High Speed Connector Driver readme

NOTE: Before using this information and the product it supports, read the general information under "Notices" in this document.

Contents

This file includes the following information:

1.0 Introduction

The purpose of this driver is to implement a server on the z/TPF system and on Linux on z Systems, and to provide an example of how to issue the tpf_send_message function, which is provided by high speed connector support.

The driver package includes the following programs:

- connserver.c − Implements a basic Linux on Z server by creating a listener socket for incoming connections. When an incoming connection is accepted, a new thread is created to process send and read requests for that new accepted socket indefinitely until the socket is closed. Has options for SSL or non−SSL connections. Currently, does not support client authentication.
- qhsc.cpp − Implements a client driver for the z/TPF system that issues the tpf_send_message function.
- qhss.c − Implements a basic z/TPF server that opens a socket to listen for incoming connections and issues an activate_on_receipt function to program QHSR to send and receive messages for the incoming connection.
- qhsr.c − Issues a read and send request if a response is requested, and then issues an activate_on_receipt function to itself (program QHSR) to read and send messages, indefinitely on the socket until it is closed.
- qhsu.c − Implements a z/TPF server to listen for incoming connections similar to qhss.c, difference being that qhsu.c assumes an SSL session has been established, thus issues an SSL_aor to program QHST to send and receive messages for the incoming connection.
- qhst.c − Issues a SSL_read and SSL_write request if a response is requested, and issues an SSL_aor to itself (program QHST). Functions similarly to program QHSR.
- qhss.h − Header for use by qhss.c, qhsu.c, qhst.c
- qhsc.mak

```
qhss.mak
qhsr.mak
qhsu.mak
qhst.mak
connserver.mak
conn.cntl
config8500.cfg   – Sample server configuration file for z/TPF server (QHSS).
config8443.cfg   – Sample server configuration file for z/TPF server (QHSU).
eptgrp.ept.xml   – Sample High Speed Connection group configuration file.
csclissl.ept.xml – Sample High Speed Connection group configuration file
                   with TLS on.
csclissl.conf – Configuration file that has information related to the SSL
                connection such as certificates paths, ciphers, etc.

Note: Configuration file containing information about the certificate,
certificate authority, and keys corresponding connserver with SSL
needs to be specified in csclissl.conf. For testing, the file was
placed under /etc/ssl/csclissl.conf.
```

2.0 Change history

```
2016Oct26  Initial release
2018Aug27  Configured High Speed Connector drivers for SSL
2019Apr20  Updates to connserver
```

3.0 Prerequisites

The following list provides the required release levels:
z/TPF PUT 13 or later, with APAR PJ43832 (High Speed Connector) applied
APAR PJ45258 enables TLS for High Speed Connector along with ENH HTTP CLient, and REST.

Dependencies: OpenSSL is required to build the High Speed Connector drivers connserver.c, QHSU, and QHST. If the user chooses not to use connserver with SSL, this can be specified using the –t flag option (see below) when issuing the command on Linux to start up the connserver. However, OpenSSL is still required for a clean compile. QHSU and QHST which are the SSL variants of QHSS and QHSR have OpenSSL as a dependency – use the latter to use the High Speed Connector driver without SSL on TPF. The OpenSSL version we used to build is version 1.0.2j.

4.0 Installing the High Speed Connector Driver

1) Use FTP to transfer the tar file (conn.tar.gz) to your Linux on z system. This file can be placed in any directory as a holding location, for example, /tmp/ztpftar
2) Create a root directory to hold the unpacked files, for example, /ztpfdrvs
3) Extract the source code from the tar file by entering the following commands:

```
cd tmp/ztpfdrvs
tar –xvzf conn.tar.gz –C ztpfdrvs
```

The driver source files will be extracted in the following directory structure:

```
conn/connserver.c
conn/connserver.mak
conn/qhsc.cpp
conn/qhsc.mak
conn/qhss.c
conn/qhss.mak
conn/qhss.h
conn/qhsr.c
conn/qhsr.mak
conn/qhsu.c
conn/qhsu.mak
conn/qhst.c
conn/qhst.mak
```

```
conn/conn.cntl
conn/config8443.cfg
conn/config8500.cfg
conn/eptgrp.ept.xml
conn/csclissl.ept.xml
conn/csclissl.conf
```

4) Create a maketpf.cfg file with the following contents:

```
APPL_ROOT := /ztpfdrvs
TPF_ROOT := /ztpf
TPF_BSS_NAME := BSS
#TPF_SS_NAME := WP
#USER_VERSION_CODE :=
```

   a) Set APPL_ROOT to the directory that contains the driver source code extracted.
   b) Set TPF_ROOT to the directory that contains the z/TPF source code.
   c) Set TPF_BSS_NAME to the basic subsystem name of your TPF system. By default, this is set to BSS.
   d) Optional: Set the TPF_SS_NAME to the subsystem name.
   e) Optional: Set USER_VERSION_CODE to any desired two−character string for user version code for the shared objects built. By default, this value is set to null.

   For more information about these variables, enter man maketpf on your Linux on z build system.

5) Build the USRSTUB program after adding the CONN driver control file to your user control file.

   a) Add the following line to your user control file base/cntl/usr.cntl:

      include conn/conn.cntl

   b) Build the USRSTUB program to generate stubs for the driver programs using the following command:

      maketpf USRSTUB −f

6) Run the maketpf tool with the accompanied control file (conn.cntl) to assemble, compile, and link the driver programs:

   bldtpf /ztpfdrvs/conn/conn.cntl

4.1 Installing the Linux on z Server Driver

1) Move the connserver executable file to the Linux on z directory of your choice.

2) Start the driver. See section 6.0.

4.2 Installing the z/TPF Server Driver

1) Use the standard load procedure to transfer and load the following driver programs to the z/TPF system:

   QHSS.so QHSR.so (for non−SSL)
   QHSU.so QHST.so (for SSL)

   For more information about building and loading programs to the z/TPF system, see the Program management topic in the z/TPF product documentation in IBM Knowledge Center.

2) Use FTP to transfer a server configuration file with the sample format in the

   config8500.cfg file to the etc/conn directory on the z/TPF system.

3) Enter the following z/TPF command. This example uses port 8500/8443 and

HSCServ as the example server name:

(non-SSL)
Step 1: Add the server
```
zinet add s-HSCServ port-8500 prot-tcp pgm-qhss model-aoa2 state-cras
      act-auto
```

Step 2: Start the server
```
zinet start s-HSCServ
```

(SSL)
Step 1: Add the server
```
zinet add s-HSCServ port-8443 prot-tcp pgm-qhsu model-ssl state-cras
      act-auto
```

Step 2: Update/Modify configuration file
INETD configuration file in /etc/inetd/servername.conf needs to be updated to include SSL information pertaining to ciphers, certificate and key location, key type, and more.

Step 3: Start the server
```
zinet start s-HSCServ
```

4.3 Installing the z/TPF Client Driver

1) Use the standard load procedure to transfer and load the following driver programs to the z/TPF system:

   QHSC.so

   Additionally, client endpoints should be configured and loaded to the z/TPF system as well. Sample endpoint files are provided (csclient.ept.xml, csclissl.ept.xml).

   For more information about building and loading programs to the z/TPF system, see the Program management topic in the z/TPF product documentation in IBM Knowledge Center.

5.0 Customizing the High Speed Connector Driver

1) You can modify the QHSS or QHSU program to specify a different directory for the server configuration files.
   You can modify the QHSC, QHSS, QHSU, and connserver programs to alter the default values for the required parameters.
   You can modify the values specified in the .cfg files. To use a different port than the example provided, you must rename the configXXXX.cfg file where XXXX is the port number you choose. When running the server program, specify this port as a parameter so the program knows to look for this .cfg file.

2) Update the base/rt/cvzz.asm program (or the tool that executes driver programs) to make an entry for this driver. The shared object QHSC is the main entry point for this driver.

3) Build and load the updated CVZZ program to the z/TPF system.

6.0 Running the High Speed Connector Driver

To start the QHSS server, enter the following command:

```
zinet add s-connXXXX port-XXXX prot-tcp pgm-qhss model-aoa2 state-cras
      act-auto
```

s-connXXXX specifies the server name, where XXXX is a 4-digit number that represents the server's port number in the configXXXX.cfg configuration file

name, and the cfg file contains the server's parms and is located in the etc/conn directory. To start analogous QHSU server, replace pgm-qhss with

```
        pgm-qhsu and model-aoa2 with model-ssl.

    To issue QHSC commands, enter the following command:

        ztest conn ENDGroup-u+++++++ Read-a REQuest-d++++++++ REQSize-d++++++
                   [RESPSize-d++++++] [RRobin-a] Timeout-d++++++ SYSBUFF-a

        Required parameters description:
            ENDGroup-The 8 alphanumeric character name of the endpoint group to send

                     messages to
            Read-[Y/N], Should the client listen for a response
                 (if Y, RESPSize must be specified)
            REQuest-[1-999999999] Number of requests to send
            REQSize-Number of bytes each request will be
            Timeout-[1-600000] Milliseconds to timeout
            Sysbuff-[Y/N], If response requested, specify Y if system will provide
                    response buffer, N if user
        Optional parameters description:
            RESPSize-Number of bytes of the response message
            RRobin-[Y/N] request that the server uses round robin when using
                endpoints to respond

    To start connserver, enter the following command:
        nohup ./connserver <OPTIONS> &

        Where options contains:
            -i SERVER_ADDRESS
                Where SERVER_ADDRESS is a valid IPv4 address.
                Default value is 0.0.0.0
            -p PORT
                Where PORT is the port for the server to listen on
                0 <= PORT <= 65535
                This option is REQUIRED.
            -d RESPONSE_DELAY
                Must be an integer between 0 and 600000, inclusive
                Default value is 0.
            -s SEND_BUFFER_SIZE
                Size of buffer in bytes in which messages are sent.
                Only used if RESP_REQ=1
                Default value is 8192.
                Must be an integer between 512 and 1048576, inclusive.

            -r RCV_BUFFER_SIZE
                Size of buffer in bytes in which messages are received from the
                    tpf_send_message function.
                Default value is 8192.
                Must be an integer between 512 and 1048576, inclusive.

            -q RESP_REQ
                Defines whether or not this server will send a response when a
                message is received.
                Default value is 0.
                Must be either 0 or 1 -- 0 or no, 1 for yes.
            -m RESP_MSG_SIZE
                The size, in bytes, of the response message sent by connserver if
                RESP_REQ=1.
                Default value is 4096.
            -t
                Having this flag will turn SSL on.
                By default, without the -t flag SSL is off.

                To use connserver with SSL, it is necessary to have a valid
                certificate/key pair placed on Linux and pointed to by connserver.c.
                To change the server key pair to point to your own, update the
                pathnames for HOME, CERT_FILE, KEY_FILE in source file connserver.c.

                Such change may require a change to KEY_PASSWD as well, if the key
                has a password. An additional update to CIPHER_LIST may also be
                necessary.
```

7.0 Known problems and workarounds

None

8.0 Other sources of information

z/TPF product documentation in IBM Knowledge Center

9.0 Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, ore) Optional: Set USER_VERSION_CODE to any desired two-character string features discussed in this document in other countries. Consult your local IBM representative for information on the products and

services currently available in your area. Any reference to an IBM product, program,

or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

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

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

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

IBM Director of Licensing

IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

9.1 Trademarks

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

9.2 Warranty

This package is provided on an "as is" basis. There are no warranties, express or implied, including the implied warranties of merchantability and fitness for a particular purpose. IBM has no obligation to provide service, defect correction, or any maintenance for the package. IBM has no obligation to supply any updates or enhancements for the package to you even if such are or later become available.