Prepared (also subject responsible if other)
ETH/RZX Péter Dimitrov +36 1 437 7244
Approved
ETH/RZXC (Tibor Csöndes)
Checked

No.
198 17-CNL 113 312 Uen
Date
2010-07-01
Rev
G
Reference
GASK2

HTTPmsg_CNL113312 Test Port for TTCN-3 Toolset with TITAN, User Guide

Contents

1 Introduction

1.1 Revision history

Date	Rev	Characteristics	Prepared
2004-01-23	PA1	First draft version	ETHECS
2004-01-29	A	Revised	ETHECS
2006-01-26	PB1	Messages with binary body added	ETHECS
2006-01-30	PB2	-lssl description added after review	ETHECS
2006-02-20	PC1	Multiple client handling added	ETHECS
2006-04-04	PC2	Revised	ETHECS
2007-01-15	PD1	Updated for TITAN R7	ETHBAAT
2007-03-08	D	Approved version	ETHBAAT
2007-10-05	PE1	Encoding-decoding functions added	ETHBAAT
2007-11-09	PE2	Editorial modification after internal review	ETHBAAT
2009-02-26	PE3	Added `server_backlog`	ETHECS
2009-04-01	PF1	Updated according to the Test Port API introduced in TITAN R7E	ECSAFEH
2010-04-23	PG1	Implementing HL79892 and HL79997	EPTEDIM

1.2 About this Document

1.2.1 How to Read this Document

This is the User's Guide for the HTTPmsg_CNL113312 (called HTTP from now on) test port. The HTTP test port is developed for the TTCN-3 Toolset with TITAN according to the Functional Specification [4]. This document is intended to be read together with Product Revision Information [3].

1.2.2 Prerequisite Knowledge

The knowledge of the TITAN TTCN-3 Test Executor [2] and the TTCN-3 language [1] is essential. Basic knowledge of the HTTP protocol is valuable when reading this document.

1.2.3 References

[1] ETSI ES 201 873-1 v3.1.1 (2005-06)
The Testing and Test Control Notation version 3; Part 1: Core Language

[2] 1/198 17-CRL 113 200 Uen
TITAN User Guide

[3] 109 21-CNL 113 312-3 Uen
HTTPmsg_CNL113312 Test Port for TTCN-3 Toolset with TITAN, Product Revision Information

[4] 155 17-CNL 113 312 Uen
HTTPmsg_CNL113312 Test Port for TTCN-3 Toolset with TITAN, Functional Specification

[5] RFC 2616
Hypertext Transfer Protocol – HTTP/1.1
http://www.ietf.org/rfc/rfc2616.txt

[6] OpenSSL toolkit
http://www.openssl.org

1.2.4 Abbreviations

ASP	Abstract Service Primitive
IUT	Implementation Under Test (HTTP 1.1 server or client)
RTE	Run-Time Environment
HTTP	Hypertext Transfer Protocol
SUT	System Under Test
SSL	Secure Sockets Layer
TTCN-3	Testing and Test Control Notation version 3

1.2.5 Terminology

-

1.3 System Requirements

In order to operate the HTTP test port the following system requirements must be satisfied:

- Platform: any platform supported by TITAN RTE, optional OpenSSL.
- TITAN TTCN-3 Test Executor R8A (1.8.pl0) or higher installed. For installation guide see [2]. Please note: This version of the test port is not compatible with TITAN releases earlier than R8A.
- The C compiler gcc version 2.95 or above is installed.
- The OpenSSL 0.9.7 or above is installed. See [6].
- The Abstract_Socket CNL 113 384, rev. R6A or later product has to be installed.

1.4 Fundamental Concepts

The test port establishes connection between the TTCN-3 test executor and the HTTP server or client trough a TCP/IP socket connection. The test port transmits and receives HTTP1.1 messages; see [4] and [5].

2 The Test Port

2.1 Overview

The HTTP test port offers HTTP message primitives and TCP connection control ASPs to the test suite in TTCN-3 format. The TTCN-3 definition of the HTTP messages and TCP notification ASPs can be found in a separate TTCN-3 module. This module should be imported into the test suite.

2.2 Installation

Since the HTTP test port is used as a part of the TTCN-3 test environment this requires TTCN-3 Test Executor to be installed before any operation of the HTTP test port. For more details on the installation of TTCN-3 Test Executor see the relevant section of [2].

When building the executable test suite intended to handle HTTP over SSL connections, the libraries compiled for the OpenSSL toolkit and the TTCN-3 Test Executor should also be linked into the executable.

Using and compiling OpenSSL is optional in the test port. See [6] for more information. OpenSSL libraries should be added to the Makefile generated by the TITAN executor see example in close 8.2.

2.3 Configuration

The executable test program behaviour is customized via the RTE configuration file. This is a simple text file, which contains various sections (e.g. [TESTPORT_PARAMETERS]) after each other. The usual suffix of the RTE configuration file is `.cfg`. For further information about the configuration file see [2].

2.3.1 HTTP test port parameters in the RTE configuration file

In the `[TESTPORT_PARAMETERS]` section you can specify parameters that are passed to the test ports. Each parameter definition consists of a component name, a port name, a parameter name and a parameter value. The component name can be either an identifier or a component reference (integer) value. The port and parameter names are identifiers while the parameter value always must be a charstring (with quotation marks). Instead of component name or port name (or both of them) the asterisk (“*”) sign can be used, which means “all components” or “all ports of the component“. More information about the RTE configuration file can be found in [2].

In the `[TESTPORT_PARAMETERS]` section the following parameters can be set for the HTTP test port. Parameters marked with bold fonts apply to **SSL** using HTTP test ports **only**. Parameter values are **case-sensitive**!

`use_notification_ASPs`

Enables receiving of 'Connect_result', 'Client_connected' and 'Listen_result' ASPs. Its default value is "no" in order to provide backward-compatibility for test suites using the port versions older than R2A.

`server_backlog`

The parameter can be used to specify the number of allowed pending (queued) connection requests on the port the server listens. It is optional in server mode and not used in client mode. The default value is "1024".

`http_debugging`

Enables detailed debugging in the test port. It has only effect when TTCN_DEBUG is also set within the logging parameters of the configuration file. Its default value is "no".

`TRUSTEDCALIST_FILE`

It specifies a PEM encoded file's path on the file system containing the certificates of the trusted CA authorities to use. Mandatory in server mode, and mandatory in client mode if `VERIFYCERTIFICATE`="yes".

`VERIFYCERTIFICATE`

The parameter is **optional**, and can be used to tell the HTTP test port whether to check the certificate of the other side. If it is defined "yes", the test port will query and check the certificate. If the certificate is not valid (i.e. the public and private keys do not match), it will exit with a corresponding error message. If it is defined "no", the test port will not check the validity of the certificate. The default value is "no".

`KEYFILE`

This parameter is **conditional**. It specifies a PEM encoded file's path on the file system containing the RSA private key. Mandatory in server mode and optional in client mode.

`CERTIFICATEFILE`

This parameter is **conditional**. It specifies a PEM encoded file's path on the file system containing the certificate chain. For detailed information see [6]. Mandatory in server mode and optional in client mode. Note that the server may require client authentication. In this case no connection can be established without a client certificate.

PASSWORD

The parameter is **optional**, and can be used to specify the password protecting the private key file. The PASSWORD has to be the password used by generation of the private key file. If the password is not defined, the SSL toolkit asks for it when the test port receives the 'LISTEN' ASP. It is recommended to define it in the config file instead.

2.4 Start Procedure

2.4.1 TTCN-3 Test Executor

Before the executable test suite can be run the TTCN-3 modules and C++ codes should be compiled and linked into an executable program. This process can be automated using the make utility. For more information about the `Makefile` see the ***Makefile*** section and [2]. Note that the c++ implementation files `HTTPmsg_PT.hh`, `HTTPmsg_PT.cc`, `Abstract_Socket.cc`, `Abstract_Socket.hh` and the TTCN-3 modules `HTTPmsg_Types.ttcn` and `HTTPmsg_PortType.ttcn` of the test port should be included in the `Makefile`.

For information on how to start the execution see [2].

2.4.2 Connecting to a server

In case of the test performs the role of a HTTP client, the '`Connect`' ASP has to be sent. Its parameters are:

- `hostname`: host name or IP address of the remote server.
- `portnumber`: port number of the remote server where it accepts connections.
- `use_ssl`: has to be '`false`' on normal TCP/IP connections, '`true`' if the server accepts HTTPS connections.

Multiple parallel connections can be opened and used. If two or more connections are used in parallel, '`use_notification_ASPs`' parameter has to be set to '`true`', see 2.3.1. In this case '`Connect_result`' ASP is returned to the test case with the '`client_id`' associated to the connection. The returned '`client_id`' has to be used in the messages targeted to send on this connection. The returned '`client_id`' with value '`-1`' means that the server did not accept the connection because an error occurred.

2.4.3 Starting a server, listening for client connections

In case of the test performs the role of a HTTP server, the '`Listen`' ASP has to be sent. Its parameters are:

- `local_hostname`: host name or IP address of the interface in the local computer. It should be set if the workstation has multiple IP interfaces, and the test has to use a specific one.
- `portnumber`: port number where the server will accept connections.

- `use_ssl`: has to be '`false`' to accept normal TCP/IP connections, '`true`' to accept HTTPS connections.

Sending the '`Listen`' ASP multiple times will cause to close the listening port and open another one.

If '`use_notification_ASPs`' parameter is set to '`true`' in the configuration, the '`Listen_result`' ASP is returned to the test case with the opened port number. The returned '`portnumber`' with value '`-1`' means that an error occurred while setting up the requested listening port.

If a client connects to the server and '`use_notification_ASPs`' parameter is set to '`true`' in the configuration, the '`Client_connected`' ASP is sent to the test case with `hostname`, `portnumber` and `client_id` fields. '`client_id`' has to be used as described above.

2.5 Sending/receiving HTTP messages

The HTTP test port is able to send and receive `HTTPMessage` structures. The `HTTPMessage` can be one of the following types:

- `HTTPRequest`
  The Request message represents a single request to perform by the HTTP server, usually to access a *resource* on the server.

- `HTTPResponse`
  The Response message is sent by the HTTP server to the client. It includes the return status code of the request and the requested resource.

- `HTTPRequest_binary_body`
  The same as the *HTTPRequest* message. It is passed to TTCN when the body of the message contains non-ascii characters.

- `HTTPResponse_binary_body`
  The same as the *HTTPResponse* message. It is passed to TTCN when the body of the message contains non-ascii characters.

In case of multiple connections, the '`client_id`' will identify the connection. When sending a HTTP message, it has to be set to the corresponding connection id. When receiving the message, the test port sets it to the corresponding connection id, and the test case will get the right value.

Apart from the *HTTPRequest* and *HTTPResponse* ASPs above, the *erronous_msg* is received by the test port and sent to the test suite:

- `HTTP_erronous_msg`
  If a message is received on the connection, which can not be decoded as a HTTP1.1 or HTTP1.0 message, the *HTTPMessage* will contain an erroneous message with a client_id, and sent to the test suite.

2.6 Stop Procedure

2.6.1 Closing connections

To close a specific client connection, the '`Close`' ASP has to be sent with the relevant '`client_id`'. If '`client_id`' is '`omit`', all client connections will be closed.

To close the server listening port, the '`Shutdown`' ASP has to be sent.

If '`use_notification_ASPs`' parameter is set to '`true`' in the configuration, the test case will receive the '`Close`' ASP if the remote end of the connection disconnects. The '`client_id`' field will identify the relevant connection.

If the remote end closes the connection, a '`Half_close`' ASP is received by the test case. '`Half_close`' means that the remote end will not send any more data, but it may receive. Some test cases may use this functionality, but in most cases a '`Close`' ASP has to be sent in reply to it, with the '`client_id`' received in the '`Half_close`' message.

2.6.2 TTCN-3 Test Executor

The TITAN executor stops the test port after the test case is finished or in case of execution error during the test case.

3 Usage as Protocol Module

The HTTP test port can be used as a protocol module, i.e. only a protocol data structure description with the appropriate encoding and decoding functions. The data structure definitions are the same as in case of http test port, they can be found in file `HTTPmsg_Types.ttcn`. The encoding and decoding functions are declared as external functions in file `HTTPmsg_Types.ttcn` but they are implemented in files `HTTPmsg_PT.cc/hh`.

The available functions are as follows:

Name	Type of formal parameters	Type of return value
`enc_HTTPMessage`	`HTTPMessage`	`octetstring`

`dec_HTTPMessage`	`in octetstring stream`
	`inout HTTPMessage msg`
	`in boolean socket_debugging`
		integer

If the test port used as protocol module, the `Makefile` is the same as in normal case i.e `HTTPmsg_PT.cc/hh` shall be used and the port definition file `HTTPmsg_PortType.ttcn` also shall be used but (generally) there will not be defined any port of this port type "`HTTPmsg_PT`".

4 Usage with IPL4 test port

To use IPL4 test port for HTTP traffic the HTTP test port provides a message length calculator function. That function can be used to determine the message boundary by the IPL4 test port.

Name	Type of formal parameters	Type of return value
`f_HTTPMessage_len`	`in octetstring stream`	integer

5 Migrating test suite using R1x

With the release of the port version R2A it has been decided to add four new received ASPs and a new message field to the test port types. This modification has been made to fulfil the requirement of a HTTP server handling multiple parallel client connections. The change causes test suites written for R1x port to fail the compilation. However, the transition to the R2 port is straightforward.

Steps to compile older test suites with HTTP port R2:

- Decide whether SSL will be used or not in the test. If yes, please see 8.2 about editing the `Makefile` to allow SSL in the test. If SSL is not used in the test, then SSL specific parts can be removed from the `Makefile`.
  Note: `OPENSSL_DIR` and `-lssl` are still needed to compile the executable.

- Remove `buffer.cc` and `buffer.hh` from the (user) sources, and add `Abstract_Socket.cc` and `Abstract_Socket.hh`.

- Do not set the '`use_notification_ASPs`' to `"yes"` unless you do not modify the test suite to handle the incoming Close, Connect_result, Client_connected and Listen_result ASPs. It is recommended to consider the usage of these ASPs since they allow the test suite to implement more complex TCP event handling. For example, a client test case can wait for a server to be started up by checking if the `client_id` is `-1` in the returned Connect_result ASP.

- Add the `client_id := omit` assignment to every HTTPRequest , HTTPResponse, HTTPRequest_binary_body, HTTPResponse_binary_body, erronous_msg, Half_close and Close variables and templates.
  Example:

```
var HTTPRequest r := { method := "GET", uri := "/",
    version_major := 1, version_minor := 1, header := hd, body := ""}
```

has to be modified to:

```
var HTTPRequest r := { client_id := omit, method := "GET", uri := "/",
    version_major := 1, version_minor := 1, header := hd, body := ""}
```

The new ASPs are only received by the test if the '`use_notification_ASPs := "yes"`' is specified in the runtime configuration file.

6 Error messages

The error messages have the following general form:

`Dynamic test case error: <error text>`

The list of the possible error messages is shown below. Note that this list contains the error messages produced by the test port. The error messages coming from the TITAN are not shown:

Parameter value <value> not recognized for parameter <name>

The specified <value> in the runtime configuration file is not recognized for the parameter <name>. See 2.3.1.

<port name>: HTTP test port is not compiled to support SSL connections. Please check the User's Guide for instructions on compiling the HTTP test port with SSL support.

`-DAS_USE_SSL` and OpenSSL related compiling instructions are missing from the Makefile. See 8.2.

Cannot connect to server

The Connect operation failed; look for the reason above this message in the log.

Cannot listen at port

The Listen operation failed; look for the reason above this message in the log.

Cannot accept connection at port

The server failed to accept an incoming connection; look for the reason above this message in the log.

Cannot open socket

There was an error while allocating a socket for a connection; look for the reason above this message in the log.

Setsockopt failed

There was an error while allocating a socket for a connection; look for the reason above this message in the log.

Cannot bind to port

There was an error while allocating the requested port number for a connection; look for the reason above this message in the log.

getsockname() system call failed on the server socket

There was an error while allocating the requested port number for a connection; look for the reason above this message in the log.

Client Id not specified although not only 1 client exists

Since multiple connections are alive, you have to specify a client id when sending a message to distinguish between the connections where the message has to be sent.

There is no connection alive, use the 'ASP_TCP_Connect' before sending anything.

Connect has to be sent before sending a message, or the server has to accept a connection first.

Send system call failed: There is no client nr <client_id> connected to the TCP server

A send operation is performed to a non-existing client.

Send system call failed: <amount> bytes were sent instead of <amount> <reason>

The send operation failed because of the <reason>.

The host name <name> is not valid in the configuration file.

The given host name in the Connect / Listen ASP cannot be resolved by the system.

Number of clients<>0 but cannot get first client, programming error

Never should show up. Please send a bug report including log files produced with all debugging possibilities turned on.

Index <amount> exceeds length of peer list.

Never should show up. Please send a bug report including log files produced with all debugging possibilities turned on.

Abstract_Socket::get_peer: Client <client_id> does not exist

Never should show up. Please send a bug report including log files produced with all debugging possibilities turned on.

Invalid Client Id is given: <client_id>.

Please send a bug report including log files produced with all debugging possibilities turned on.

Peer <client_id> does not exist.

Never should show up. Please send a bug report including log files produced with all debugging possibilities turned on.

6.1 Additional error messages in case SSL connections are used

No SSL CTX found, SSL not initialized

Never should show up.

Creation of SSL object failed

Never should show up.

Binding of SSL object to socket failed

The SSL object could not be bound to the TCP socket

SSL error occurred

A general SSL error occurred. Check the test port logs to see previous error messages showing the real problem.

<name> is not defined in the configuration file

The test port parameter with <name> is mandatory, but is not defined in the configuration file.

No SSL data available for client <client_id>

Please send a bug report including log files produced with all debugging possibilities turned on.

Could not read from /dev/urandom

The read operation on the installed random device is failed.

Could not read from /dev/random

The read operation on the installed random device is failed.

Could not seed the Pseudo Random Number Generator with enough data.

As no random devices found, a workaround is used to seed the SSL PRNG. Consider upgrading your system with the latest available patches. HelpDesk should correct this within a day.

The seeding failed.

Please send a bug report including log files produced with all debugging possibilities turned on.

SSL method creation failed.

The creation of the SSL method object failed.

SSL context creation failed.

The creation of the SSL context object failed.

Can't read certificate file

The specified certificate file could not be read.

Can't read key file

The specified private key file could not be read.

Can't read trustedCAlist file

The specified certificate of the trusted CAs file could not be read.

Cipher list restriction failed for <name>

The specified cipher restriction list could not be set.

Unknown SSL error code: <error code>

Please send a bug report including log files produced with all debugging possibilities turned on.

7 Warning messages

The following list shows the possible warning messages produced by the test port:

HTTPmsg__PT::set_parameter(): Unsupported Test Port parameter: <name>

The specified parameter is not recognized by the test port. Check 2.3.1 for parameter names. The parameter names have to be given case sensitive.

<port name>: to switch on HTTP test port debugging, set the '*.<port name>.http_debugging := "yes" in the port's parameters.

HTTP test port produces detailed logs if you specify the http_debugging := “yes” in the configuration file.

Error when reading the received TCP PDU.

There was an error while reading incoming data from the connection. The connection gets disconnected immediately, the test is informed about the disconnect by a Close ASP with the relevant client_id.

Cannot open socket when trying to open the listen port: <reason>

The Listen operation failed because of <reason>.

Setsockopt failed when trying to open the listen port: <reason>

There was an error while allocating a socket because of <reason>. The test is informed about the failure by receiving a Listen_result ASP with portnumber = -1.

Cannot bind to port when trying to open the listen port: <reason>

There was an error while binding to the requested port because of <reason>. The test is informed about the failure by receiving a Listen_result ASP with portnumber = -1.

Cannot listen at port when trying to open the listen port: <reason>

There was an error while trying to listen for incoming connections because of <reason>. The test is informed about the failure by receiving a Listen_result ASP with portnumber = -1.

getsockname() system call failed on the server socket when trying to open the listen port: <reason>

There was an error while trying to listen on the specified port because of <reason>. The test is informed about the failure by receiving a Listen_result ASP with portnumber = -1.

Cannot open socket when trying to open client connection: <reason>

There was an error while allocating a socket for a connection because of <reason>. The test is informed about the failure by receiving a Connect_result ASP with client_id = -1.

Setsockopt failed when trying to open client connection: <reason>

There was an error while allocating a socket for a connection because of <reason>. The test is informed about the failure by receiving a Connect_result ASP with client_id = -1.

Cannot bind to port when trying to open client connection: <reason>

There was an error while binding to the requested port to the socket because of <reason>. The test is informed about the failure by receiving a Connect_result ASP with client_id = -1.

connect() returned error code EADDRINUSE. Perhaps this is a kernel bug. Trying to connect again.

If the connect system call fails because of the 'address is already in use' error, the test port automatically does 16 retry. Meanwhile this warning is logged.

Cannot connect to server when trying to open client connection: <reason>

The Connect operation failed; look for the reason above this message in the log. A Connect_result with client_id = -1 will be returned to the test.

Abstract_Socket::remove_client: <client_id> is the server listening port, can not be removed!

The specified client_id in the Close ASP belongs to the server listening port. Wrong client_id is specified.

Client <client_id> has not been removed, programming error

Please send a bug report including log files produced with all debugging possibilities turned on.

Warning: race condition while setting current client object pointer

There are multiple instances of the port running trying to access a common resource concurrently. This may cause problem.

Connection from client <client_id> is refused

The connection from a client is refused in the server.

Connection to server is refused

The connection from the client is refused by the server.

Server did not send a session ID

The connection from the client is refused by the server.

Verification failed

The verification of the other side is failed. The connection will be shut down.

SSL object not found for client <client_id>

Please send a bug report including log files produced with all debugging possibilities turned on.

SSL_Socket::receive_message_on_fd: SSL connection was interrupted by the other side

The TLS/SSL connection has been closed. If the protocol version is SSL 3.0 or TLS 1.0, this warning appears only if a closure alert has occurred in the protocol, i.e. if the connection has been closed cleanly. Note that in this case it does not necessarily indicate that the underlying transport has been closed.

SSL_Socket::send_message_on_fd: SSL connection was interrupted by the other side

See above.

Other side does not have certificate.

The other side of the SSL connection does not have a certificate.

Solaris patches to provide random generation devices are not installed. See http://www.openssl.org/support/faq.html "Why do I get a "PRNG not seeded" error message?" A workaround will be used.

Solaris patches to provide random generation devices are not installed. A workaround will be used to seed the PRNG.

Private key does not match the certificate public key

The private key specified for the test port does not match with the public key.

8 Examples

8.1 Configuration file

An example RTE configuration file is included in the ‘demo’ directory of the test port release.

8.2 Makefile

In this section the most important parameters are listed in the `Makefile`. The following gives some detail about them:

OPENSSL_DIR =
Specifies the OpenSSL installation directory. It has to contain the lib/libssl.a file and the include directory. It is not needed if OpenSSL is installed by root in the default location. It is recommended to change the already-present OPENSSL_DIR entry, which is included by the Makefile generation process.

CPPFLAGS = -D$(PLATFORM) -I$(TTCN3_DIR)/include -DAS_USE_SSL -I$(OPENSSL_DIR)/include

The `-DAS_USE_SSL` switch activates the SSL-specific code in the test port. If the switch is missing, SSL functionality will not be available, and the test port will generate dynamic test case error when connecting or listening with parameter including 'use_ssl=true' setting.
This `-I$(OPENSSL_DIR)/include` switch tells the C++ compiler where to look for the OpenSSL header files. It is not needed if OpenSSL is installed by root in the default location.

TTCN3_MODULES =
The list of TTCN-3 modules needed.

USER_SOURCES =
The list of other external C++ source files.

$(TARGET): $(OBJECTS)

**$(CXX) $(LDFLAGS) -o $@ $(OBJECTS) -L$(TTCN3_DIR)/lib -l$(TTCN3_LIB) **

-L$(OPENSSL_DIR)/lib –lssl -lcrypto $($(PLATFORM)_LIBS)
The `-L$(OPENSSL_DIR)/lib` and `-lssl` parameter tells the linker to use the `libssl.a` compiled in the `$(OPENSSL_DIR)/lib` directory.