IMS TCP/IP OTMA Connection Programmer's Reference

itocpr-0002-01

March 13, 1998

Candace Garcia
IBM Corporation

IMS TCP/IP OTMA Connection Programmer's Reference

itocpr-0002-01

IMS TCP/IP OTMA Connection

Contents

Trademarks

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

IBM	IMS
MVS	OS/2
RACF	

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. IMS TCP/IP OTMA Connection commands

This section describes the following commands:

All IMS TCP/IP OTMA Connection commands must be immediately preceded on the command line of the MVS system console by the reply number of the outstanding HWS reply message (for example, *nn*HWSCMD where *nn* is the reply number).

CLOSEHWS

The CLOSEHWS command terminates the HWS.

Parameters

None

Usage All work that is currently in progress, or that is queued for processing, is completed before the HWS is terminated. No new work is accepted after this command has been entered and accepted.

The HWS shuts down in the following order:

1. All active units of work for TCP/IP clients/browsers are completed.
2. Communication between the HWS and IMS is terminated.
3. The HWS terminates.

Example

To close the HWS:

*nn*CLOSEHWS

OPENDS

The OPENDS command starts communication between the HWS and a datastore.

Parameters

datastore_id

Specifies the name of the datastore. This name must be defined to the HWS through the configuration member HWSCFGxx, and must match one of the IDs that is defined in the DATASTORE configuration statement or statements.

Usage Use this command to reestablish communication with a datastore after communication fails between the HWS and the datastore. For example, use

this command to restart communication when all activity for the datastore in the HWS is terminated, or after a STOPDS command has terminated communication with the datastore.

Use the VIEWDS command to display information about datastores if you are not sure about the activity of a particular datastore.

The OPENDS command does not affect a datastore that is already active or a datastore that is not defined to the HWS in the configuration member HWSCFGxx.

Example

To open communication to datastore IMSA:

*nn*OPENDS IMSA

OPENPORT

The OPENPORT command reestablishes HWS communication with TCP/IP to allow listening on TCP/IP ports.

Parameters

portid

Identifies the number of the port to be opened. This port number must match one of the port numbers that is defined in the PORTID substatement of the TCPIP configuration statement in the HWSCFGxx configuration member.

Usage Use this command to reestablish a TCP/IP connection to allow listening on a TCP/IP port. Use this command when communication stops between the HWS and a TCP/IP port, but the HWS has not terminated.

Example

To reestablish the TCP/IP connection between the HWS and port 9999 so that the HWS can listen on that port:

*nn*OPENPORT 9999

SETRACF

The SETRACF command turns on and off the RACF flag.

Parameters

ON/OFF

Identifies if the RACF flag is turned on or off.

Usage To enable or disable the RACF user identification and verification.

Example

To turn on the RACF:

*nn*SETRACF ON

STOPCLNT

The STOPCLNT command immediately terminates communication with a client using a specific TCP/IP port.

Parameters

portid

Identifies the port that the client is using for the TCP/IP connection with the HWS. This port number must match a port number that is defined in the PORTID substatement of the TCPIP configuration statement in the HWSCFGxx configuration member.

clientid

Specifies the name of the client (the client name is dynamically generated by the IMS Web Runtime component).

Usage Work currently in progress for that client is ended.

Use this command whenever a client is unable to accept response messages being sent to it, or when a client is waiting for a nonexistent response message (for example, when an error occurred that caused a response message to be lost before it was sent back to the client).

Use the VIEWPORT command to display the name and state of the client.

Example

To force the HWS to terminate communication with client CLIENT01, who is communicating with the HWS using port 9999:

```
nnSTOPCLNT 9999 CLIENT01
```

STOPDS

The STOPDS command immediately terminates communication between the HWS and a datastore.

Parameters

datastore_id

Specifies the name of the datastore. This name must match an ID that is defined in a DATASTORE configuration statement of the HWSCFGxx configuration member.

Usage Work currently in progress for a datastore is aborted and communications with that datastore and its threads are terminated. Messages that are queued for the datastore are released and the originator of the queued messages is notified. No new messages are accepted after the STOPDS command is accepted.

Use this command to release messages that are queued for an unavailable datastore or for a datastore whose queued work belongs to unavailable clients. It can also be used for any type of error situation that requires immediate termination of communication with a datastore.

Use the OPENDS command to open communication with the datastore at a later time.

Example

To stop communication to datastore IMSA:

```
nnSTOPDS IMSA
```

STOPPORT

The STOPPORT command immediately terminates listening on a TCP/IP port.

Parameters

portid

Identifies the number of the port on which listening is to stop. This port number must match one of the port numbers that is defined in the PORTID substatement of the TCPIP configuration statement in the HWSCFGxx configuration member.

Usage

Work currently in progress is allowed to continue for existing clients. Only the listening for new request messages on the port is terminated immediately. When existing work has completed, the port is no longer active.

Use the VIEWPORT command to display the state of the port and any clients using that port.

Example

To stop listening on port 9999:

*nn*STOPPORT 9999

VIEWDS

The VIEWDS command displays the current activity of a datastore.

Parameters

datastore_id

Specifies the name of the datastore for which information is to be displayed or ALL. If a datastore name is used, this name must match the ID parameter of a DATASTORE configuration statement of the HWSCFGxx configuration file and only the information for this datastore is displayed. If ALL is used, information for all datastores that are defined in a DATASTORE configuration statement in the HWSCFGxx configuration member is displayed.

Usage This command displays the current information for one or all datastores. The information displayed for each datastore is:

Datastore Name
Name of the datastore, as defined in the ID substatement of the DATASTORE configuration statement in the HWS configuration member HCTCFGxx.

Status
State of the datastore, ACTIVE, NOT ACTIVE or NOT DEFINED.

Group XCF group name for the group to which the HWS and IMS OTMA belong.

Member
HWS member name in the XCF group listed.

Target Member
IMS OTMA member name in the XCF group listed.

Example

To view the information for a single datastore, IMSA:

*nn*VIEWDS IMSA

To view the information for all datastores defined to the HWS:

*nn*VIEWDS ALL

VIEWHWS

The VIEWHWS command displays the current activity of the HWS.

Parameters

None.

Usage Information displayed includes:

HWS ID
Name of the HWS, as defined in the ID substatement of the HWS configuration statement in the HWSCFGxx configuration member.

Datastore Name
Name of the datastore, as defined in the ID substatement of the DATASTORE configuration statement in the HWSCFGxx configuration member or No active Datastores.

Status (Datastore)
State of the datastore, whether ACTIVE, NOT ACTIVE or NOT DEFINED.

Group XCF group name for the group to which the HWS and IMS OTMA belong.

Member
HWS member name in the XCF group listed.

Target Member
IMS OTMA member name in the XCF group listed.

Port Identifies the port or ports that are defined in the PORTID substatement of the TCPIP configuration statement in the HWSCFGxx configuration member or No active Ports.

Status (Port)
State of the port, whether ACTIVE or INACTIVE.

Client Specifies the name of the client (the client name is dynamically generated by the IMS Web Server) or NO active Clients.

Status (Client)
State of the client's thread, whether ACTIVE or IDLE.

You can use the VIEWDS command to display information for datastores only or the VIEWPORT command to display information for ports only.

Example

To view the HWS:

*nn*VIEWHWS

VIEWPORT

The VIEWPORT command displays the current activity of a port.

Parameters

portid

Specifies the number of the port for which information is to be displayed or ALL. If a port number is used, this number must match a port number that is defined in the PORT substatement of the TCPIP configuration statement in the HWSCFGxx configuration member and the information is displayed for this port

only. If ALL is used, information is displayed for all ports that are defined in the TCPIP configuration statement in the HWSCFGxx configuration member.

Usage Information displayed includes:

Port Identifies the port or ports that are defined in the PORTID substatement of the TCPIP configuration statement in the HWSCFGxx configuration member.

Status (Port)
State of the port, whether ACTIVE or INACTIVE.

Client Specifies the name of the client (the client name is dynamically generated by the IMS Web Runtime component).

Status (Client)
State of the client's thread, whether ACTIVE or IDLE.

Example

To view the information for a single port, 9999:

```
nnVIEWPORT 9999
```

To view the information for all ports defined to the HWS:

```
nnVIEWPORT ALL
```

Chapter 2. Sample Java client for the ITOC

JAVASAMP.exe is a self-extracting, compressed file that contains a sample Java client and a sample user exit for IMS TCP/IP OTMA Connection, along with documentation. JAVASAMP.exe is packaged with the IMS TCP/IP OTMA Connection in HWSMH*vrm*.exe, where *vrm* represents the level (version, release, modification) of IMS TCP/IP OTMA Connection. This sample Java client consists of code used to prepare a Java program that accesses IMS applications and data. Both IMS Version 5 OTMA and the IMS TCP/IP OTMA Connection are prerequisites to using the sample.

The client source code is written in Java. The following are also included in the JAVASAMP file:

- A .gif file (used by the applet)
- An .html file

The sample files are:

- ClientLauncher.java
- ClientLauncherApplet.java
- ClientLauncherFrame.java
- FrameInput.java
- CanvasAbout.java
- IMSCanvas.java
- SampleTran.java
- AboutDialog.java
- QuitDialog.java
- ClientLauncher.html
- ims2.gif

The documentation files are:

- JavaSampleChanges.text
- JavaSampleChanges.html
- JavaSampleInstallation.text
- JavaSampleInstallation.html

Downloading the sample programs

1. Following the download instructions, download HWSMH*vrm*.exe to an OS/2 or Windows platform.
2. Run HWSMH*vrm*.exe by entering HWSMHvrm *d:**outpath* at a command line prompt, where *d:**outpath* is the output directory where the expanded files will be placed.
3. Move the JAVASAMP.exe file from the output directory to a Windows NT platform. The remaining files in the output directory are used for installing the IMS TCP/IP OTMA Connection.
4. Run JAVASAMP.exe by entering `JAVASAMP` *d:**outpath* at a Windows NT command line prompt, where *d:**outpath* is the output directory in which the expanded files will be placed.

Note: 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.

5. Following the instructions in the next section, install the Java client.

Installing the sample Java client

The following procedures are specific for Windows NT; however, they can serve as a reference for installation on the other platforms.

1. Create an NTFS directory to accommodate the Java files and place all the downloaded client files in this directory.
2. Modify the source code to match your environment. (See “Modifying the sample Java client”.)
3. Enter `javac *.java` at the command prompt to create the 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. However, the Java client assumes that the maximum number of output segments is 5. If the maximum of 5 output segments in one transaction is not acceptable, you must also change the SampleTran.java file to increase the number of segments, according to your needs.

HWS requires that all active clients have unique client LUNAMEs that, for the sample Java client, are taken from the Userid 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 Userid will be unique for each active client at any given time or ensure in some other way that the Userid for each active client is unique. For test purposes, just be sure that you use a unique Userid for each Java client when you press the Submit button.

Running the sample Java client

You can use the sample Java program as a Java applet or a Java application, depending on how you start it. The overhead of implementing Java clients in this way is minor, compared with the benefits.

In order for Java to run correctly on NT when a Netscape browser is used, you must define a user variable in your NT environment, as follows:

```
CLASSPATH=.;
```

Running the client as a Java application

In order to run the sample Java client as an application, you must have installed the Java Development Kit on your workstation. After the sample Java source code has been compiled, enter `java ClientLauncher` at a command line prompt to run the sample client as a Java application.

Running a client applet locally

Open ClientLauncher.html as a local file from a Java-enabled Web browser that supports JDK 1.1.3 and an applet for socket codes.

Running the client as a remote applet

Before you can run the sample Java client as a remote applet, your Internet server must be running on the host where the IMS TCP/IP OTMA Connection is running (must have the same MVS image). Copy all class, .gif, and .html files to your MVS Internet Connection Server html directory in the same MVS image where the IMS TCP/IP OTMA Connection is running, and then bring up the ClientLauncher (ClientLauncher.html) in a Java-enabled Web browser.

ClientLauncher starts a new Transaction Input client frame every time you click the Start A Client button. If you start more than one client windows, the newer windows hide the existing client window. All of these client frame windows run independently, so you must close each window individually if you are running the sample Java client as an applet. If you are running the sample client as a Java application, you can either close the client frame windows individually (as you must do if the sample Java client is running as an applet) or all at the same time by closing the original window (the ClientLauncher running as an application). When you close the ClientLauncher application, all client frame windows that were generated by that application close automatically.

Chapter 3. HWSWEB00 User Exit for ITOC

The IMS Web user exit routine, HWSWEB00, allows you to call IMSSEC, issue the RACF function in this user exit routine, or use the ITOC user RACF function.

```
                HWSWEB00 TITLE 'HWSWEB00 IMS Web user exit'
*********************************************************************
*        Disclaimer
*
*        This user exit program is provided for tutorial purposes only.
*        A complete handling of error conditions has not been shown or
*        attempted, and this program has not been submitted to formal
*        IBM testing. This program is distributed on an 'AS IS' basis
*        without any warranties either expressed or implied.
*
*********************************************************************
*-------------------------------------------------------------------------------
*
* Module name:
*     HWSWEB00
* Entry point:
*     HWSWEB00
*
* Function:
*    This exit is called by the IMS Web IMS TCP/IP OTMA connection
*    to process client data destined for IMS and output data from IMS.
*    This exit will check the userid and group name in OTMA headers
*    (mapped by macro HWSOMPFX ) and provide correct password
*    in userdata for ITOC to do the RACFINIT.
*
*
*-------------------------------------------------------------------------------------------
 SPACE
*********************************************************************
*
* Registers on entry:
*
*    R1    Parameter list passed on entry:
*           Mapped by DSECT HWSEXPRM
*    R13  Save area address
*    R14  Caller's return address
*    R15  HWSWEB00's entry point
*
* Registers on exit:
*    R0-R15  Restored
*********************************************************************
*********************************************************************
*
* Change Activity:
*
*********************************************************************
   EJECT
HWSWEB00 CSECT ,  ENTRY POINT
HWSWEB00 AMODE 31 Addressing mode is 31-bit
HWSWEB00 RMODE ANY Residency mode is anywhere in virtual storage
*
    STM   R14,R12,12(R13) Save caller's regs
    LR     R12,R15 Set module base register
    USING HWSWEB00,R12 USING on 1st base reg
    LR R10,R1 Load parameter list address
    USING HWSEXPRM,R10 Map the parameter list
    L R8,EXPRM_TOKEN Get address of 1k user area
    USING SAMPWORK,R8 Map the exit user area
    LA R15,SAM_SAVE Point to our save area
    ST R13,4(,R15) Save caller's save area
    LR R13,R15 Set R13 to Exit's save area
```

```
*  Find the function and branch to appropriate routine
   CLC  EXPRM_FUNCTION,=CL4'INIT' Calling INIT subroutine ?
   BE  INIT0000 Yes, do INIT
   CLC  EXPRM_FUNCTION,=CL4'READ' Calling READ subroutine ?
   BE  READ0000 Yes, do READ
   CLC  EXPRM_FUNCTION,=CL4'XMIT' Calling XMIT subroutine ?
   BE  XMIT0000 Yes, do XMIT
   CLC  EXPRM_FUNCTION,=CL4'TERM' Calling TERM subroutine ?
   BE  TERM0000 Yes, do TERM
   CLC  EXPRM_FUNCTION,=CL4'EXER' Calling EXER subroutine ?
   BE  EXER0000 Yes, do EXER
   DS  0H
   DC  C'Subroutine not found' Just abend
   EJECT
*********************************************************************
* Subroutine to pass back to caller the EBCDIC and
* ASCII string tokens identifying the client. These
* 8-byte string tokens are used by the caller to match
* against the first segment of client data.
*********************************************************************
INIT0000 DS  0H
  MVC  EXPINI_STRING1,EBCDICID Move in EBCDIC token
  MVC  EXPINI_STRING2,ASCIIID  Move in ASCII token
  L  R5,MAXBUFSZ Get Maximum output buffer size
  ST R5,EXPINI_MAXBUF Move in Maximum output buffer size
  SLR  R1,R1 Zero out R1
  ST  R1,EXPINI_RETCODE Return good return code
  B  RETURN return
  EJECT
*********************************************************************
* Subroutine to process IMS Web client data for IMS/OTMA
* handling.
*********************************************************************
READ0000 DS  0H
   L  R11,EXPXMT_INBUF Get the address of input data
   L  R9,EXPXMT_OUTBUF Get the address of output data
   USING INPUTMSG,R11 Map the input message
   L  R1,EXPXMT_IBUFSIZE Get input data length
   S  R1,=F'12' Adjust the length
   LA R14,INM_DATA Get address of input data
   LR  R2,R9 Get address of output data
   LR  R3,R1 Get length to copy
   LR  R15,R3 Dup length to copy
   MVCL  R2,R14 Move data
   ST  R1,EXPREA_DATALEN  SET OUTPUT DATA LENGTH
   SR  R1,R1 Clean r1
   ST  R1,EXPREA_RETCODE Clean up return code
   ST  R1,EXPREA_RSNCODE Clean up reason code
   B  RETURN Return
   DROP  R11
   EJECT
*********************************************************************
* Subroutine to add identification to the
* IMS transaction output data.
*********************************************************************
XMIT0000 DS   0H
   L  R11,EXPXMT_INBUF Get the address of input data
   L  R9,EXPXMT_OUTBUF Get the address of output data
   USING OUTPTMSG,R9 Map the output message
   MVC  OUM_ID,EBCDICID Set EBCDICID
   L  R1,EXPXMT_IBUFSIZE Get input data length
   L  R1,EXPXMT_IBUFSIZE Get input data length
   LA  R2,OUM_DATA Get address of output data
   LR  R14,R11 Get address of input data
   LR  R3,R1 Get length to copy
   LR  R15,R3 dup length to copy
   MVCL  R2,R14 Move data
```

```
   LA  R1,12(R1) Add default output data length
   ST  R1,EXPXMT_DATALEN Set output data length
   ST  R1,OUM_LEN Dup output data length
   SR  R1,R1 Clean r1
   ST  R1,EXPXMT_RETCODE Clean up return code
   ST  R1,EXPXMT_RSNCODE Clean up reason code
   B  RETURN Return
   DROP  R9
   EJECT
*************************************************************************
* Subroutine called when HWS is shutting down.
* Currently their is nothing to clean up, so just return.
*************************************************************************
TERM0000 DS  0H
   SLR  R1,R1 Zero out R1
   ST  R1,EXPTRM_RETCODE Return good return code
   B  RETURN Return
   EJECT
*************************************************************************
* Subroutine called when an error has been detected by
* IMS TCP/IP OTMA Connection.
*************************************************************************
EXER0000 DS  0H
   SLR  R1,R1 Zero out R1
   ST  R1,EXPXER_RETCODE Return code = 0
   ST  R1,EXPXER_RSNCODE Reason code = 0
   B  RETURN Return
   EJECT
*************************************************************************
* Return to caller
*************************************************************************
RETURN   DS  0H
   L  R13,4(,R13) Get caller's save area
   LM  R14,R12,12(R13) Restore R14-R12
   BSM  0,R14 Return to caller
   DROP  R8,R10,R12
   EJECT
*-------------------------------------------------------------
*Constants*----------------------------------------------------
EBCDICID DC  C'*HWSWEB*' EBCDIC id string
ASCIIID  DC  X'2A4857535745422A' ASCII id string
MAXBUFSZ DC  F'72000' Maximum output buffer size
*-------------------------------------------------------------
* Dsects
*-------------------------------------------------------------
   EJECT
INPUTMSG DSECT
INM_LEN  DS  F Total message length
INM_ID  DS   CL8 Message identifier
INM_DATA DS  0C Data
INM_END equ  * 1st data segment
INMLEN equ *-INM_LEN Dsect length
*
*
SAMPWORK DSECT
SAM_SAVE DS  18F Save Area
SAM_END equ  *
SAMLEN equ  *-SAM_SAVE Dsect length
*
DATASEG  DSECT
DATA_LL DS H Data segment length
DATA_ZZ DS XL2 ZZ
DATA_APP DS 0C Data begin
*
OUTPTMSG DSECT
OUM_LEN DS F Total message length
OUM_ID DS CL8 Message identifier
```

```
OUM_DATA DS 0C 1st data segment
OUM_END equ *
OUMLEN equ *-OUM_LEN Dsect length
*
   HWSEXPRM
   HWSOMPFX
* REGISTER EQUATES
* REGISTER EQUATES
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
RA EQU 10
R10 EQU 10
RB EQU 11
R11 EQU 11
RC EQU 12
R12 EQU 12
RD EQU 13
R13 EQU 13
RE EQU 14
R14 EQU 14
R14 EQU 14
RF EQU 15
R15 EQU 15
   END HWSWEB00
```