SunRPCasp_CNL113493 Test Port for TTCN-3 Toolset with TITAN, User Guide

Csaba Fehér

Version 198 17-CNL 113 493, Rev. C, 2009-04-06

Table of Contents

About This Document

How to Read This Document

This is the User’s Guide for the SUNRPCasp_CNL113439 (called SunRPC from now on) test port. The SunRPC test port is developed for the TTCN-3 Toolset with TITAN according to the Functional Specification [3].

Prerequisite Knowledge

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

System Requirements

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

- Platform: any platform supported by TITAN RTE, optional OpenSSL.
- TITAN TTCN-3 Test Executor version R8A (1.8.pl0) or higher installed. For installation guide see [2].

NOTE This version of the protocol module 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 [5].
- The Abstract_Socket CNL 113 384, rev. R6A or later product has to be installed

Fundamental Concepts

The test port establishes connection between the TTCN-3 test executor and the Sun RPC server or client implementation through a TCP/IP socket connection. The test port transmits and receives Sun RPC messages; see [3] and [4].

The Test Port

Overview

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

Installation

Since the SunRPC 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 SunRPC 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 Sun RPC 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 [5] for more information. OpenSSL libraries should be added to the *Makefile* generated by the TITAN executor see example in Makefile.

Configuration

The executable test program behavior 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].

SunRPC 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 SunRPC test port. Parameters marked with bold fonts apply to **SSL** using SunRPC test ports **only**. Parameter values are **case-sensitive**!

- `socket_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"`.

- `ssl_use_ssl`

  The parameter is optional, and can be used to specify whether to use SSL on the top of the TCP connection or not.

  The default value is `"no"`.

- `ssl_use_session_resumption`

  The parameter is optional, and can be used to specify whether to use/support SSL session resumptions or not.

  The default value is `"yes"`.

- `ssl_private_key_file`

  The parameter is **conditional**, and have to be passed to SSL enabled SunRPC server test ports. The KEYFILE specifies a 'pem' encoded file’s path on the file system containing the server’s RSA private key.

- `ssl_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** for server, and **mandatory** for client **if** `ssl_verify_certificate="yes"`.

- `ssl_certificate_chain_file`

  It specifies a PEM encoded file’s path on the file system containing the certificate chain. **Mandatory** for server and **optional** for client. Note that the server may require client authentication. In this case no connection can be established without a client certificate. For detailed information see [5].

- `ssl_private_key_password`

  The parameter is **optional**, and may be passed to SSL enabled SunRPC server test ports. The PASSWORD has to be the password used by generation of the server’s private key file. If the password is not defined, the SSL toolkit asks for it when the test port receives the `ASP_SunRPC_Listen` ASP. It is recommended to define it in the config file instead.

- `ssl_allowed_ciphers_list`

  The parameter is **optional**, and can be used to specify the allowed cipher list. The value is passed directly to the SSL toolkit.

- `ssl_verify_certificate`

  The parameter is **optional**, and can be used to tell the client SunRPC test port to check the server authentication. If it is defined `"yes"`, the test port will query and check the server’s certificate. If the certification is not valid, it will exit with a corresponding error message.

  Its default value is `"no"`.

Start Procedure

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 section Makefile and [2].

NOTE The c++ implementation files *SunRPCasp_PT.hh*, *SunRPCasp_PT.cc*, *Abstract_Socket.hh.cc*, *Abstract_Socket.hh.hh* (from product `Abstract_Socket' CNL 113 384) and the TTCN-3 modules SunRPCasp_Types.ttcn, and *SunRPCasp_PortType.ttcn* should be included in the *Makefile*.

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

Connecting to a Server

In case of the test performs the role of a SunRPC client, the `ASP_SunRPC_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.
- `local_hostname` - select local interface for the local end of connection. It should be set if the workstation has multiple IP interfaces, and the test has to use a specific one.
- `local_portnumber` - select local port number for the local end of connection

Multiple parallel connections can be opened and used. `ASP_SunRPC_Connect_result` ASP is returned to the test case with the `client_id` associated to the opened 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.

Starting a Server, Listening for Client Connections

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

- `portnumber` - port number where the server will accept connections.
- `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.

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

The `ASP_SunRPC_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, the `ASP_SunRPC_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.

Sending/Receiving SunRPC Messages

The SunRPC test port is able to send and receive `SunRPC_message` and `SunRPC_message_multiple_client` structures. The `SunRPC_message_multiple_client` is a record of a `client_id` and a `SunRPC_message`, enabling the test suite to handle multiple clients through the same test port instance. The structure of the `SunRPC_message` message is described in [4].

The test port sends and receives the messages over the TCP/IP protocol, and does the encoding and decoding of the Record Marking Standard (see [4]) automatically.

Stop Procedure

ASP_SunRPC_Close

The `ASP_SunRPC_Close` shuts down the client connection between the test port and the IUT. The `client_id` parameter of the `ASP_SunRPC_Close` ASP identifies the connection to be closed. If it is set to `"omit"`, all current connections will be closed.

The test suite receives `ASP_SunRPC_Close` if the remote end closes the connection.

ASP_SunRPC_Shutdown

Instructs the test port to close the server listening port. The client connections will remain open. The server will not accept further client connections until an `ASP_SunRPC_Listen` ASP is sent again.

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.

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 SunRPC Test Port Parameters in the RTE Configuration File.

`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 `ASP_SunRPC_Connect` / `ASP_SunRPC_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.

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.

Warning Messages

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

`SunRPCasp__PT::set_parameter(): Unsupported Test Port parameter: <name>`

The specified parameter is not recognized by the test port. Check SunRPC Test Port Parameters in the RTE Configuration File for parameter names. The parameter names have to be given case sensitive.

`<port name>: to switch on SunRPC test port debugging, set the .<port name>.socket_debugging := "yes" in the port’s parameters.`

SunRPC test port produces detailed logs if you specify `thesocket_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 an `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 `ASP_SunRPC_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 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 https://www.openssl.org/docs/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.

Examples

Configuration file

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

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.

  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.

Terminology

- **Sockets** – The sockets is a method for communication between a client program and a server program in a network. A socket is defined as "the endpoint in a connection." Sockets are created and used with a set of programming requests or "function calls" sometimes called the sockets application programming interface (API). The most common sockets API is the Berkeley UNIX C language interface for sockets. Sockets can also be used for communication between processes within the same computer.
- **Single connection** – The test port controls a single connection, either initiated by the test sending a single `ASP_SunRPC_Connect` message, or the listening server accepts a single connection.
- **Multiple connections** – The test port controls multiple connections, either initiated by the test sending multiple `ASP_SunRPC_Connect` messages, or the listening server accepts multiple connections.

Abbreviations

API	Application Program Interface
ASP	Abstract Service Primitive
ES	ETSI Standard
ETSI	European Telecommunications Standards Institute
IUT	Implementation Under Test
RPC	Remote Procedure Call
RTE	Run-Time Environment
SUT	System Under Test
SSL	Secure Sockets Layer
TTCN-3	Testing and Test Control Notation version 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] Programmer’s Technical Reference for TITAN TTCN-3 Test Executor

[3] SUNRPCasp_CNL113493 Test Port for TTCN-3 Toolset with TITAN, Functional Specification

[4] RFC 1057
RPC: Remote Procedure Call - Protocol Specification Version 2
http://www.ietf.org/rfc/rfc1057.txt

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