Prepared (also subject responsible if other)		No.		
ETH/RZX Csaba Fehér +36 1 439 5641		155 17-CNL 113 384 Uen		
Approved	Checked	Date	Rev	Reference
ETH/RZXC (Elemér Lelik)		2009-04-01	E	GASK2

Abstract Socket Test Port for TTCN-3 Toolset with TITAN, Function Specification

Contents

Prepared (also subject responsible if other)		No.		
ETH/RZX Csaba Fehér +36 1 439 5641		155 17-CNL 113 384 Uen		
Approved	Checked	Date	Rev	Reference
ETH/RZXC (Elemér Lelik)		2009-04-01	E	GASK2

1 Introduction

1.1 Revision history

Date	Rev	Characteristics	Prepared
2005-01-19	PA1	First draft version	EANTWUH
2005-04-22	A	Accepted after review	EANTWUH
2005-08-23	PB1	Connection ASPs added	ETHECS
2006-10-16	B	Release	ETHECS
2006-11-22	PC1	New feature: use with non-blocking TCP socket	EGERGFT
2006-12-12	PC2	Reviewed	EGERGFT
2008-09-25	PD1	IPv6 support added	ETHJGI
2008-10-03	PD2	Updated after review	ETHJGI
2009-04-01	PE1	Updated according to the Test Port API introduced in TITAN R7E	ECSAFEH

1.2 How to Read this Document

This is the Function Specification for the Abstract Socket test port. The Abstract Socket test port is developed for the TTCN-3 Toolset. This document should be read together with Product Revision Information [2].

1.3 Scope

The purpose of this document is to specify the functionality of the Abstract Socket test port. The document is primarily addressed to the end users of the product. Basic knowledge of TTCN-3 and TITAN TTCN-3 Test Executor and the SSL protocol is valuable when reading this document (see [1] and [3]).

1.4 References

[1] ETSI ES 201 873-1 v.3.1.1 (2005-06)
The Testing and Test Control Notation version 3. Part 1: Core Language

[2] 109 21-CNL 113 384-5 Uen
Abstract Socket Test Port for TTCN-3 Toolset with TITAN, Product Revision Information

[3] 2/198 17-CRL 113 200 Uen
Programmer's Technical Reference for the TITAN TTCN-3 Test Executor

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

[5] Hickmann, Kipp, "The SSL Protocol", Netscape Communications Corp., Feb 9, 1995.

[6] A. Frier, P. Karlton, and P. Kocher, “The SSL 3.0 Protocol”, Netscape Communications Corp., Nov 18, 1996.

[7] RFC 2246 (1999), T. Dierks, C. Allen: “The TLS Protocol Version 1.0”

[8] 198 17-CNL 113 384 Uen
Abstract Socket Test Port for TTCN-3 Toolset with TITAN, User Guide

1.5 Abbreviations

API	Application Program Interface
ASP	Abstract Service Primitive
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
PEM	Privacy Enhanced Mail
RTE	Run-Time Environment
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TTCN-3	Testing and Test Control Notation version 3

1.6 Terminology

Sockets – The socket 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 socket API is the Berkeley UNIX C language interface for sockets. Sockets can also be used for communication between processes within the same computer.

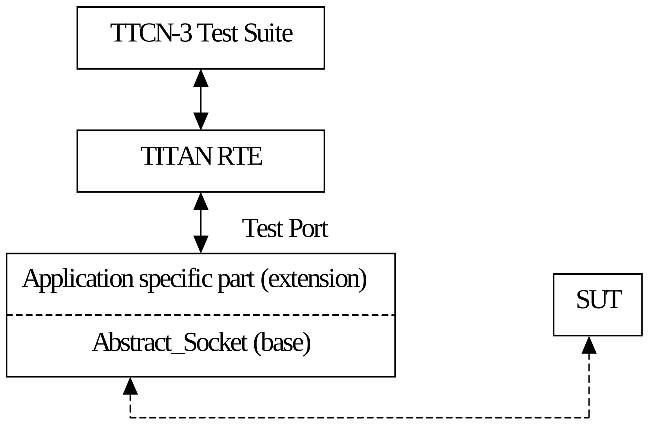
Blocking and non-blocking sockets – using a blocking socket, some socket operations (send, receive, connect, accept) will block until the operation is finished or an error occurs. Using a non-blocking socket, these operations will never block but return with an error and set *errno* to the appropriate value.

OpenSSL - The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. For more information on the OpenSSL project see [4].

Prepared (also subject responsible if other)		No.		
ETH/RZX Csaba Fehér +36 1 439 5641		155 17-CNL 113 384 Uen		
Approved	Checked	Date	Rev	Reference
ETH/RZXC (Elemér Lelik)		2009-04-01	E	GASK2

2 General

The Abstract Socket is a common component that can serve as a basis for test ports that need TCP connections with or without SSL on the top of it. The socket can be used either with blocking or non-blocking mode. Using a non-blocking socket an algorithm is also implemented in the Abstract Socket to avoid TCP deadlock. The Abstract Socket implements basic sending, receiving and socket handling routines. By extending it with additional functionality the desired test port can be built.



3 Function Specification

3.1 Implementation

3.1.1 Environment

The Abstract Socket makes use of the services provided by the UNIX socket interface. When connecting to an SSL enabled IUT, the connection is secured with the OpenSSL toolkit based on configuration data. The Abstract Socket provides the basis to build test ports that can work as a client or as a server. The Test Port can be used with or without connection ASPs (see section 3.3.1. of this document).

The Abstract Socket supports TCP over IPv4 and IPv6.

3.1.2 Module structure

The Abstract Socket common component is implemented in the following files:

- `Abstract_Socket.hh`
- `Abstract_Socket.cc`

3.2 Configuration

The configuration of the test port is done by the TITAN RTE configuration file. The description of the specific parameters can be found in the Abstract Socket User's Guide [8].

3.3 Start Procedure

3.3.1 Connection ASPs

When choosing to use connection ASPs, the Abstract Socket is able to open a server listening port (acting like a server) or client connections at the same time.

3.3.2 Server mode

When the test port is mapped by TITAN RTE, the server creates a TCP socket and starts listening on it. Depending on the transport channel specified in the runtime configuration file, it will accept either TCP or SSL connections.

3.3.3 Client mode

When the test port is mapped by TITAN RTE, the client creates a TCP socket and tries to connect to the server. If the transport channel is SSL, the client starts an SSL handshake after the TCP connection is established. If the SSL handshake is successful, the SSL connection is established and the `map()` operation is finished.

The SSL handshake may fail due to several reasons (e.g. no shared ciphers, verification failure, etc.). For possible causes study [4].

3.4 Sending/receiving messages

Only basic octetstring sending and receiving is handled by the Abstract Socket. This functionality probably must be extended in order to build a test port for the desired protocol. First the TTCN-3 mapping of the target protocol messages must be elaborated and then the message processing functions (outgoing and incoming operations, see [3]) can be developed.

3.5 Logging

The type of information that will be logged can be categorized into two groups. The first one consists of information that shows the flow of the internal execution of the test port, e.g. important events, which function that is currently executing etc. The second group deals with presenting valuable data, e.g. presenting the content of a PDU. The logging printouts will be directed to the RTE log file. The user is able to decide whether logging is to take place or not by setting appropriate configuration data, see [3].

Prepared (also subject responsible if other)		No.		
ETH/RZX Csaba Fehér +36 1 439 5641		155 17-CNL 113 384 Uen		
Approved	Checked	Date	Rev	Reference
ETH/RZXC (Elemér Lelik)		2009-04-01	E	GASK2

3.6 Error Handling

Erroneous behaviour detected during runtime is directed into the RTE log file. The following two types of messages are taken care of:

- Errors: information about errors detected is provided. If an error occurs the execution of the test case will stop immediately. The test ports will be unmapped.

- Warnings: information about warnings detected is provided. The execution continues after the warning is shown.

3.7 Closing Down

The connection can be shut down either performing the `unmap()` operation on the port or if connection ASPs are used with the appropriate ASP.

3.8 IPv6 Support

It is possible to select the address family used for server socket and client connections in the configuration file or during runtime. The following address families are supported: IPv4, IPv6 and UNSPEC.

3.9 SSL functionality

The Abstract Socket can use SSL or TCP as the transport channel. The same version of OpenSSL library must be used as in TITAN.

The protocols SSLv2, SSLv3 and TLSv1 are supported.

3.9.1 Compilation

The usage of SSL and even the compilation of the SSL related code parts are optional. This is because SSL related code parts cannot be compiled without the OpenSSL installed.

The compilation of SSL related code parts can be disabled by not defining the *AS_USE_SSL* macro in the Makefile during the compilation. If the macro is defined in the Makefile, the SSL code parts are compiled to the executable test code. The usage of the SSL then can be enabled/disabled in the runtime configuration file, see [8]. Naturally, the test port parameter will be ignored if the *AS_USE_SSL* macro is not defined during compilation.

3.9.2 Authentication

The Abstract Socket provides both server side and client side authentication. When authenticating the other side, a certificate is requested and the own trusted certificate authorities' list is sent. The received certificate is verified whether it is a valid certificate or not (the public and private keys are matching). No further authentication is performed (e.g. whether hostname is present in the certificate). The verification can be enabled/disabled in the runtime configuration file, see [8].

In server mode the test port will always send its certificate and trusted certificate authorities' list to its clients. If verification is enabled in the runtime configuration file, the server will request for a client's certificate. In this case, if the client does not send a valid certificate or does not send a certificate at all, the connection will be refused. If the verification is disabled, the connection will never be refused due to verification failure.

In client mode the test port will send its certificate to the server on the server's request. If verification is enabled in the runtime configuration file, the client will send its own trusted certificate authorities' list to the server and will verify the server's certificate as well. If the server's certificate is not valid, the SSL connection will not be established. If verification is disabled, the connection will never be refused due to verification failure.

The own certificate(s), the own private key file, the optional password protecting the own private key file and the trusted certificate authorities' list file can be specified in the runtime configuration file, see [8].

The test port will check the consistency between its own private key and the public key (based on the own certificate) automatically. If the check fails, a warning is issued and execution continues.

3.9.3 Other features

Both client and server support SSLv2, SSLv3 and TLSv1, however no restriction is possible to use only a subset of these. The used protocol will be selected during the SSL handshake automatically.

The usage of SSL session resumption can be enabled/disabled in the runtime configuration file, see [8].

The allowed ciphering suites can be restricted in the runtime configuration file, see [8].

The SSL re-handshaking requests are accepted and processed, however re-handshaking cannot be initiated from the test port.

3.9.4 Limitations

- No restriction is possible on the used protocols (e.g. use only SSLv2), it is determined during SSL handshake between the peers.
- SSL re-handshaking cannot be initiated from the test port.
- The own certificate file(s), the own private key file and the trusted certificate authorities' list file must be in PEM format. Other formats are not supported.