to the client for message formatting if the name is supplied by the IMS transaction. You can also use your own formats to pass the client's authentication and have this or another user exit verify client authentication. The user exit offers unlimited possibilities for customization. To install the sample user exit, perform the following steps.

1. Assemble HWSSMPL0. HWSSMPL0 and the four macro files, HWSIMSCB, HWSIMSEA, HWSEXPRM and HWSOMPFX, are members of the PDS into which you receive the GENLIBB dataset in step 9 on page 10 of Installing the IMS TCP/IP OTMA Connection (IMSHWS.GENLIBB in the example)*
2. link-edit the output from the assembled job to create a load module named HWSSMPL0.
3. Copy the load module into your RESLIB.
4. Modify the IMS TCP/IP OTMA Connection configuration file so that it includes the HWSSMPL0 user exit in the TCPIP statement, as follows:

   ```
   TCPIP=(...,EXIT=(HWSSMPL0),..)
   ```
5. Restart IMS TCP/IP OTMA Connection.

For a description of the user exit message structures, see “User exit message description and structures” on page 44.

Installing the HWSWEB00 sample user message exit

The purpose of this sample user exit is to provide IMS Web users with the flexibility to do their own security checking.

1. Assemble HWSWEB00. HWSWEB00 and the four macro files, HWSIMSCB, HWSIMSEA, HWSEXPRM and HWSOMPFX, are members of the PDS into which you receive the GENLIBB dataset in step 9 on page 10 of Installing the IMS TCP/IP OTMA Connection (IMSHWS.GENLIBB in the example).
2. link-edit the output from the assembled job to create a load module named HWSWEB00.
3. Install the load module in your RESLIB.

   IMS TCP/IP OTMA loads HWSWEB00 from the RESLIB during initialization.

For a description of the user exit message structures, see “User exit message description and structures” on page 44.

Installing the HWSIMSO0 user message exit

The HWSIMSO0 user message exit replaces the EZAIMSO0 exit that was previously provided by TCP/IP. You must use this exit to replace the TCP/IP EZAIMSO0 exit.

Requirement: The JCL that is provided in HWSJCLIN link-edits HWSIMSO0 with the name EZAIMSO0. However, you can change the name to any name you want to use. But, if you change the name, you must remove *EZAIMSO0* from the configuration file, PARM EXIT=, and replace EZAIMSO0 with the new name.

The HWSIMSO0 user message exit is installed as part of HWSxxxxx module installation, using step 3 of HWSJCLIN. (HWSJCLIN is a sample link-edit job contained in GENLIB.BIN.) During installation, you need to modify the JCL for this exit routine for it to add the TCP/IP library. The following example is the DD statement that you need to change during installation:

```
//AEZAMOD1 DD DSN=W32A.TCPIP.V3R2.AEZAMOD1,DISP=SHR,UNIT=SYSDA,
//     VOL=SER=MVSS06
```

The following lines are the INCLUDE statements for HWSIMSO0 from step 3 of HWSJCLIN:

```
//SYSLIN DD *
INCLUDE AEZAMOD1(EZAAE05F)
INCLUDE LOAD(HWSIMSO0)
ENTRY  HWSIMSO0
MODE RMODE(24),AMODE(31)
NAME EZAIMSO0(R)
```

You can add the following INCLUDE statement to add the TCP/IP security exit, IMSLSECX:

```
  INCLUDE USERLOAD(IMSLSECX)
```

Downloading the IMS client for Java sample program

Attention: IMS TOC and the IMS client for Java sample program are packaged using Info-ZIP's compression utility to create a self-extracting executable archive (zip) file. This program uses UnZip internally to extract the archived files. Info-ZIP's software (Zip, UnZip and related utilities) is free and can be obtained as source code or executables from various anonymous-ftp sites, including ftp.uu.net:/pub/archiving/zip/* or from the Web at http://www.cdrom.com/pub/infozip/.

1. Following the download instructions for IMS TCP/IP OTMA Connection (see "Installing the IMS TCP/IP OTMA Connection" on page 8), download HWSMH*vrm*.exe to a directory on an OS/2 or Windows platform.
2. If you choose to do so, you can enter `HWSMHvrm[.exe] -t` at an OS/2 or Windows command prompt. This will cause an integrity check of the HWSMH*vrm*.exe zip file to execute without actually extracting any of the files from the zip file. To extract files to the current directory, enter `HWSMHvrm[.exe]` at an OS/2 or Windows command prompt.
3. Move the JAVASAMP.exe file from the current directory (the directory into which it was extracted) to a directory on an OS/2 or Windows NT drive that supports long file names. This directory can be the directory where the sample client will be installed or another temporary directory. The remaining files in the directory where HWSMH*vrm*.exe was extracted are used for installing the IMS TCP/IP OTMA Connection.

   Attention: This directory (where you move the JAVASAMP.exe file) **must** be located on an OS/2 or Windows drive that supports the long file names used for the Java files.
4. Optionally, you can enter `JAVASAMP[.exe] -t` at an OS/2 or Windows command prompt. This will cause an integrity check of the JAVASAMP.exe zip file to execute without actually extracting any of the files from the zip file. Run JAVASAMP.exe by entering `JAVASAMP[.exe]` at an OS/2 or Windows command prompt. The expanded files will be placed in the current directory where JAVASAMP.exe is executed.
5. Following the instructions in the next three sections, install the HWSSMPL0 sample user exit and the Java client.

Installing the IMS client for Java sample program

The IMS client for Java sample program can be installed on any platform on which a Sun compatible Java virtual machine has been installed.

1. If the IMS client for Java sample program needs to be installed on a platform other than on the one where it was expanded, copy the files for it to the platform where the sample program will be installed.
2. Modify the source code to match your environment. (See “Modifying the sample Java client”.)
3. Enter `javac *.java` at a command prompt for which the current directory is set to the directory containing all of the Java source files for the sample program. This will create the IMS client for Java class files. Java Development Kit v1.1 users can enter `javac -deprecation *.java` to see deprecated methods.

Modifying the sample Java client

You must modify the FrameInput.java file to construct input data that matches your environment (hostname, port number, transaction, and so forth).

The HWSSMPL0 program does not impose any limitations on the number of input and output message segments. The Java client uses multi-segment input and output text areas.

IMS TOC requires that all active clients have unique client LUNAMEs that, for the sample Java client, are taken from the user ID field defined in FrameInput.java. Therefore, if you intend to allow multiple sample Java clients to run simultaneously, which is usually the case, you must either modify the FrameInput.java file so that the user ID will be unique for each active client at any given time or ensure in some other way that the user ID for each active client is unique. For test purposes, just be sure that you use a unique user ID for each Java client when you click Submit.

Chapter 4. IMS TCP/IP OTMA Connection user exit support

The IMS TCP/IP OTMA Connection communicates with clients using an OTMA message header that is defined in the HWSOMPFX macro, and communicates with IMS via an XCF session. Clients who use TCP/IP Socket calls as their communication vehicle can design a user exit routine that runs with the IMS TCP/IP OTMA Connection 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 via a TCP/IP connection. The IMS TCP/IP OTMA Connection automatically sends and receives messages when they are formatted correctly.

This section describes:

How the IMS TOC communicates with a TCP/IP client

The IMS TCP/IP OTMA Connection expects all client messages that it receives to start with a common 32-byte message prefix. The following table shows the fixed format preceding the input message sent to IMS TOC from IMS Web clients.

Table 1. Fixed format

Field	Length	Meaning
HDR_LLLL	4 bytes	Length of the total message, including this 32-byte message prefix. This field is read as a big-endian binary number. The value must be between 32 and 10,000,000 inclusive.
HDR_LL	2 bytes	Length of the prefix format. For messages from the IMS Web client, the length is X'1C' or binary '00000000 00011100'.
HDR_ZZ	2 bytes	Reserved.
HDR_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 the IMS TOC communicates with user exits" on page 36.
Reserved	4 bytes	Reserved for future use. Initialized to binary zeros.
HDR_FLG5	1 byte	Binary value. Input message type. Binary '10000000' - OTMA headers built by client. Binary '01000000 - translation done by client.

Table 1. Fixed format (continued)

Field	Length	Meaning
HDR_RESV	3 bytes	Reserved for future use.
HDR_CLID	8 bytes	Character string. It specifies the name of the client ID that is used by IMS TOC. If this string is not supplied from the client, then the user exit must generate it. The client ID is returned to IMS TOC from the exit in the EXIT PARMLIST field, EXPREA_CLID.

This message prefix tells the IMS TCP/IP OTMA Connection how long the message is, and to which user exit the message is to be passed. For the complete IMS Web message structure, see the table under “IMS Web message structure - type 1” on page 48.

The IMS TCP/IP OTMA Connection will support prior IMS TOC applications only after you make modifications to the current message structure.

To remove the 32K message restriction and to separate the LTERM override name from the client name, it was necessary in the 2.1 release of IMS TOC to change the format of the message for TCP/IP applications.

Restriction: IMS TOC 2.1 no longer supports the TCP/IP message format that was supported by IMS TOC 1.1 and 1.2.

The base structure for non-IMS Web clients is shown in the following table. It contains the 32-byte message prefix followed by the user-defined structure.

Table 2. Base structure

Field	Length	Meaning
HDR_LLLL	4 bytes	Length of the total message, including this 32-byte message prefix. This field is read as a big-endian binary number. The value must be between 32 and 10,000,000 inclusive.
HDR_LL	2 bytes	Length of the prefix format. For messages from the non-IMS Web client, the length includes the common 32-byte prefix plus the user-defined portion.
HDR_ZZ	2 bytes	Reserved.
HDR_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 the IMS TOC communicates with user exits” on page 36.
Reserved	4 bytes	Reserved for future use.
HDR_FLG5	1 byte	Binary value. Input message type. Binary ’10000000’ - OTMA headers built by client. Binary ’01000000 - translation done by client.
HDR_RESV	3 bytes	Reserved for future use.

Table 2. Base structure (continued)

Field	Length	Meaning
HDR_CLID	8 bytes	Character string. It specifies the name of the client ID that is used by IMS TOC. If this string is not supplied from the client, then the user exit must generate it. The client ID is returned to IMS TOC from the exit in the EXIT PARMLIST field, EXPREA_CLID.

Following the client ID (HDR_CLID) of the common portion of the prefix is the user portion, which must have the following (required by IMS TOC):

Table 3. 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 TOC in the HWSEXPRM structure in field EXPREA_DSID.

The following items should be considered for your installation:

- RACF values
  - User ID
    8 bytes
  - Group Name
    8 bytes
  - Pass ticket
    8 bytes
  - Transaction code
    8 bytes
- LTERM override name
  8 bytes
- MFS MOD name
  8 bytes
- Other required data for user-written exit
- Flag bytes for:
  - MFS MOD name to be returned
  - Commit mode
  - Sync level
  - ACK, NACK, and deallocate
  - Security scope

Recommendation: If your installation is considering Java and the support of UNICODE, you might want to increase the 8-byte field (starting with client ID) to 16-byte fields, and left justify the data with blank pads to the right. By providing 16-byte fields, you will need to do minimal rework if and when UNICODE support is required.

The following table shows an example of one new TCP/IP (non-IMS Web) fixed format preceding the input message. This format is the format used for both the HWSIMSO0 and HWSSMPL0 exits.

Table 4. New TCP/IP fixed format

Field	Length	Meaning
HDR_LLLL	4 bytes	Length of the total message, including this 4-byte field. This field is read as a big-endian binary number. The value must be between 32 and 10,000,000 inclusive. The first 32 bytes are the same for all input messages.
HDR_LL	2 bytes	Length of the TCP/IP IRM header. For messages from non-IMS Web clients, the length is X'50' or binary '00000000 01010000'.
HDR_ZZ	2 bytes	Reserved.
HDR_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 the IMS TOC communicates with user exits" on page 36.
Reserved	4 bytes	Reserved for future use.
HDR_FLG5	1 byte	Binary value. Input message type. Binary '10000000' - OTMA headers built by client. Binary '01000000' - Translation done by client.
HDR_RESV	3 bytes	Reserved for future use.
HDR_CLID	8 bytes	Character string. It specifies the name of the client ID that is used by IMS TOC. If this string is not supplied from the client, then the user exit must generate it. The client ID is returned to IMS TOC from the exit in the EXIT PARMLIST field, EXPREA_CLID.
HDR_FLG1	1 byte	Binary value. This value is used to specify if the MFS mod name is to be returned. Binary '00000000' - user requests no MFS mod name to be returned. Binary '10000000' - user requests MFS mod name to be returned. If this value is not supplied by the client, the user exit must use a default value. The MFS mod name flag is returned to IMS TOC from the exit in the EXIT PARMLIST field, EXPREA_FLAG1.

Table 4. New TCP/IP fixed format (continued)

Field	Length	Meaning
HDR_FLG2	1 byte	Binary value. It specifies the commit mode. Binary '01000000' - commit mode '0'. Binary '00100000' - commit mode '1'. If this value is not supplied from the client, the user exit must use a default value. The commit mode flag is returned to IMS TOC from the exit in the OTMA header field, OMHDRSYN.
HDR_FLG3	1 byte	Binary value. It specifies the sync level. Binary '00000000' - sync level is 'NONE'. Binary '00000001' - sync level is 'CONFIRM'. If this value is not supplied from the client, the user exit must use a default value. The sync level flag is returned to IMS TOC from the exit in the OTMA header field, OMHDRSLV.
HDR_FLG4	1 byte	Character value. It specifies if the client is sending: • A = ACK - Positive acknowledgment • N = NACK - Negative acknowledgment • D = DEALLOCATE - Deallocate connection The value is sent to IMS TOC 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 TOC.
HDR_TRAN	8 bytes	Character string. It specifies the IMS transaction code.
HDR_DSID	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 TOC from the exit in the OTMA header field, OMUSR_DESTID.
HDR_LTERM	8 bytes	Character string. It specifies the IMS LTERM override. This field can be set to a valid name or to blanks. The LTERM override name is returned to IMS TOC from the exit in the OTMA header field, OMHDRLM.

Table 4. New TCP/IP fixed format (continued)

Field	Length	Meaning
HDR_RFUID	8 bytes	Character string. It specifies the RACF user ID. The client must provide it if RACF is to be used. The RACF user ID name is returned to IMS TOC from the exit in the OTMA header field, OMSECUID.
HDR_RFGPN	8 bytes	Character string. It specifies the RACF group name. The client must provide it if RACF is to be used. The RACF group name is returned to IMS TOC from the exit in the OTMA header field, OMSECGRP.
HDR_RFPT	8 bytes	Character string. It specifies the RACF PASSTICKET. The client must provide it if RACF is to be used. The PASSTICKET value is returned to IMS TOC from the exit in the OTMA header field, OMUSR_PASSTICK.

For the complete non-IMS Web message structure used by the HWSIMSO0 and HWSSMPL0 exits, see the table under “Non-IMS Web message structure - type 2” on page 49.

The output from the user exit that is returned to IMS TOC has a new structure for release 2.1. The change was made to eliminate a series of getmains, freemains, and move operations of the message.

The following table shows the structure (one occurrence per message) of the message returned by the non-IMS Web client exit.

Table 5. Structure 1

Field	Length	Meaning
BPE header	64 bytes	Defined in the following section.
OTMA structure	Total length of OTMA header	See the HWSOMPFX macro (full OTMA structure) at the end of this section.
LLZZDATA	n bytes	• LL - length of segment • ZZ - set to binary zeros • DATA - user data
The above LLZZDATA is repeated to a maximum of 32 KB overall length. If there is more data, then the structures continues as shown in Table 6 on page 35.		
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.

The following table shows the structure that is repeated until all data has been mapped to be returned to IMS TOC.

Table 6. Structure 2

Field	Length	Meaning
BPE header	64 bytes	Defined in the following section.
OTMA structure	32 bytes	See the HWSOMPFX macro (control OTMA structure only) at the end of this section.
LLZZDATA	n bytes	• LL - length of segment • ZZ - set to binary zeros • DATA - user data
The above 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.

The following table shows the BPE header layout.

Table 7. 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 TOC 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. These chain pointers are set by the non-IMS Web user exit.
STORAGE TYPE	8 bytes	This field is managed by IMS TOC and should not be modified by the user exit.
TYPE ACCESS	4 bytes	This field is managed by IMS TOC and should not be modified by the user exit.
SUBPOOL	1 byte	This field is managed by IMS TOC and should not be modified by the user exit.
Reserved	43 bytes	This field is managed by IMS TOC and should not be modified by the user exit.

How the IMS TOC communicates with user exits

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

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

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

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

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

In order to provide full user exit support in the IMS TOC environment, every user exit routine must include the subroutines INIT, READ, XMIT, TERM, and EXER. Only assembler language exits are supported by IMS TOC.

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

Register contents on subroutine entry

Register	Contents
1	Pointer to a parmlist that is defined in the HWSEXPRM macro.
14	Return address of the IMS TCP/IP OTMA Connection.
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

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 entry

Field	Length	Meaning
EXPRM_FUNCTION	4 bytes	Character string of value INIT. Specifies that the function to be performed is: `Initialize user exit`.

Field	Length	Meaning
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.

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 TOC. Typically, one of the MSGIDs is used by ASCII clients and the other by EBCDIC clients. IMS TOC 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 two reserved MSGIDs (see the following restriction), provided that the values are both unique among user exits called by a given IMS TOC. Blanks and binary 0 are significant. The IMS TOC 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.

Restriction: In addition to the reserved MSGID, `'*IRMREQ*'`, mentioned above for the support of existing IMS TCP/IP messages, `'*HWSWEB*'` is also a reserved MSGID and is used for IMS Web client support. A user exit that tries to use `'*HWSWEB*'` is rejected. In the case of duplicate MSGID identifiers, one of the user exits that uses the conflicting identifier must be dropped or rewritten with a unique identifier. A system administrator should coordinate the assignment of MSGIDs.

Contents of parmlist pointed to by register 1 at exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPINI_RETCODE	4 bytes	Binary. Specifies the return code. • 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.

Field	Length	Meaning
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 TOC 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 TOC is ready to receive messages from TCP/IP application programs. The IMS TOC 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 TOC. The IMS TOC 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

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. This work area is freemained upon return to IMS TOC.
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.

Field	Length	Meaning
EXPREA_OBUFSIZE	4 bytes	Binary. Specifies the size of the output buffer.
EXPREA_FLAG1	1 byte	Data string flag • X'80' - Input data contains a MSGID maching EXPINI_STRING1. • X'40' - Input data contains a MSGID matching EXPINI_STRING2.
EXPREA_FLAG2	1 byte	Data flag • X'01' - Data moved by exit from INBUF to OUTBUF.
Reserved	2 bytes	Reserved space.
EXPREA_RACFID	8 bytes	Character string. Specifies the default user ID for RACF.
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.

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.

The IMS TCP/IP OTMA Connection 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 exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPREA_RETCODE	4 bytes	Binary. Specifies the return code. • 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.

Field	Length	Meaning
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 the IMS TCP/IP OTMA Connection. This field is only meaningful when EXPREA_RETCODE = 0 or 4.
EXPREA_UFLAG1	1 byte	User flag.
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 Web 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 TOC prepares to present the data to a datastore. EXPREA_UFLAG1 is also saved by the IMS TOC. 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. The IMS TOC 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 the IMS TOC 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. If the IMS TOC does not detect any errors in the output data, the XMIT subroutine is called where the data is passed to IMS OTMA for processing by the datastore. The IMS TOC 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 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.

Field	Length	Meaning
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. This work area is freemained upon return to IMS TOC.
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 exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPXMT_RETCODE	4 bytes	Binary. Specifies the return code. • 0=XMIT function was successful. Process the data. • 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 the IMS TCP/IP OTMA Connection. 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 the IMS TCP/IP OTMA Connection 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 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 TCP/IP OTMA Connection 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. This work area is freemained upon return to IMS TOC.

The user exit finishes all its termination processes here.

Contents of parmlist pointed to by register 1 at exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.
EXPTRM_RETCODE	4 bytes	Binary. Specifies the return code. • 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, HWSSMPL0, and HWSWEB00).

IMS TCP/IP OTMA Connection shutdown proceeds independently of the return code value. The return code merely indicates the completeness of the user exit cleanup.

EXER subroutine

When the IMS TCP/IP OTMA Connection 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 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. This work area is freemained upon return to IMS TOC.
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 • 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.
Reserved	2 bytes	Reserved space.
EXPXER_CODE	4 bytes	Binary. Specifies the failure code. • 4=Error in the output bufer from the previous READ function.
EXPXER_REASON	4 bytes	Binary. Specifies the failure reason. • 20=Segment length error • 24=Missing first in chain flag • 28=Missing last in chain flag • 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 exit

Field	Length	Meaning
Reserved	68 bytes	Reserved space.

Field	Length	Meaning
EXPXER_RETCODE	4 bytes	Binary. Specifies the return code. • 4=Send the data in EXPXER_OUTBUF back to client. • 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, the IMS TCP/IP OTMA Connection 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 TOC allows up to 15 user exits to be defined in the configuration file (see the example on page 14). There are two input message structures supported by IMS TOC and two message structures supported on return from a user exit.

Input messages from client

The following table shows the structure for input messages received from the client by IMS TOC.

Input message structure type	OTMA header present	Exit data translated	Exit type flag (IRMHDR_FLG5)	Supporting msg type
1	Y	Y	11000000	HWSWEB00
1	Y	N	10000000	HWSSMPL0 modified to not build OTMA headers
2	N	Y	01000000	HWSSMPL0 modified to not translate data
2	N	N	00000000	HWSIMSO0 or HWSSMPL0

The following table shows the structure for input messages from the client that are returned by the exit back to IMS TOC.

Input message structure type	Exit output message structure type	Exit type flag (IRMHDR_FLG5)	Supporting msg type
1	1	11000000	HWSWEB00
1	1	10000000	HWSSMPL0 modified to not build OTMA headers

Input message structure type	Exit output message structure type	Exit type flag (IRMHDR_FLG5)	Supporting msg type
2	3	01000000	HWSSMPL0 modified to not translate data
2	3	00000000	HWSIMSO0 or HWSSMPL0

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 TOC

All output messages received by IMS TOC 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 TOC

The message returned to IMS TOC 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 TOC TCP/IP message exit (HWSIMSO0)

This TCP/IP exit is shipped with IMS TOC, and link-edited into the IMS TOC RESLIB. You must use this TCP/IP exit rather than the one shipped by TCP/IP. The installation must place the IMS TOC RESLIB that contains the IMS TOC supplied exit (HWSIMSO0) in front of the TCP/IP RESLIB. The HWSIMSO0 exit is shipped as object code only (OCO). (See the sample user exit, HWSIMSO0. You can install it as described in “Installing the HWSIMSO0 user message exit” on page 25.) 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 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 4 on page 32).

The IMS TOC HWSIMSO0 exit performs the translation from ASCII to EBCDIC and builds 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 exit HWSSMPL0, you can modify it and link-edit it out as HWSIMSO0 to replace the copy supplied by IMS TOC, if you want to change any of the options (for example, translation, OTMA build, commit mode, sync level, etc.) in HWSIMSO0.

The IMS TOC supplied user exit, HWSIMSO0, calls the user-provided security exit, IMSLSECX, and passes a parm list in register 1 to the security exit if one is defined in the exit. For the security parm list structure, see “Security exit” on page 47.

Input message structure passed to HWSIMSO0 exit
The message structure (type 2) is defined in “Non-IMS Web message structure - type 2” on page 49.

Input message structure returned from HWSIMSO0 exit
The message structure (type 3) is defined in “Non-IMS Web message structure - type 3” on page 50.

Output message passed to HWSIMSO0 exit
The message structure is defined in “Non-IMS Web message structure” on page 53.

Output message returned from HWSIMSO0 exit
The message structure is defined in “Non-IMS Web message structure” on page 54 .

Sample user exit message (HWSSMPL0)

The sample exit performs the same functions as the IMS TOC HWSIMSO0 exit, is supplied as source code, and can be modified by the installation.

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 4 on page 32). Or, you can change the exit to set the COMMIT mode and SYNC level to the desired values.

The IMS TOC HWSSMPL0 exit performs the translation from ASCII to EBCDIC and builds 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.

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 47.

Input message structure passed to HWSSMPL0 exit
The message structure is defined in “Non-IMS Web message structure - type 2” on page 49.

Input message structure returned from HWSSMPL0 exit
The message structure is defined in “Non-IMS Web message structure - type 3” on page 50.

Output message passed to HWSSMPL0 exit
The message structure is defined in “Non-IMS Web message structure” on page 53.

Output message returned from HWSSMPL0 exit
The message structure is defined in “Non-IMS Web message structure” on page 54.

IMS Web client user exit message (HWSWEB00)

This IMS Web client exit is shipped with IMS TOC, 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 Web client server. HWSWEB00 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”.

Input message structure passed to HWSWEB00 exit
The message structure is defined in “IMS Web message structure - type 1” on page 48.

Input message structure returned from HWSWEB00 exit
The message structure is defined in “IMS Web message structure - type 1” on page 50.

Output message passed to HWSWEB00 exit
IMS Web output messages are not passed to the exit.

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 TOC if RACF parameters are provided in the OTMA header when the message exit returns the message.

The name of the security exit called by HWSIMSO0 is *IMSLSECX*. The name of the security exit called by HWSSMPL0 is *user supplied* and is defined in the HWSSMPL0 message exit. If you require a security exit, you must provide the exit and modify the HWSSMPL0 to provide the security exit name. The name of the security exit called by HWSWEB00 is *user supplied* and must be defined in the HWSWEB00 message exit. The name of the security exit called by HWSJAVA0 is *user supplied* and is defined in the HWSJAVA0 message exit.

Parameter list for user security exit:

The following is the list and order of parameters being passed to the security exit, IMSLSECX. The order of the parameters is *fixed* for the IMS TOC supplied exit, HWSIMSO0.

- 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.
- Addr of RACF group ID

  The address points to a field containing blanks.

Message structures

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

Input message from client and passed to exit

Input messages from the client consist of IMS Web and non-IMS Web message structures.

IMS Web message structure - type 1: The following table shows the input message format supported by IMS TOC from an IMS Web client.

Field	Length	Meaning
LLLL	4 bytes	Length of entire message, including LLLL field
LL	2 bytes	Length of IMS Web interface header, including LLZZ field
ZZ	2 bytes	Reserved (set to binary zeros)
ID	8 bytes	Char value of *HWSWEB*
Reserved	4 bytes	Reserved (set to binary zeros)
FLG5	1 byte	Binary value for input message type
Reserved	3 bytes	Reserved (set to binary zeros)
Client ID	8 bytes	Char value of a unique client ID
OTMA HDR	466 bytes	See OTMA DSECT 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

Field	Length	Meaning
OTMA HDR	20 bytes	See OTMA DSECT 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 HDR	20 bytes	See OTMA DSECT 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 Web message structure - type 2: The following table shows the input message format supported by IMS TOC from a non-IMS Web client.

Field	Length	Meaning
LLLL	4 bytes	Length of entire message, including LLLL field
LL	2 bytes	Length of TCP/IP interface header
ZZ	2 bytes	Reserved (set to binary zeros)
ID	8 bytes	Char value of *IRMREQ*
Reserved	4 bytes	Reserved for future use.
FLG5	1 byte	Binary value for input message type
Reserved	3 bytes	Reserved (set to binary zeros)
Client ID	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.		
FLG1	1 byte	Binary MFS flag
FLG2	1 byte	Binary COMMENT MODE flag
FLG3	1 byte	Binary SYNC LEVEL flag
FLG4	1 byte	Char value conversation byte
TRANCODE	8 bytes	Char value for user transaction code
DATASTORE	8 bytes	Char value for Datastore ID
LTERM NAME	8 bytes	Char value for LTERM override name

Field	Length	Meaning
RACF UID	8 bytes	Char value for RACF user ID
RACF GNM	8 bytes	Char value for RACF group name
PASSTICKET	8 bytes	RACF passticket/password
The following is the data structure for all non-IMS Web clients.		
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 and last data segment
ZZ	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data with this data segment (no tran code)
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 Web and non-IMS Web message structures.

IMS Web message structure - type 1: The IMS Web exit output message format that is supported by IMS TOC is the same message format of the input message. See “IMS Web message structure - type 1” on page 48 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 Web message structure - type 3: The following table shows the output message format supported by IMS TOC from the supplied HWSIMSO0 and HWSSMPL0 exits (non-IMS Web client exits). 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.

Field	Length	Meaning
BPE HEADER	64 bytes	See BPE header definition that follows this table

Field	Length	Meaning
OTMA HDR	466 bytes	See OTMA DSECT 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 and last data segment
ZZ	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data with this data segment (no tran code)
YY	2 bytes	Binary value of zero
BPE HEADER	64 bytes	See BPE header definition that follows this table
OTMA HDR	32 bytes	See OTMA DSECT 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)
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 with this data segment (no tran code)
...		
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)
YY	2 bytes	Binary value of zero
BPE HEADER	64 bytes	See BPE header definition that follows this table
OTMA HDR	32 bytes	See OTMA DSECT in GENLIB for description
LL	2 bytes	Length of this data segment
ZZ	2 bytes	Reserved (set to binary zeros)

Field	Length	Meaning
DATA	n bytes	User data with the tran code first
...		
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)
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:

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

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***.

Output message from IMS to IMS TOC

Output messages from IMS to IMS TOC consist of the IMS Web and non-IMS Web message structures.

IMS Web message structure: The following table shows the message format from IMS TOC to the client. IMS Web output is not passed to the IMS Web exit.

Field	Length	Meaning
LLLL	4 bytes	Total message length
Id	8 bytes	*HWSWEB*
OTMA HDR	466 bytes	See OTMA DSECT 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 HDR	20 bytes	See OTMA DSECT in GENLIB for description

Field	Length	Meaning
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 HDR	20 bytes	See OTMA DSECT 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)
OTMA HDR	32 bytes	See OTMA DSECT 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)
...		
LL	2 bytes	Length of this data segment
ZZ	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data nth data segment (no tran code)
...		
LL	2 bytes	Length of this and last data segment
ZZ	2 bytes	Reserved (set to binary zeros)

Non-IMS Web message structure: The following table shows the message format from IMS TOC to the exit.

Field	Length	Meaning
OTMA HDR	Length of total OTMA headers.	See OTMA DSECT 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 HDR	20 bytes	See OTMA DSECT in GENLIB for description
LL	2 bytes	Length of 2nd data segment

Field	Length	Meaning
ZZ	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data 2nd data segment (no tran code)
...		
OTMA HDR	20 bytes	See OTMA DSECT 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)

Output message from message exit

Output messages from the message exit consist of the non-IMS Web message structures.

Non-IMS Web message structure: The non-IMS Web 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

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

Field	Length	Meaning
LL	2 bytes	Length of CSM message
ZZ	2 bytes	Reserved (set to binary zeros)
ID	8 bytes	Char value of *CSMOKY*

RSM - Request Status Message

Returned as the only structure of an output message if IMS TOC or the message exit determined an error occurred

Field	Length	Meaning
LL	2 bytes	Length of RSM message

Field	Length	Meaning
ZZ	2 bytes	Reserved (set to binary zeros)
ID	8 bytes	Char value of *REQSTS*
RETC	4 bytes	Return code
RESC	4 bytes	Reason code

The output message is in one of the following formats:

- MFS MOD name requested and data is being sent

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 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 nth and last data segment
ZZ	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data nth data segment (no tran code)
CSM	12 bytes	Complete status message

- MFS MOD name not requested and only data is being sent

Field	Length	Meaning
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 nth and last data segment

Field	Length	Meaning
ZZ	2 bytes	Reserved (set to binary zeros)
DATA	n bytes	User data nth data segment (no tran code)
CSM	12 bytes	Complete status message

- Error detected by IMS TOC or the message exit

Field	Length	Meaning
RSM	20 bytes	Return/Reason codes

Macros

Four macros are supported by IMS TOC: HWSEXPRM, HWSOMPFX, HWSIMSCB, and HWSIMSEB.

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 GENLIB. To see the structure, print the source.

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 GENLIB. To see the structure, print the source.

HWSIMSCB

This macro maps the IMS request messages and BPE header formats used by HWSSMPL0. A copy of this macro is in GENLIB. To see the structure, print the source.

HWSIMSEB

This macro maps the storage area used by HWSSMPL0. A copy of this macro is in GENLIB. To see the structure, print the source.

Glossary

base primitive environment (BPE). A system service component that underlies the IMS TOC address space.

datastore. An IMS TM system that provides transaction and database processing.

IMS TOC. An MVS application program that uses TCP/IP communications to TCP/IP clients or Web browsers and IMS OTMA communication to IMS.

IMS DB. An IMS Database Manager database, which provides host data for TCP/IP clients.

IMS Open Transaction Manager Access (OTMA). A transaction-based connectionless client/server protocol.

port. The interface between TCP and a local process. This interface enables the process to call TCP, and in turn enables TCP to deliver data streams to the appropriate process.

socket. The host IP address appended to the port number. This address is unique on the internet. The connection between two sockets provides a full duplex communication path between the end processes.

TCP/IP. Transmission Control Protocol/Internet Protocol.

XCF. MVS Extended Coupling Facility.

XCF group. A logical collection of XCF members. The datastore and the IMS TOC must join to the same group.

XCF member. An MVS application that joins an XCF group.

Readers' Comments — We'd Like to Hear from You

IMS TCP/IP OTMA Connection
User's Guide and Reference

Publication No. itocug-0002-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
itocug-0002-02

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department BWE/H3
P.O. Box 49023
San Jose, CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line