IMS TCP/IP OTMA Connection User's Guide

itocug-0002-01

March 16, 1998

Candace Garcia
IBM Corporation

IMS TCP/IP OTMA Connection User's Guide

itocug-0002-01

Contents

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 is a trademark of Sun Microsystems, Inc.

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 is a TCP/IP server that enables remote workstations to exchange messages with IMS OTMA. As shown in Figure 1, this server provides communication linkages between remote workstations 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 ITOC” on page 11.

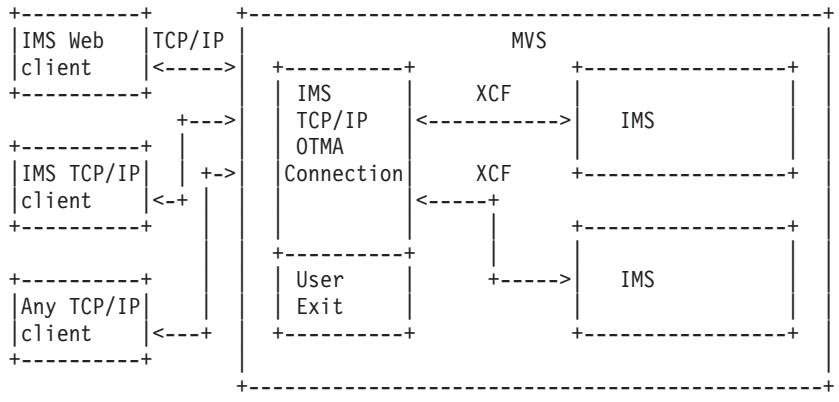


Figure 1. System overview

The IMS TCP/IP OTMA Connection performs router functions between remote workstations and datastores. Request messages received from remote workstations, 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 remote workstations. 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 also supports TCP/IP clients communicating with socket calls; for example, existing IMS TCP/IP clients. 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 EZAIMSO0. For details, see “Chapter 4. IMS TCP/IP OTMA Connection user exit support” on page 25 .

Components

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

- Workstation communication component (WCC)
- Datastore communication component (DCC)
- Command component (CMD)

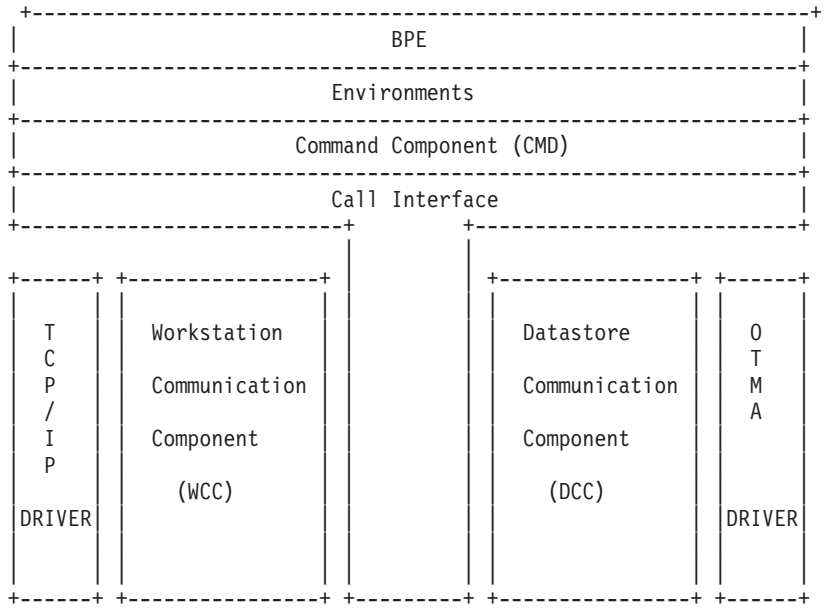


Figure 2. IMS TCP/IP OTMA Connection component layout

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, while 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

Workstation communication component

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

Datastore communication component

The datastore communication component handles communications between the IMS TCP/IP OTMA Connection and the IMS datastores.

Command component

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.”

Driver functions

The following sections list the functions for the TCP/IP communication driver and the OTMA communication driver.

TCP/IP communication driver functions

Open Allocates an open structure, sets up TCP/IP structures, and loads and initializes user exits.

Close Deallocates the open structure and terminates user exits.

Open thread
Allocates a communication structure and intializes a port and listen socket.

Term thread
Deallocates the communication structure and closes the listen socket.

Connect
Creates an accept socket for each request message.

Disconnect
Closes the accept socket.

Transmit
Returns a message to the originating workstation. Invokes a user exit if necessary.

Receive
Receives a message from the workstation. Invokes a user exit if necessary.

Post exit
Driven by MVS TCP/IP, posts a waiting thread when socket calls are completed.

OTMA communication driver functions

Open Allocates an open structure and queries XCF join status.

Close Deallocates the open structure.

Open thread
Allocates a communication structure. Queries an XCF group for the active target member and joins the XCF group.

Term thread
Leaves the XCF group and deallocates the communication structure.

Connect
Issues the client bid to the target server.

Disconnect
Terminates operation.

Transmit
Issues the IXCMSGO send message to MVS/XCF.

Receive
Receives the messages from MVS/XCF.

XCF group exit
Determines if the target member has terminated and alters the ctoken field CXTOKEN_STATE with `TERM`.

XCF message exit
Allows message exit processing by the HWS when driven by XCF.

Driver function flows

The IMS TCP/IP OTMA Connection is a concurrent server that supports multi-threading for transactional requests. The following sections describe the function flows for the TCP/IP driver and the OTMA driver.

TCP/IP driver flows

Figure 3 shows the TCP/IP driver multi-threading flow.

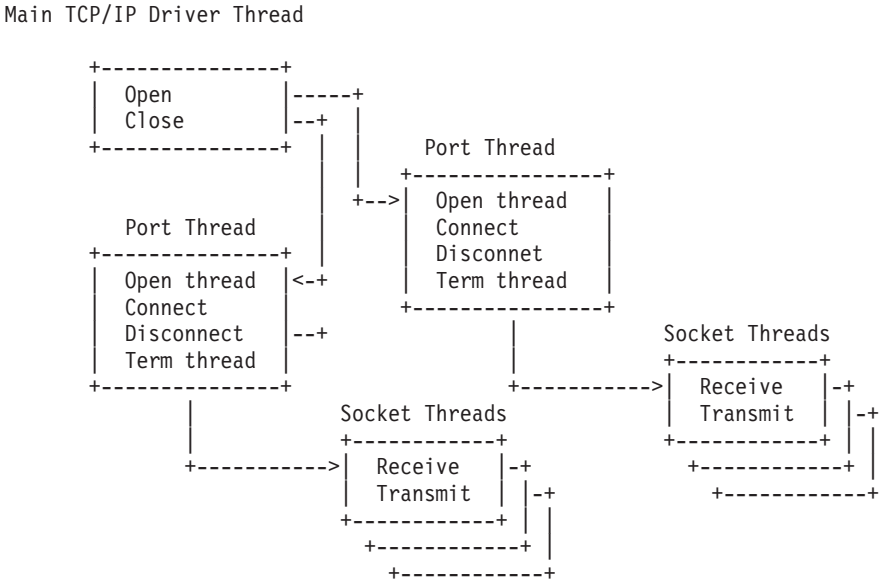


Figure 3. TCP/IP driver multi-threading

For each *portid* defined in the HWSCFG configuration member, the IMS TCP/IP OTMA Connection creates a port thread. This thread does all the initial setup and binds the port with a listen socket. When a request message arrives, the following sequence occurs:

1. The port thread accepts the request and returns to the WCC.
2. The WCC generates a socket thread to process the request.
3. The socket thread receives the entire message and returns to the WCC.
4. The IMS TCP/IP OTMA Connection (HWS) passes the message to the DCC.
5. The DCC invokes the OTMA driver to retrieve data.
6. When the IMS TCP/IP OTMA Connection (HWS) receives a response message from the datastore, the socket thread takes control and transmits the message back.
7. The socket thread terminates.

A socket thread exists only for the duration of one data retrieval cycle. It uses the BPE wait and post mechanism to optimize system usage.

OTMA driver flows

Figure 4 shows the OTMA driver multi-threading flow.

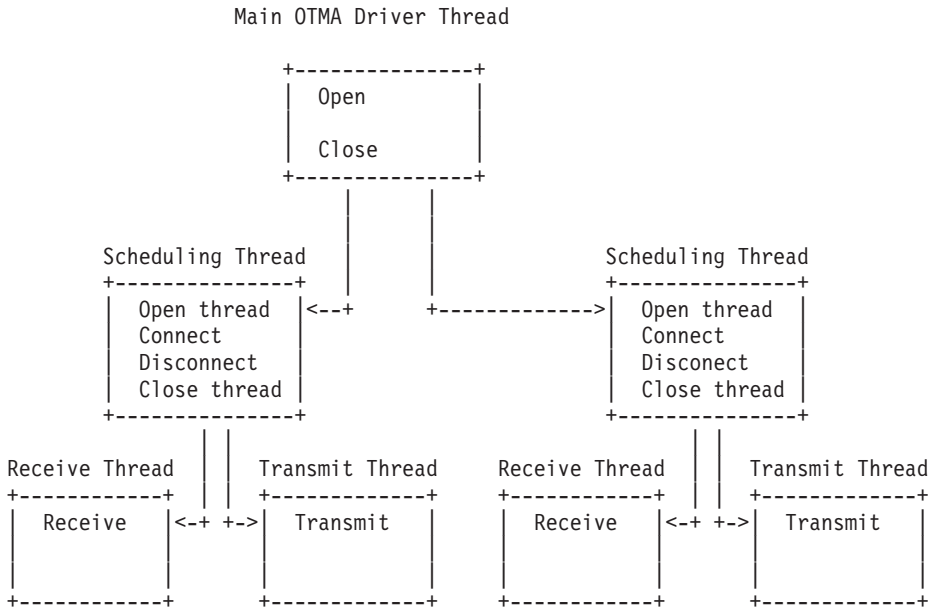


Figure 4. OTMA driver multi-threading

For each datastore that is defined in the HWSCFG configuration member, the IMS TCP/IP OTMA Connection creates a scheduling thread. It then creates a receive thread for receiving messages and a transmit thread for sending them.

Tips

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

1. You can check the status of the IMS TCP/IP OTMA Connection from IMS using the following 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*
-HWSNAME           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
HWSNAME
IMSNAME            1         1      0
```

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 IMS Web client Web browser (or to check the readiness of the host datastore), you can enter the following 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 these 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 IMS Web client browser, 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 LTERM names. IMS Web Runtime creates unique RUNames which identify each request that an application makes to execute an IMS transaction. IMS, in turn, uses the RUName as the name of the LTERM for the request message that uses that RUName. If you are using non-IMS Web TCP/IP clients, you must ensure that your clients each use a unique RUName (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.”

What's new in release 210

The following enhancements have been made for release 210:

- Added conversational support
  - Added ability for user to retain message continuity from a given terminal
  - Added new message HWSP149E
- Enhanced security
  - Added new command, SETRACF
  - Added new error message, HWSP1500E
- Added a new exit routine, HWSWEB00
- Updated installation and configuration
- Added a trace dump example in the messages and codes section

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 later
- IBM DATABASE 2 Version 2.3 (optional)
- Virtual Telecommunications Access Method (VTAM) Version 3.4
- One or more of the following languages, as supported by IMS Transaction Manager:
  - IBM COBOL for MVS & VM and IBM Language Environment for MVS & VM
  - IBM PL/I for MVS & VM and IBM Language Environment for MVS & VM
  - IBM Assembler H or IBM High Level Assembler/MVS
- Resource Access Control Facility (RACF) Version 1.9.2 or equivalent product

Chapter 3. How to install and configure the ITOC

The IMS TCP/IP OTMA Connection provides the following components that enable remote workstations to exchange messages with IMS:

Host Web service (HWS)
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 HWS

This section describes the following:

Installing the IMS TCP/IP OTMA Connection

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

1. Go to the IMS TOC download page.
2. Select the Download link for IMS WebHost - 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 Web). Using the Save dialog box, save the downloaded file to a temporary directory.
3. Run the downloaded .exe file, which is a self-extracting, compressed file, by entering `HWSMHvrm install_directory` at a DOS command prompt (where again *vrm* represents the version, release, and modification of IMS Web and *install_directory* represents a valid DOS path that specifies the directory into which the uncompressed files will be extracted. Use '.' (a single period) to represent the current directory, `HWSMHvrm imsweb` to extract to the imsweb directory on the current drive, or `HWSMHvrm d:\imsweb` to extract to the \imsweb directory on the d: drive).

   Execution of this file generates the sequential files GENLIBB.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 two sample link edit jobs, HWSJCLIN and HWSBPEIN, that you use to link edit the IMS TCP/IP OTMA Connection and BPE load modules into a RESLIB (see "IMS 5.1 considerations" on page 13 and "IMS 6.1 considerations" on page 13). It also contains two sample exit routines, IMS TCP/IP OTMA Connection user exit (HWSSMPL0), for use with the IMS TCP/IP OTMA Connection clients and two macros (HWSEXPRM and HWSOMPFX) that

can be used by that user exit or any IMS TCP/IP OTMA Connection user exits that you write, and the IMS Web user exit (HWSWEB00), where you can issue a RACF function.

LOAD.BIN contains the IMS TCP/IP OTMA Connection load modules, which must be link edited and copied out into the RESLIB that you plan to use for IMS TCP/IP OTMA Connection. The sample user exit contains code to translate messages into the format required by IMS Version 5 Open Transaction Manager Access (OTMA) and is used in conjuction with two macro files, HWSEXPRM and HWSOMPFX.

JAVASAMP.exe contains a sample Java client for IMS TCP/IP OTMA Connection, and HTML and text versions of both the installation instructions and a list of changes to the contents of JAVASAMP.exe. Run JAVASAMP.exe by entering `JAVASAMP install_directory` at a Windows NT command line prompt where *install_directory* represents a valid DOS path that specifies the output directory into which the expanded files will be placed. Use '.' (a single period) to represent the current directory.

CAUTION:
This output directory must be located on a Windows NT NTFS drive in order to support the long file names used for the Java files.

4. If your file transfer tool will not do so automatically, allocate two fixed block record format, 80-byte record length sequential data sets, SEQ.GENLIBB and SEQ.LOAD, into which the sequential data sets will be transferred (see the next step in this procedure). Note that you may use names other than SEQ.GENLIBB and SEQ.LOAD 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 pushbutton and then press 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.

5. 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.

6. 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(SEQ.GENLIBB)
   DSNAME(PDS.GENLIBB)
   RECEIVE INDSN(SEQ.LOAD)
   DSNAME(PDS.LOAD)
   ```

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. The input data set names (SEQ.GENLIBB and SEQ.LOAD in the previous example) are the names of the data sets referred to in steps 4 on page 12 through 6 on page 12 of this procedure.

c. MVS prompts you for the restore data set names (PDS.GENLIBB and PDS.LOAD in the previous example, or names of your choice) after you have entered the `RECEIVE INDSN(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, any previous customization of HWSJCLIN is lost when the previous version is overwritten during the receive of the new GENLIBB data set.

d. If PDS.GENLIBB is a temporary data set, copy the members of PDS.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 PDS.GENLIBB.)

7. 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, and then change the //SYSLMOD DD card to point to your RESLIB. Ensure that your RESLIB has enough directory block space. Submit the modified HWSJCLIN, which then builds the RESLIB for you.
8. Define the IMS TCP/IP OTMA Connection environment for IMS Web, as described in “Defining the IMS TCP/IP OTMA Connection environment” on page 14 .

IMS 5.1 considerations

For IMS 5.1 environments, HWSJCLIN must be executed to link-edit the IMS TCP/IP OTMA Connection (ITOC) load modules (HWSxxxxx) into a HWS RESLIB. We recommend that you place the ITOC modules in a separate HWS RESLIB. The HWSBPEIN must be executed to link-edit the base primitive environment (BPE) load modules (BPExxxxx) into a separate BPE RESLIB. By placing the HWS and BPE modules in two separate RESLIBs in an IMS 5.1 environment, it 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. The BPE maintenance will be part of the IMS Web and ITOC maintenance.

IMS 6.1 considerations

For IMS 6.1 environments, HWSJCLIN must be executed to link-edit the IMS TCP/IP OTMA Connection (ITOC) load modules (HWSxxxxx) into a HWS RESLIB. We recommend that you place the ITOC modules in a separate HWS RESLIB. The HWSBPEIN must NOT be executed. The base primitive environment (BPE) load modules will exist in the IMS 6.1 RESLIB. You must concatenate the HWS RESLIB

and the IMS 6.1 RESLIB, which contains the BPE modules for your HWS region JCL. The BPE maintenance will be part of the normal IMS 6.1 maintenance.

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 HWS startup JCL statements show, both HWS 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 host Web service (HWS)

HWS supports communication between one or more TCP/IP clients and IMS systems. HWS 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 HWS to trigger access to multiple IMS TM systems to balance the workload of TCP/IP client requests. If a single IMS TM system cannot handle the workload of TCP/IP client requests, you can use HWS to balance that workload across multiple IMS TM systems.

To configure HWS, 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 HWS to run in authorized supervisor state and in key 7.
3. Create an HWS configuration member to hold the configuration statements that HWS uses during initialization.

The following sections describe these actions in more detail.

Authorizing HWS to the APF

The resident library in which the HWS 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 HWS is executed in supervisor state and key 7, you add an entry for it in the MVS Program Properties Table (PPT) as follows:

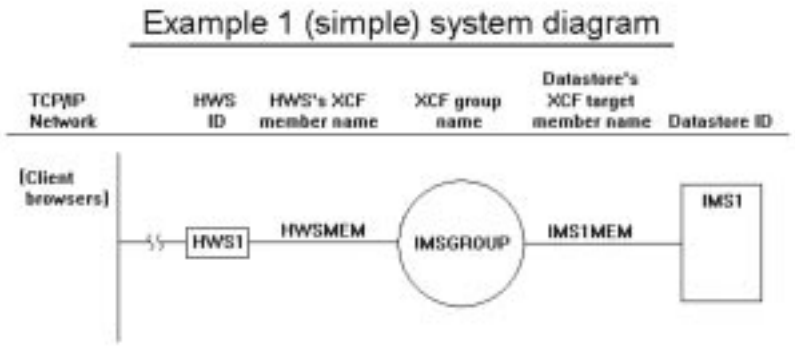
1. Edit the SCHEDxx member of the SYS1.PARMLIB data set.
2. Add the following entry in the MVS PPT:

```
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 HWS configuration member

Specify the environment for HWS as a member in your PROCLIB data set. HWS 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 HWS startup. Specify the member name to use in the HWSCFG= parameter of the HWS startup JCL (see the previous HWS startup JCL example on page 14).



- In the following HWS configuration member, the HWS ID is defined as `HWS01`. This HWS 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`.

```
************************************************
* HWS EXAMPLE 1 CONFIGURATION FILE
************************************************
HWS (ID=HWS01,RACF=N)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),EXIT=(EZAEXIT))
DATASTORE  (ID=IMS1,GROUP=IMSGROUP,MEMBER=HWSMEM,TMEMBER=IMS1MEM)
```

HWS configuration statement parameters

As shown in the previous HWS configuration member example, you define the TCP/IP and IMS OTMA communication to HWS (HWS, TCP/IP, and the datastores with which HWS is to communicate). Each configuration statement begins with `HWS`, `TCPIP`, or `DATASTORE`.

HWS Specify only one HWS.

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

id The HWS 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 TCPIP.

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.

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 HWS to IMS.

DATASTORE
To access IMS OTMA, specify each datastore with which the HWS 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

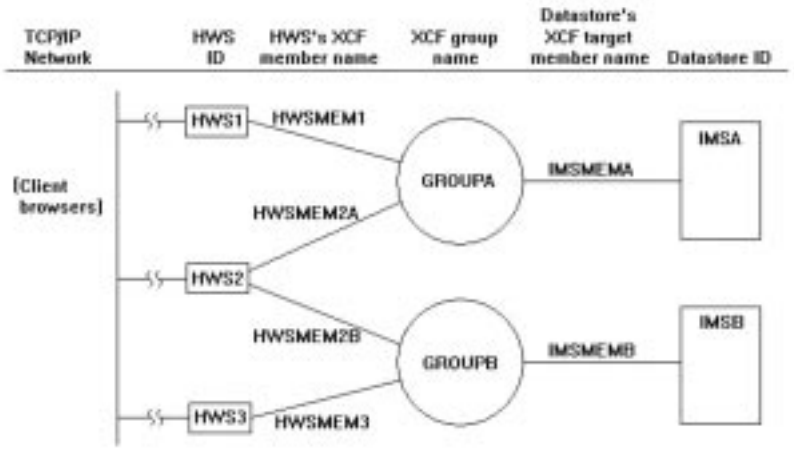
This ID must match the datastore ID that is used in the IMS Web generated (or user generated/modified) CGI source file.

group The XCF group name for the IMS OTMA. HWS uses this value to join the appropriate XCF group(s). Because HWS 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 HWS can join any number of groups

member
The XCF member name for IMS that identifies HWS in the XCF group specified by the &GROUP parameter. This name is the XCF name that IMS uses to communicate with HWS in that XCF group. This XCF member name for HWS 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 HWS 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. In IMS 5.1, the IMS startup parameters must include "...,OTMA=Y,GRNAME=&GROUP,***USERVAR***=&***TMEMBER***,...". In IMS 6.1, USERVAR is replaced by APPLID1 "...,OTMA=Y,GRNAME=&GROUP,***APPLID1***=&***TMEMBER***,..." (Each datastore definition within an HWS configuration member must contain a unique tmember name.

Example 2 (complex) system diagram



- In this example, three HWS's are configured. Each HWS has its own configuration member.
- Each HWS uses a different port number for TCP/IP communications and can belong to multiple XCF groups.
- One or more IMS's can belong to each XCF group.

- When defining multiple datastores which belong to the same XCF group in a single HWS configuration member, the XCF member name for that HWS 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 HWS configuration member. For example, observe that the XCF member name for HWS in the IMSA and IMSB DATASTORE statements in the HWS2 configuration member in this example, 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 HWS 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 HWS security

You can start HWS as a job or as a procedure.

If the datastore (which is IMS) is RACF protected, you have to start HWS as a job with the JOB card specifying a valid *USERID* in order to make the connection from HWS to IMS. The *USERID=&userid* parameter specified in the JOB card of the HWS job JCL is used as the security vehicle to ensure HWS 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 HWS 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 Web 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 HWS 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 HWS trace tables. BPE provides several internal trace tables to capture diagnostic information for the services that it provides. In addition, HWS has several trace tables for tracing its functions.

For each trace table type that BPE and HWS 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 HWS 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 HWS 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 HWS 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*.

HWS trace tables

CMDT HWS command trace table, which traces command activity. When you specify CMDT as the *type*, specify HWS as the *product*.

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

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

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

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

HWSW
HWS to TCP/IP driver trace table, which traces

communication activity and events between TCP/IP drivers and the HWS. When you specify HWSW as the *type*, specify HWS 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.

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 HWS operation. If you have operations problems, increase the level of tracing.

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.

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 HWS:

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

pages Specifies how many 4K 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 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 (48K bytes) 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 HWS to OTMA driver trace table:

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

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

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

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

In this example, the HWS toTCP/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.

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 (HWSs) 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. Using a job to start HWS has the added advantage of allowing you to specify the RACF user ID on the JOB card (in fact, your installation's security procedures might require you to specify this userid). 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 HWS 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 HWS configuration information.

Installing the HWSSMPL0 sample user exit

The purpose of the sample user exit is to provide Internet users with the flexibility to use their own message formats to fit their specific business needs. As the sample program shows, HWSSMPL0 passes back the MOD name that a Java client can use to format its own output messages. The users can also use their 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. Compile HWSSMPL0. HWSSMPL0 and the two macro files, HWSEXPRM and HWSOMPFX, are members of the PDS into which you receive the GENLIBB dataset in step 6 on page 12 of "Installing the IMS TCP/IP OTMA Connection" on page 11 (PDS.GENLIBB in the example).
2. Link edit the output from the compile job to create a loadable module named HWSSMPL0. Ensure that you include the statement **INCLUDE AEZAMOD1(EZAAE05F)** in the linkage file, where *AEZAMOD1* refers to the MVS TCP/IP AEZAMOD1 data set that includes the TCP/IP translation table in the load module. AEZAMOD1 can be found in the MVS TCP/IP libraries installed on your system.
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.

Installing the HWSWEB00 sample user exit

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

1. Compile HWSWEB00. HWSWEB00 and the two macro files, HWSEXPRM and HWSOMPFX, are members of the PDS into which you receive the GENLIBB dataset in step 6 on page 12 of “Installing the IMS TCP/IP OTMA Connection” on page 11 (PDS.GENLIBB in the example).
2. Link edit the output from the compile job to create a loadable module named HWSWEB00.
3. Replace the load module in your reslib.

   IMS TCP/IP OTMA loads the reslib module.

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

The IMS TCP/IP OTMA Connection communicates with IMS Web 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 ITOC communicates with a TCP/IP client"
- "How the ITOC communicates with user exits"
- "Macros" on page 33

How the ITOC communicates with a TCP/IP client

The IMS TCP/IP OTMA Connection expects all client messages that it receives to start with a 12-byte message prefix. The following table describes the prefix format.

Field	Length	Meaning
MSGLength	4 bytes	Length of the total message, including this 12-byte message prefix. This field is read as a big-endian binary number. The value must be between 12 and 10,000,000 inclusive.
MSGID	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 ITOC communicates with user exits".

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, without knowing the actual message format.

The IMS TCP/IP OTMA Connection also supports existing IMS TCP/IP applications. In order to support these applications, the IMS TCP/IP OTMA Connection accepts MSGLength of LLZZ when `MSGID = '*IRMREQ*'`, in either EBCDIC or ASCII format.

How the ITOC communicates with user exits

When the IMS TCP/IP OTMA Connection 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 HWS startup JCL as follows:

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

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

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

In order to provide full user exit support in the IMS TCP/IP OTMA Connection environment, every user exit routine must include the subroutines INIT, READ, XMIT, TERM, and EXER. Only assembler language is supported in release 1 of the IMS TCP/IP OTMA Connection.

When a user exit takes control, it saves the contents of the registers and restores them when returning to the caller. The IMS TCP/IP OTMA Connection provides a 1K buffer in the parmlist to be used for this purpose. The register contents are listed in the following two tables.

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 1K 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 maximum output buffer size for its READ, XMIT, and EXER subroutines. Typically, one of the MSGIDs would be used by ASCII clients and the other by EBCDIC clients. Because the IMS TCP/IP OTMA Connection does not know the actual size of the output data, it allocates the maximum output 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 HWS. Blanks and binary 0 are significant. The IMS TCP/IP OTMA Connection 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 applications, `'*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 which 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.
EXPINI_MAXBUF	4 bytes	Binary. Specifies the maximum size of the output buffer needed to perform later conversion functions.

If the INIT subroutine fails to complete the initialization function successfully, the IMS TCP/IP OTMA Connection 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 TCP/IP OTMA Connection is ready to receive messages from TCP/IP application programs. The IMS TCP/IP OTMA Connection uses the TCP/IP Socket API to receive stream data across the net. The completion of a message is determined by its `MSGLength` value. The IMS TCP/IP OTMA Connection 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 an IMS Web 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 1K buffer for user exit use. The user exit can use this storage for a save area and local variables.
EXPREA_INBUF	4 bytes	Address of the input buffer.
EXPREA_IBUFSIZE	4 bytes	Binary. Specifies the size of the input buffer.
EXPREA_OUTBUF	4 bytes	Address of the output buffer.
EXPREA_OBUFSIZE	4 bytes	Binary. Specifies the size of the output buffer.
EXPREA_FLAGI	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.`
Reserved	3 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.

Field	Length	Meaning
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 may 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 an ASCII-to-EBCDIC conversion is required.

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 RXPREA_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.
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.

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 TCP/IP OTMA Connection prepares to present the data to a datastore. EXPREA_UFLAG1 is also saved by the IMS TCP/IP OTMA Connection. 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 TCP/IP OTMA Connection 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 TCP/IP OTMA Connection 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 TCP/IP OTMA Connection 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 TCP/IP OTMA Connection 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.`
EXPRM_TOKEN	4 bytes	Address of a 1K buffer for user exit use. The user exit can use this storage for a save area and local variables.
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_FLAGI	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 can use `EXPXMT_FLAG1` to determine where the request message from the client originated and whether an EBCDIC-to-ASCII conversion is required. 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 `RXPXMT_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 `EXPXMP_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 1K buffer for user exit use. The user exit can use this storage for a save area and local variables.

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.

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 1K buffer for user exit use. The user exit can use this storage for a save area and local variables.
EXPXER_OUTBUF	4 bytes	Address of the output buffer.
EXPXER_OBUFSIZE	4 bytes	Binary. Specifies the size of the output buffer.
EXPXER_FLAGI	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.

Field	Length	Meaning
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.
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.

Macros

Following are the macros that the user exits use for mapping.

HWSEXPRM

This macro provides the mapping for the parmlist that is passed to the user exit on each subroutine call.

```
MACRO                                                          00010000
HWSEXPRM                                                       00050000
GBLB  &HWSEXPRM 1ST TIME SW FOR MAPPING DSECT                  00090000
AIF   (NOT &HWSEXPRM).AREAGO                                   00130000
MEXIT                                                          00170000
```

```
.AREAGO  ANOP  ,                                                       00210000
&HWSEXPRM SETB  1                                                      00250000
.*                                                                     00290000
********************************************************************** 00330000
***                                                                  *** 00370000
***      HWSEXPRM     - EXIT PARAMETER LIST MAPPING                  *** 00410000
***                     MAPS THE FIELDS POINTED TO BY REGISTER 1     *** 00450000
***                     ON ENTRY TO EXTERNAL EXITS.                  *** 00490000
***                                                                  *** 00530000
***            RESIDENCY: HWS REGION - PRIVATE, KEY 7                *** 00570000
***                                                                  *** 00610000
********************************************************************** 00650000
               SPACE 1                                                 00690000
HWSEXPRM      DSECT              PARAMETER LIST DSECT                  00730000
EXPRMBGN      EQU   *                                                  00770000
********************************************************************** 00810000
*     MAPPING FOR ALL FUNCTIONS                                        00850000
********************************************************************** 00890000
EXPRM_FUNCTION DS    CL4          'INIT' INITIALIZE                    00930000
*                                 'READ' PROCESS INPUT DATA FROM CLIENT 00970000
*                                 'XMIT' PROCESS OUTPUT DATA FOR CLIENT 01010000
*                                 'TERM' PERFORM TERMINATION           01050000
*                                 'EXER' ERROR FOUND IN EXIT DATA      01090000
EXPRM_TOKEN    DS    A            ADDRESS OF 1K BUFFER FOR EXIT'S USE  01130000
EXPRM          EQU   *                                                  01170000
               DS    CL60         INPUT AREA                           01210000
EXRET          EQU   *                                                  01250000
               DS    CL40         RETURN DATA AREA                     01290000
               SPACE 2                                                 01330000
********************************************************************** 01370000
*  INIT FUNCTION SECTION                                               01410000
********************************************************************** 01450000
               ORG   EXRET                                             01490000
*   RETURNED DATA                                                      01530000
EXPINI_RETCODE DS    F            RETURN CODE RETURNED BY EXIT         01570000
*                                 0 = INIT CALL WAS SUCCESSFUL         01610000
*                                 4 = INIT CALL FAILED                 01650000
EXPINI_RSNCODE DS    F            REASON CODE RETURNED BY EXIT         01690000
EXPINI_STRING1 DS    CL8          CLIENT IDENTIFYING STRING 1          01730000
EXPINI_STRING2 DS    CL8          CLIENT IDENTIFYING STRING 2          01770000
EXPINI_MAXBUF  DS    F            MAX OUTPUT BUFFER SIZE               01810000
********************************************************************** 01850000
               SPACE 2                                                 01890000
********************************************************************** 01930000
*  READ FUNCTION SECTION                                               01970000
********************************************************************** 02010000
               ORG   EXPRM                                             02050000
EXPREA_INBUF   DS    A            ADDR OF BUFFER TO BE PROCESSED       02090000
EXPREA_IBUFSIZE DS   F            SIZE OF INPUT BUFFER PASSED TO EXIT  02130000
EXPREA_OUTBUF  DS    A            ADDR OF BUFFER TO BE FILLED BY EXIT  02170000
EXPREA_OBUFSIZE DS   F            SIZE OF OUTPUT BUFFER PASSED TO EXIT 02210000
EXPREA_FLAG1   DS    X            FLAGS                                02250000
EXPREA_STRING1 EQU   X'80'        DATA IS FROM CLIENT MATCHING STRING 1 02290000
EXPREA_STRING2 EQU   X'40'        DATA IS FROM CLIENT MATCHING STRING 2 02330000
               DS    XL3          RESERVED                             02370000
EXPREA_RACFID  DS    CL8          DEFAULT USER ID FOR RACF             02410000
EXPREA_NAMEID  DS    0CL16        CLIENT ID STRUCTURE                  02450000
EXPREA_FAMILY  DS    H            CLIENT FAMILY TYPE                   02490000
EXPREA_PORT    DS    H            CLIENT PORT                          02530000
EXPREA_ADDRESS DS    F            CLIENT IP ADDRESS                    02570000
EXPREA_RESERVE DS    CL8          RESERVED                             02610000
               ORG   EXRET                                             02650000
*   RETURNED DATA                                                      02690000
EXPREA_RETCODE DS    F            RETURN CODE RETURNED BY EXIT         02730000
*                                 0 = READ CALL WAS SUCCESSFUL         02770000
*                                 4 = ERROR, SEND OUTBUF, THEN CLEANUP 02810000
*                                 8 = ERROR, CLEANUP                   02850000
```

```
EXPREA_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          02890000
EXPREA_DATALEN DS    F           SIZE OF DATA MOVED TO OUTBUF BY EXIT  02930000
EXPREA_UFLAG1  DS    X           USER FLAG                             02940000
               DS    XL3         RESERVED                              02950000
               SPACE 2                                                 02970000
*********************************************************************** 03010000
*  XMIT FUNCTION SECTION                                              03050000
*********************************************************************** 03090000
               ORG   EXPRM                                             03130000
EXPXMT_INBUF   DS    A           ADDR OF BUFFER TO BE PROCESSED        03170000
EXPXMT_IBUFSIZE DS   F           SIZE OF INPUT  BUF PASSED TO EXIT     03210000
EXPXMT_OUTBUF  DS    A           ADDR OF BUFFER WITH PROCESSED DATA    03250000
EXPXMT_OBUFSIZE DS   F           SIZE OF OUTPUT BUF PASSED TO EXIT     03290000
EXPXMT_FLAG1   DS    X           FLAGS                                 03330000
EXPXMT_STRING1 EQU   X'80'       DATA IS FROM CLIENT MATCHING STRING 1 03370000
EXPXMT_STRING2 EQU   X'40'       DATA IS FROM CLIENT MATCHING STRING 2 03410000
EXPXMT_UFLAG1  DS    X           USER FLAG                             03430000
               DS    XL2         RESERVED                              03450000
               ORG   EXRET                                             03490000
*   RETURNED DATA                                                      03530000
EXPXMT_RETCODE DS    F           RETURN CODE RETURNED BY EXIT          03570000
*                                0 = XMIT CALL WAS SUCCESSFUL          03610000
*                                4 = N/A                               03650000
*                                8 = ERROR, CLEANUP                    03690000
EXPXMT_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          03730000
EXPXMT_DATALEN DS    F           SIZE OF DATA MOVED TO OUTBUF BY EXIT  03770000
               SPACE 2                                                 03810000
*********************************************************************** 03850000
*  TERM FUNCTION SECTION                                              03890000
*********************************************************************** 03930000
               ORG   EXRET                                             03970000
*   RETURNED DATA                                                      04010000
EXPTRM_RETCODE DS    F           RETURN CODE RETURNED BY EXIT          04050000
EXPTRM_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          04090000
*********************************************************************** 04130000
*  EXER FUNCTION SECTION                                              04170000
*********************************************************************** 04210000
               ORG   EXPRM                                             04250000
EXPXER_OUTBUF  DS    A           ADDR OF BUFFER WITH PROCESSED DATA    04290000
EXPXER_OBUFSIZE DS   F           SIZE OF OUTPUT BUF PASSED TO EXIT     04330000
EXPXER_FLAG1   DS    X           FLAGS                                 04370000
EXPXER_STRING1 EQU   X'80'       DATA IS FROM CLIENT MATCHING STRING 1 04410000
EXPXER_STRING2 EQU   X'40'       DATA IS FROM CLIENT MATCHING STRING 2 04450000
EXPXER_UFLAG1  DS    X           USER FLAG                             04470000
               DS    XL2         RESERVED                              04490000
EXPXER_CODE    DS    F           RETURN CODE PASSED TO EXIT BY CALLER  04530000
*                                0 = N/A                               04570000
*                                4 = ERROR IN THE OUTPUT BUFFER FROM   04610000
*                                    THE PREVIOUS 'READ' OR 'XMIT'     04650000
EXPXER_REASON  DS    F           REASON CODE PASSED TO EXIT BY CALLER  04690000
*                                20 = SEGMENT LENGTH ERROR             04730000
*                                24 = MISSING FIC                      04770000
*                                28 = MISSING LIC                      04810000
*                                INDICATING PROBLEM WITH DATA RETURNED 04850000
*                                BY EXIT.                              04890000
               ORG   EXRET                                             04930000
*   RETURNED DATA                                                      04970000
EXPXER_RETCODE DS    F           RETURN CODE RETURNED BY EXIT          05010000
*                                0 = N/A                               05050000
*                                4 = ERROR, SEND OUTBUF, THEN CLEANUP  05090000
*                                8 = ERROR, CLEANUP                    05130000
EXPXER_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          05170000
EXPXER_DATALEN DS    F           SIZE OF DATA MOVED TO OUTBUF BY EXIT  05210000
*********************************************************************** 05250000
*  LENGTH OF PARAMETER LIST AREA                                      05290000
*********************************************************************** 05330000
               ORG   EXPRM                                             05370000
```

```
               DS    CL100        ENSURE 100 BYTE PARAMETER AREA        05410000
EXPRM_LEN      EQU   *-EXPRMBGN   LENGTH OF PARMS                       05450000
EXPRM_END      EQU   *            END OF DSECT                          05490000
               MEND                                                     05530000
```

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.

```
        MACRO                                                           00010000
        HWSEXPRM                                                        00050000
        GBLB  &HWSEXPRM 1ST TIME SW FOR MAPPING DSECT                   00090000
        AIF   (NOT &HWSEXPRM).AREAGO                                    00130000
        MEXIT                                                           00170000
.AREAGO  ANOP  ,                                                        00210000
&HWSEXPRM SETB  1                                                       00250000
.*                                                                      00290000
*********************************************************************** 00330000
***                                                                 *** 00370000
***      HWSEXPRM     - EXIT PARAMETER LIST MAPPING                 *** 00410000
***                     MAPS THE FIELDS POINTED TO BY REGISTER 1    *** 00450000
***                     ON ENTRY TO EXTERNAL EXITS.                 *** 00490000
***                                                                 *** 00530000
***            RESIDENCY: HWS REGION - PRIVATE, KEY 7               *** 00570000
***                                                                 *** 00610000
*********************************************************************** 00650000
               SPACE 1                                                  00690000
HWSEXPRM      DSECT              PARAMETER LIST DSECT                   00730000
EXPRMBGN      EQU   *                                                   00770000
*********************************************************************** 00810000
*     MAPPING FOR ALL FUNCTIONS                                         00850000
*********************************************************************** 00890000
EXPRM_FUNCTION DS    CL4          'INIT' INITIALIZE                     00930000
*                                 'READ' PROCESS INPUT DATA FROM CLIENT 00970000
*                                 'XMIT' PROCESS OUTPUT DATA FOR CLIENT 01010000
*                                 'TERM' PERFORM TERMINATION            01050000
*                                 'EXER' ERROR FOUND IN EXIT DATA       01090000
EXPRM_TOKEN    DS    A            ADDRESS OF 1K BUFFER FOR EXIT'S USE   01130000
EXPRM          EQU   *                                                   01170000
               DS    CL60         INPUT AREA                            01210000
EXRET          EQU   *                                                   01250000
               DS    CL40         RETURN DATA AREA                      01290000
               SPACE 2                                                  01330000
*********************************************************************** 01370000
*  INIT FUNCTION SECTION                                                01410000
*********************************************************************** 01450000
               ORG   EXRET                                              01490000
*   RETURNED DATA                                                       01530000
EXPINI_RETCODE DS    F            RETURN CODE RETURNED BY EXIT          01570000
*                                 0 = INIT CALL WAS SUCCESSFUL          01610000
*                                 4 = INIT CALL FAILED                  01650000
EXPINI_RSNCODE DS    F            REASON CODE RETURNED BY EXIT          01690000
EXPINI_STRING1 DS    CL8          CLIENT IDENTIFYING STRING 1           01730000
EXPINI_STRING2 DS    CL8          CLIENT IDENTIFYING STRING 2           01770000
EXPINI_MAXBUF  DS    F            MAX OUTPUT BUFFER SIZE                01810000
*********************************************************************** 01850000
               SPACE 2                                                  01890000
*********************************************************************** 01930000
*  READ FUNCTION SECTION                                                01970000
*********************************************************************** 02010000
               ORG   EXPRM                                              02050000
EXPREA_INBUF   DS    A            ADDR OF BUFFER TO BE PROCESSED        02090000
EXPREA_IBUFSIZE DS   F            SIZE OF INPUT BUFFER PASSED TO EXIT   02130000
EXPREA_OUTBUF  DS    A            ADDR OF BUFFER TO BE FILLED BY EXIT   02170000
```

```
EXPREA_OBUFSIZE DS   F           SIZE OF OUTPUT BUFFER PASSED TO EXIT  02210000
EXPREA_FLAG1   DS    X           FLAGS                                 02250000
EXPREA_STRING1 EQU   X'80'       DATA IS FROM CLIENT MATCHING STRING 1 02290000
EXPREA_STRING2 EQU   X'40'       DATA IS FROM CLIENT MATCHING STRING 2 02330000
               DS    XL3         RESERVED                              02370000
EXPREA_RACFID  DS    CL8         DEFAULT USER ID FOR RACF              02410000
EXPREA_NAMEID  DS    0CL16       CLIENT ID STRUCTURE                   02450000
EXPREA_FAMILY  DS    H           CLIENT FAMILY TYPE                    02490000
EXPREA_PORT    DS    H           CLIENT PORT                           02530000
EXPREA_ADDRESS DS    F           CLIENT IP ADDRESS                     02570000
EXPREA_RESERVE DS    CL8         RESERVED                              02610000
               ORG   EXRET                                             02650000
*   RETURNED DATA                                                      02690000
EXPREA_RETCODE DS    F           RETURN CODE RETURNED BY EXIT          02730000
*                                0 = READ CALL WAS SUCCESSFUL          02770000
*                                4 = ERROR, SEND OUTBUF, THEN CLEANUP  02810000
*                                8 = ERROR, CLEANUP                    02850000
EXPREA_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          02890000
EXPREA_DATALEN DS    F           SIZE OF DATA MOVED TO OUTBUF BY EXIT  02930000
EXPREA_UFLAG1  DS    X           USER FLAG                             02940000
               DS    XL3         RESERVED                              02950000
               SPACE 2                                                 02970000
********************************************************************** 03010000
*  XMIT FUNCTION SECTION                                              03050000
********************************************************************** 03090000
               ORG   EXPRM                                             03130000
EXPXMT_INBUF   DS    A           ADDR OF BUFFER TO BE PROCESSED        03170000
EXPXMT_IBUFSIZE DS   F           SIZE OF INPUT  BUF PASSED TO EXIT     03210000
EXPXMT_OUTBUF  DS    A           ADDR OF BUFFER WITH PROCESSED DATA    03250000
EXPXMT_OBUFSIZE DS   F           SIZE OF OUTPUT BUF PASSED TO EXIT     03290000
EXPXMT_FLAG1   DS    X           FLAGS                                 03330000
EXPXMT_STRING1 EQU   X'80'       DATA IS FROM CLIENT MATCHING STRING 1 03370000
EXPXMT_STRING2 EQU   X'40'       DATA IS FROM CLIENT MATCHING STRING 2 03410000
EXPXMT_UFLAG1  DS    X           USER FLAG                             03430000
               DS    XL2         RESERVED                              03450000
               ORG   EXRET                                             03490000
*   RETURNED DATA                                                      03530000
EXPXMT_RETCODE DS    F           RETURN CODE RETURNED BY EXIT          03570000
*                                0 = XMIT CALL WAS SUCCESSFUL          03610000
*                                4 = N/A                               03650000
*                                8 = ERROR, CLEANUP                    03690000
EXPXMT_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          03730000
EXPXMT_DATALEN DS    F           SIZE OF DATA MOVED TO OUTBUF BY EXIT  03770000
               SPACE 2                                                 03810000
********************************************************************** 03850000
*  TERM FUNCTION SECTION                                              03890000
********************************************************************** 03930000
               ORG   EXRET                                             03970000
*   RETURNED DATA                                                      04010000
EXPTRM_RETCODE DS    F           RETURN CODE RETURNED BY EXIT          04050000
EXPTRM_RSNCODE DS    F           REASON CODE RETURNED BY EXIT          04090000
********************************************************************** 04130000
*  EXER FUNCTION SECTION                                              04170000
********************************************************************** 04210000
               ORG   EXPRM                                             04250000
EXPXER_OUTBUF  DS    A           ADDR OF BUFFER WITH PROCESSED DATA    04290000
EXPXER_OBUFSIZE DS   F           SIZE OF OUTPUT BUF PASSED TO EXIT     04330000
EXPXER_FLAG1   DS    X           FLAGS                                 04370000
EXPXER_STRING1 EQU   X'80'       DATA IS FROM CLIENT MATCHING STRING 1 04410000
EXPXER_STRING2 EQU   X'40'       DATA IS FROM CLIENT MATCHING STRING 2 04450000
EXPXER_UFLAG1  DS    X           USER FLAG                             04470000
               DS    XL2         RESERVED                              04490000
EXPXER_CODE    DS    F           RETURN CODE PASSED TO EXIT BY CALLER  04530000
*                                0 = N/A                               04570000
*                                4 = ERROR IN THE OUTPUT BUFFER FROM   04610000
*                                    THE PREVIOUS 'READ' OR 'XMIT'     04650000
EXPXER_REASON  DS    F           REASON CODE PASSED TO EXIT BY CALLER  04690000
```

```
*                               20 = SEGMENT LENGTH ERROR             04730000
*                               24 = MISSING FIC                      04770000
*                               28 = MISSING LIC                      04810000
*                               INDICATING PROBLEM WITH DATA RETURNED 04850000
*                               BY EXIT.                              04890000
              ORG   EXRET                                             04930000
*   RETURNED DATA                                                     04970000
EXPXER_RETCODE DS    F          RETURN CODE RETURNED BY EXIT          05010000
*                               0 = N/A                               05050000
*                               4 = ERROR, SEND OUTBUF, THEN CLEANUP  05090000
*                               8 = ERROR, CLEANUP                    05130000
EXPXER_RSNCODE DS    F          REASON CODE RETURNED BY EXIT          05170000
EXPXER_DATALEN DS    F          SIZE OF DATA MOVED TO OUTBUF BY EXIT  05210000
********************************************************************** 05250000
*  LENGTH OF PARAMETER LIST AREA                                      05290000
********************************************************************** 05330000
              ORG   EXPRM                                             05370000
              DS    CL100        ENSURE 100 BYTE PARAMETER AREA       05410000
EXPRM_LEN     EQU   *-EXPRMBGN   LENGTH OF PARMS                      05450000
EXPRM_END     EQU   *            END OF DSECT                         05490000
              MEND                                                    05530000
```

Glossary

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

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

host Web service (HWS). 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 remote workstations.

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 internetwork. 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 HWS should join to the same group.

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