|  |  |
| --- | --- |
|  |  |

EPTF Core Library Transport, User Guide

Contents

[1 Introduction 2](#_Toc500158302)

[1.1 Revision history 2](#_Toc500158303)

[1.2 About this document 2](#_Toc500158304)

[1.2.1 How to read this document 2](#_Toc500158305)

[1.2.2 Abbreviations 2](#_Toc500158306)

[1.2.3 Terminology 3](#_Toc500158307)

[1.3 System requirements 3](#_Toc500158308)

[2 Transport 3](#_Toc500158309)

[2.1 Overview 3](#_Toc500158310)

[2.2 Description of files in this feature 4](#_Toc500158311)

[2.3 Description of required files from other features 5](#_Toc500158312)

[2.3.1 IPL4 transport: 5](#_Toc500158313)

[2.3.2 IPL2 transport: 7](#_Toc500158314)

[2.3.3 Common transport 8](#_Toc500158315)

[2.4 Installation 11](#_Toc500158316)

[2.5 Configuration 11](#_Toc500158317)

[2.6 Usage 11](#_Toc500158318)

[2.6.1 Choose testport 11](#_Toc500158319)

[2.6.2 Calculate the length of a message 12](#_Toc500158320)

[2.6.3 Processing the messages 13](#_Toc500158321)

[2.6.4 Comparison of the testport-specific transport layers 13](#_Toc500158322)

[2.6.5 IPL2 Routing 17](#_Toc500158323)

[2.6.6 Getting the next-hop MAC for an IP address in IPL2 17](#_Toc500158324)

[2.6.7 IPsec handling 19](#_Toc500158325)

[3 DTE handling 20](#_Toc500158326)

[4 Error messages 20](#_Toc500158327)

[5 Warning messages 21](#_Toc500158328)

[6 Examples 22](#_Toc500158329)

[6.1 Configuration file 22](#_Toc500158330)

[6.2 Demo module 22](#_Toc500158331)

[7 References 22](#_Toc500158332)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Rev** | **Characteristics** | **Prepared** |
| 2008-01-28 | PA1 | First draft version | ENORPIN |
| 2009-02-27 | PB1 | Updated with sctp support | ENORPIN |
| 2010-05-25 | PC1 | Updated with IPL2 Routing/ARP support | ethjgi |
| 2010-05-31 | PC1 | Updated filelist | EJNOSVN |
| 2010-06-08 | PC1 | Description of IPL4, IPL2 and common transport | ELSZSKU |
| 2010-06-24 | PD1 | Warning messages update | EJNOSVN |
| 2011-01-25 | PF1 | Implementation of HN51809 | EANTWUH |
| 2014-10-08 | PG1 | Added section DTE handling | EDORNAG |
| 2014-10-09 | PG2 | Documentation numbers updated | ESZILSZ |
| 2014-11-21 | PG3 | Added comment on DTE handling | ESZILSZ |
| 2014-12-02 | G | Updated for release | ESZILSZ |
| 2017-01-26 | PH1 | Added IPsec support | EIMRENA |
| 2017-01-26 | PH2 | Editorial fixes | EIMRENA |
| 2017-06-09 | H | Updated for release | EIMRENA |
| 2017-11-09 | PJ1 | Changes in IPsec handling | ZNAGIMR |
| 2017-12-01 | J | Updated for release | ZLENZOL |

## About this document

### How to read this document

This is the User Guide for EPTF Transport of the Ericsson Performance Test Framework (EPTF), Core Library (CLL). EPTF Core Library is developed for the TTCN-3 ‎[1] Toolset with TITAN [2]. This document should be read together with the Function Description of the EPTF Transport feature ‎[6]. For more information on the EPTF Core Library please consult the Product Revision Information ‎[3], the User Guide ‎[4] and the Function Specification ‎[5] of the TitanSim.

### Abbreviations

CLL Core Library

EPTF Ericsson Load Test Framework, formerly TITAN Load Test Framework

TitanSim Ericsson Load Test Framework, formerly TITAN Load Test Framework

TTCN-3 Testing and Test Control Notation version 3 ‎[1]

LGen See LoadGen

LoadGen Load generator component type. See also ‎[9]

AppLib Application Library. See also ‎[9]

### Terminology

*Core Library (CLL)* is that part of the TitanSim software that is totally project independent. (i.e., which is not protocol-, or application-dependent). The Core Library is to be supplied and supported by the TCC organization. Any Core Library development is to be funded centrally by Ericsson.

*Stack* A specific transport type, including TTCN-3 code, testport code, and host OS code.

## System requirements

In order to use the EPTF Transport feature the system requirements listed in EPTF Core Library User Guide ‎[5] should be fulfilled.

# Transport

## Overview

The EPTF Transport provides many features to help the development of a new transport component. The features are implemented in components. The features are the following:

* The Routing feature

With the use of this feature the user can register two call-back functions, which can be used to route the incoming and outgoing messages between the LoadGen components.

* The MessageBufferManager feature

With this feature the user can store octetstring messages and charstring information. It can be used to store messages until the connection is not alive.

* The Transport feature

This feature gives a common layer to send and receive messages in a testport-independent way. It contains also testport-specific transport features. All types implement the same functionality. The testport-independent feature can use any of the implemented testport-specific transports, such as IPL4 transport, IPL2 transport, and so on. The functions of this layer call the functions of the appropriate testport-specific layer.

* The IPL4 transport feature

This testport-specific transport provides functions to send and receive messages via the IPL4 testport. We can create and delete dynamically interfaces and we can set up in runtime receive altsteps and call-back functions.

* The IPL2 Transport feature

It supports the communication through the LANL2 testport.

* IPsecHandler feature: With this feature the, user can create secure connections over IP (IPsec) using the IPL4 transport. IPL2 is not supported.

## Description of files in this feature

The EPTF Core Library Transport API includes the following files:

* EPTF\_CLL\_TransportRouting\_Definitions.ttcn: This TTCN-3 module contains the type definitions of the EPTF routing feature.
* EPTF\_CLL\_TransportRouting\_Functions.ttcn: This TTCN-3 module contains the implementation of the EPTF routing feature.
* EPTF\_CLL\_TransportMessageBufferManager\_Definitions.ttcn: This TTCN-3 module contains the type definitions of the EPTF MessageBufferManager feature
* EPTF\_CLL\_TransportMessageBufferManager\_Functions.ttcn: This TTCN‑3 module contains the implementation of the EPTF MessageBufferManager feature.
* EPTF\_CLL\_TransportIPL2\_Definitions.ttcn: This TTCN-3 module contains the type definitions of the EPTF IPL2 Communication Transport feature
* EPTF\_CLL\_TransportIPL2\_Functions.ttcn: This TTCN-3 module contains the implementation of the EPTF IPL2 Communication Transport feature.
* EPTF\_CLL\_TransportIPL2\_ExternalFunctions.cc: This C++ module contains the external C++ function implementations of the EPTF IPL2 Communication Transport feature.
* EPTF\_CLL\_Transport\_Definitions.ttcn: This TTCN-3 module contains the type definitions of the EPTF\_CLL\_Transport feature
* EPTF\_CLL\_Transport\_Functions.ttcn: This TTCN-3 module contains the implementation of the EPTF\_CLL\_Transport feature.
* EPTF\_CLL\_Transport\_CommonDefinitions.ttcn: This TTCN-3 module contains the common type definitions of the EPTF\_CLL\_Transport feature. These data types are used by all transport types.
* EPTF\_CLL\_TransportIPL4\_Definitions.ttcn: This TTCN-3 module contains the type definitions of the EPTF IPL4 Communication Transport feature
* EPTF\_CLL\_TransportIPL4\_Functions.ttcn: This TTCN-3 module contains the implementation of the EPTF IPL4 Communication Transport feature.
* EPTF\_CLL\_Transport\_IPsecHandler\_Definitions.ttcn: This TTCN-3 module contains the type definitions of the IPsec support over the EPTF IPL4 Communication Transport feature
* EPTF\_CLL\_Transport\_IPsecHandler\_Functions.ttcn: This TTCN-3 module contains the implementation of the IPsec support over the EPTF IPL4 Communication Transport feature

There are files which remained for backward compatibility purposes only.

* EPTF\_Transport\_CommPort\_IPL4\_Definitions.ttcn: This TTCN-3 module contains the obsolete type definitions of the EPTF IPL4 Communication port feature
* EPTF\_Transport\_CommPort\_IPL4\_Functions.ttcn: This TTCN-3 module contains the implementation of the obsolete functions of the EPTF IPL4 Communication port feature.

## Description of required files from other features

The EPTF Transport feature is part of the EPTF Core Library (CLL). It relies on several features of the CLL, the TCC Useful Functions, and other TCC products. The user has to obtain different products or files to use the Transport feature, depending on the used transport type.

### IPL4 transport:

|  |  |  |
| --- | --- | --- |
| CLL | Base | EPTF\_CLL\_Base\_Functions.ttcn |
| EPTF\_CLL\_Base\_ExternalFunctions.cc |
| EPTF\_CLL\_Base.grp |
| EPTF\_CLL\_Base\_Definitions.ttcn |
| Common | EPTF\_CLL\_Common\_Functions.ttcn |
| EPTF\_CLL\_Common.grp |
| EPTF\_CLL\_Common\_RndDefinitions.ttcn |
| EPTF\_CLL\_Common\_RndFunctions.ttcn |
| EPTF\_CLL\_Common\_IndexArrayDefinitions.ttcn |
| EPTF\_CLL\_Common\_ExternalFunctions.cc |
| EPTF\_CLL\_Common\_Definitions.ttcn |
| EPTF\_CLL\_Common\_IndexArrayFunctions.ttcn |
| EPTF\_CLL\_Common\_PrivateDebugDefinitions.ttcnpp |
| FreeBusyQueue | EPTF\_CLL\_FBQ\_Definitions.ttcn |
| EPTF\_CLL\_FreeBusyQueue.grp |
| EPTF\_CLL\_FBQ\_PrivateFunctions.ttcn |
| EPTF\_CLL\_FBQ\_Functions.ttcn |
| HashMap | EPTF\_CLL\_HashMap.grp |
| EPTF\_CLL\_HashMapOct2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapInt2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapStr2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapInt2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapOct2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapStr2Int\_ExternalFunctions.cc |
| Logging | EPTF\_CLL\_Logging\_Functions.ttcn |
| EPTF\_CLL\_Logging.grp |
| EPTF\_CLL\_Logging\_Definitions.ttcn |
| Transport | EPTF\_CLL\_TransportMessageBufferManager\_Functions.ttcn |
| EPTF\_CLL\_TransportRouting\_Functions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL4\_Functions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL4\_Definitions.ttcn |
| EPTF\_CLL\_TransportRouting\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_CommonDefinitions.ttcn |
| EPTF\_CLL\_TransportMessageBufferManager\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Client\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Client\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_CommonDefinitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_Functions.ttcn |
| Variable | EPTF\_CLL\_Variable.grp |
| EPTF\_CLL\_Variable\_Definitions.ttcn |
| EPTF\_CLL\_Variable\_Functions.ttcn |
| EPTF\_CLL\_Variable\_ExternalFunctions.cc |
| TCCUsefulFunctions |  | TCCInterface.cc |
| TCCFileIO\_Functions.ttcn |
| TCCFileIO.cc |
| TCCDateTime\_Functions.ttcn |
| TCCInterface\_Functions.ttcn |
| TCCInterface\_ip.h |
| TCCDateTime.cc |
| TCCIPsec\_Definitions.ttcn |
| TCCIPsec\_Functions.ttcn |
| TCCIPsec.cc |
| TCCIPsec\_XFRM.hh |
| TCCIPsec\_XFRM\_SA.cc |
| TCCIPsec\_XFRM.cc |
| TCCIPsec\_XFRM\_Definitions.ttcn |
| TCCIPsec\_XFRM\_SP.cc |
| TestPorts | IPL4asp | IPL4asp\_PT.hh |
| IPL4asp\_PortType.ttcn |
| IPL4asp\_discovery.cc |
| IPL4asp\_PT.cc |
| IPL4asp\_Types.ttcn |
| IPL4asp\_protocol\_L234.hh |
| IPL4asp\_Functions.ttcn |
| Socket\_API | Socket\_API\_Definitions.ttcn |

### IPL2 transport:

|  |  |  |
| --- | --- | --- |
| CLL | Base | EPTF\_CLL\_Base\_Functions.ttcn |
| EPTF\_CLL\_Base\_ExternalFunctions.cc |
| EPTF\_CLL\_Base\_Definitions.ttcn |
| Common | EPTF\_CLL\_Common\_Functions.ttcn |
| EPTF\_CLL\_Common\_RndDefinitions.ttcn |
| EPTF\_CLL\_Common\_RndFunctions.ttcn |
| EPTF\_CLL\_Common\_IndexArrayDefinitions.ttcn |
| EPTF\_CLL\_Common\_ExternalFunctions.cc |
| EPTF\_CLL\_Common\_Definitions.ttcn |
| EPTF\_CLL\_Common\_IndexArrayFunctions.ttcn |
| EPTF\_CLL\_Common\_PrivateDebugDefinitions.ttcnpp |
| FreeBusyQueue | EPTF\_CLL\_FBQ\_Definitions.ttcn |
| EPTF\_CLL\_FBQ\_PrivateFunctions.ttcn |
| EPTF\_CLL\_FBQ\_Functions.ttcn |
| HashMap | EPTF\_CLL\_HashMapOct2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapInt2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapStr2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapInt2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapOct2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapStr2Int\_ExternalFunctions.cc |
| Logging | EPTF\_CLL\_Logging\_Functions.ttcn |
| EPTF\_CLL\_Logging\_Definitions.ttcn |
| RedBlackTree | EPTF\_CLL\_RBtree\_Definitions.ttcn |
| EPTF\_CLL\_RBtreeFloat\_Functions.ttcn |
| EPTF\_CLL\_RBtree\_Functions.ttcn |
| EPTF\_CLL\_RBtree\_PrivateFunctions.ttcn |
| EPTF\_CLL\_RBtreeInteger\_Functions.ttcn |
| EPTF\_CLL\_RBtreeFloat\_PrivateFunctions.ttcn |
| EPTF\_CLL\_RBtreeInteger\_PrivateFunctions.ttcn |
| Scheduler | EPTF\_CLL\_RBTScheduler\_Functions.ttcnpp |
| EPTF\_CLL\_Scheduler\_Functions.ttcnin |
| EPTF\_CLL\_RBTScheduler\_Definitions.ttcnpp |
| EPTF\_CLL\_Scheduler\_Definitions.ttcnin |
| EPTF\_CLL\_Scheduler\_Definitions.ttcn |
| Semaphore | EPTF\_CLL\_Semaphore\_Definitions.ttcn |
| EPTF\_CLL\_Semaphore\_Functions.ttcn |
| Transport | EPTF\_CLL\_TransportMessageBufferManager\_Functions.ttcn |
| EPTF\_CLL\_TransportRouting\_Functions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL2\_Functions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL2\_Definitions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL2\_ExternalFunctions.cc |
| EPTF\_CLL\_TransportRouting\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_CommonDefinitions.ttcn |
| EPTF\_CLL\_TransportMessageBufferManager\_Definitions.ttcn |
| Variable | EPTF\_CLL\_Variable\_Definitions.ttcn |
| EPTF\_CLL\_Variable\_Functions.ttcn |
| EPTF\_CLL\_Variable\_ExternalFunctions.cc |
| TCCUsefulFunctions |  | TCCInterface.cc |
| TCCFileIO\_Functions.ttcn |
| TCCMaths\_GenericTypes.ttcn |
| TCCConversion\_Functions.ttcn |
| TCCFileIO.cc |
| TCCConversion.cc |
| TCCMaths.cc |
| TCCSystem.cc |
| TCCSystem\_Functions.ttcn |
| TCCDateTime\_Functions.ttcn |
| TCCInterface\_Functions.ttcn |
| TCCInterface\_ip.h |
| TCCMaths\_Functions.ttcn |
| TCCDateTime.cc |
| TestPorts | LANL2asp | LANL2asp\_PT.hh |
| LANL2asp\_PT.cc |
| LANL2asp\_Types.ttcn |
| LANL2asp\_PortType.ttcn |
| Socket\_API | Socket\_API\_Definitions.ttcn |
| ProtocolModules | COMMON | General\_Types.ttcn |
| IP | IP\_EncDec.cc |
| IP\_Types.ttcn |
| TCP | TCP\_Types.ttcn |

### Common transport

|  |  |  |
| --- | --- | --- |
| CLL | Base | EPTF\_CLL\_Base\_Definitions.ttcn |
| EPTF\_CLL\_Base\_ExternalFunctions.cc |
| EPTF\_CLL\_Base\_Functions.ttcn |
| Common | EPTF\_CLL\_Common\_Definitions.ttcn |
| EPTF\_CLL\_Common\_ExternalFunctions.cc |
| EPTF\_CLL\_Common\_Functions.ttcn |
| EPTF\_CLL\_Common\_IndexArrayDefinitions.ttcn |
| EPTF\_CLL\_Common\_IndexArrayFunctions.ttcn |
| EPTF\_CLL\_Common\_PrivateDebugDefinitions.ttcnpp |
| EPTF\_CLL\_Common\_RndDefinitions.ttcn |
| EPTF\_CLL\_Common\_RndFunctions.ttcn |
| FreeBusyQueue | EPTF\_CLL\_FBQ\_Definitions.ttcn |
| EPTF\_CLL\_FBQ\_Functions.ttcn |
| EPTF\_CLL\_FBQ\_PrivateFunctions.ttcn |
| HashMap | EPTF\_CLL\_HashMapInt2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapInt2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapOct2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapOct2Int\_Functions.ttcn |
| EPTF\_CLL\_HashMapStr2Int\_ExternalFunctions.cc |
| EPTF\_CLL\_HashMapStr2Int\_Functions.ttcn |
| Logging | EPTF\_CLL\_Logging\_Definitions.ttcn |
| EPTF\_CLL\_Logging\_Functions.ttcn |
| RedBlackTree | EPTF\_CLL\_RBtree\_Definitions.ttcn |
| EPTF\_CLL\_RBtree\_Functions.ttcn |
| EPTF\_CLL\_RBtree\_PrivateFunctions.ttcn |
| EPTF\_CLL\_RBtreeFloat\_Functions.ttcn |
| EPTF\_CLL\_RBtreeFloat\_PrivateFunctions.ttcn |
| EPTF\_CLL\_RBtreeInteger\_Functions.ttcn |
| EPTF\_CLL\_RBtreeInteger\_PrivateFunctions.ttcn |
| Scheduler | EPTF\_CLL\_RBTScheduler\_Definitions.ttcnpp |
| EPTF\_CLL\_RBTScheduler\_Functions.ttcnpp |
| EPTF\_CLL\_Scheduler\_Definitions.ttcn |
| EPTF\_CLL\_Scheduler\_Definitions.ttcnin |
| EPTF\_CLL\_Scheduler\_Functions.ttcnin |
| Semaphore | EPTF\_CLL\_Semaphore\_Definitions.ttcn |
| EPTF\_CLL\_Semaphore\_Functions.ttcn |
| Transport | EPTF\_CLL\_Transport\_CommonDefinitions.ttcn |
| EPTF\_CLL\_Transport\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_Functions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL2\_Definitions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL2\_ExternalFunctions.cc |
| EPTF\_CLL\_TransportCommPortIPL2\_Functions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL4\_Definitions.ttcn |
| EPTF\_CLL\_TransportCommPortIPL4\_Functions.ttcn |
| EPTF\_CLL\_TransportMessageBufferManager\_Definitions.ttcn |
| EPTF\_CLL\_TransportMessageBufferManager\_Functions.ttcn |
| EPTF\_CLL\_TransportRouting\_Definitions.ttcn |
| EPTF\_CLL\_TransportRouting\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Client\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Client\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_CommonDefinitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Functions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_Definitions.ttcn |
| EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_Functions.ttcn |
| Variable | EPTF\_CLL\_Variable\_Definitions.ttcn |
| EPTF\_CLL\_Variable\_ExternalFunctions.cc |
| EPTF\_CLL\_Variable\_Functions.ttcn |
| TestPorts | IPL4asp | IPL4asp\_discovery.cc |
| IPL4asp\_Functions.ttcn |
| IPL4asp\_PortType.ttcn |
| IPL4asp\_protocol\_L234.hh |
| IPL4asp\_PT.cc |
| IPL4asp\_PT.hh |
| IPL4asp\_Types.ttcn |
| LANL2asp | LANL2asp\_PortType.ttcn |
| LANL2asp\_PT.cc |
| LANL2asp\_PT.hh |
| LANL2asp\_Types.ttcn |
| Socket\_API | Socket\_API\_Definitions.ttcn |
| TCCUsefulFunctions |  | TCCConversion.cc |
|  | TCCConversion\_Functions.ttcn |
|  | TCCDateTime.cc |
|  | TCCDateTime\_Functions.ttcn |
|  | TCCFileIO.cc |
|  | TCCFileIO\_Functions.ttcn |
|  | TCCInterface.cc |
|  | TCCInterface\_Functions.ttcn |
|  | TCCInterface\_ip.h |
|  | TCCMaths.cc |
| TCCMaths\_Functions.ttcn |
| TCCMaths\_GenericTypes.ttcn |
| TCCSystem.cc |
| TCCSystem\_Functions.ttcn |
| TCCIPsec\_Definitions.ttcn |
| TCCIPsec\_Functions.ttcn |
| TCCIPsec.cc |
| TCCIPsec\_XFRM.hh |
| TCCIPsec\_XFRM\_SA.cc |
| TCCIPsec\_XFRM.cc |
| TCCIPsec\_XFRM\_Definitions.ttcn |
| TCCIPsec\_XFRM\_SP.cc |
| ProtocolModules | COMMON | General\_Types.ttcn |
| IP | IP\_EncDec.cc |
| IP\_Types.ttcn |
| TCP | TCP\_Types.ttcn |

## Installation

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

If not otherwise noted in the respective sections, the following are needed to use EPTF Transport:

* Copy the files listed in section [2.1, 2.3] to the directory of the test suite or create symbolic links to them.
* Import the Transport demo or write your own application using Transport.
* Create Makefile or modify the existing one. For more details see the relevant section of [2].
* Edit the config file according to your needs; see following section [2.5].

## Configuration

The executable test program behavior is determined through the run-time configuration file. This is a simple text file which contains various sections. The usual suffix of configuration files is .cfg. For further information on the configuration file, see [2].

We do not need any configuration parameter to use this features.

## Usage

The user needs only extend and configure the right feature component in the user defined component.

### Choose testport

The goal of the transport layer is to provide a testport-independent API. The signatures of the functions of all transport component types are the same. Therefore if you want to change the used testport, you need to change your code minimally, or even do not need to change it.

You can choose from the testports to be used either compile-time, or runtime.

#### Using the EPTF\_Transport\_Common\_CT

Each function of the transport API has a parameter of type EPTF\_Transport\_TransportType. If you used the EPTF\_Transport\_Common\_CT component type as transport layer, this transport type determines the testport-dependent transport layer to be used. The supported testport-dependent transport types are IPL2 and IPL4 now. Implicitly IPL2 means that the EPTF\_CommPort\_IPL2\_CT is the used layer, and IPL4 means that the EPTF\_CommPort\_IPL4\_CT is.

There f\_EPTF\_Transport\_init function of the EPTF\_Transport\_Common\_CT is a bit special. If you use IPL2 or IPL4 as the pl\_transportType parameter, you can use only the same value in each transport function in the following.

However, you can use the element "ALL" of the EPTF\_Transport\_TransportType enumerated. This means that you can use all the supported testport-dependent transport types later. In all other functions you must use a testport-specific element of this enumerated.

Of course if you open a connection using one testport, e.g. IPL2, you must use the same EPTF\_Transport\_TransportType element during the whole process of the connection.

#### Using testport-specific transport layer

If you know, that you do not want to use more than one testport-specific transport layer, and you know which one you want to use, you can use the specific transport layer directly. That is you do not have to use the EPTF\_Transport\_Common\_CT, but you can use either the EPTF\_CommPort\_IPL2\_CT, or the EPTF\_CommPort\_IPL4\_CT. You do not have to change your code, only in the import part you have to import the appropriate testport-specific function modules, and your component must extend the appropriate transport component type.

### Calculate the length of a message

Users can register callback functions to calculate the length of the received messages. The registered function must be the type declared in the EPTF\_Transport\_GetMsgLen\_FT function type.

The callback functions can be registered by LoadGen types. These LoadGen types are identified by a charstring name. To open a connection, the name of the LoadGen type must be specified. When a packet arrives, the associated LoadGen type is retrieved from the connection ID, and the transport calls the appropriate message length calculating function.

The message length calculating function receives the type of the transport, the available stream in octetstring format, and an integer list, which can be specified at the registration of the callback.

### Processing the messages

Users can register callback functions which are called when a whole message arrived.

These callback functions can be registered by LoadGen types, just like the message length calculating functions.

It is very important to mention that in the message handler(s) it is forbidden to use any blocking function calls (for example, any functions which are using semaphores or blocking operations like receive and timeout statements). Otherwise it could lead to the corruption of the internal variables (see TR number HN51809)

### Comparison of the testport-specific transport layers

There are different transports which use different testports. The reason is that they have different advantages and disadvantages. The following sections contain the comparison of them.

#### Socket handling

*EPTF\_CommPort\_IPL4\_CT*

The socket handling of the EPTF\_CommPort\_IPL4\_CT transport type is based on the kernel.

*EPTF\_CommPort\_IPL2\_CT*

The socket handling of this transport type is moved into the user-space to make it possible to optimize independently from the kernel.

#### Scalability

The LINUX kernel implementations generally use one CPU for routing and packet decoding. Therefore this is not scalable even if we use more CPUs.

One of the reasons of moving the socket handling into user space was to make the transport layer scalable.

Figure 1 shows the measurement results of the scalability of the IPL2 transport type.

Figure 1 Scalability of the IPL2 common transport

#### Handling numerous sockets

Another reason of moving the socket handling into user space was that the general LINUX kernel implementations are optimized for cases when there are few IP addresses in the host machine, and from these few IP addresses we want to open socket connections to several different IP addresses. Therefore if you want to optimize for other cases, the kernel must be changed.

Figure 2 shows the performance changes when there are many IP addresses to open several sockets. Figure 3 shows the performance changes when there are many ports to open several sockets. Both cases the performance of the IPL4 transport downs steeply while the performance of the IPL2 transport remains almost unchanged.

**Msg/sec by IP addresses, 4 LGens**

0

20000

40000

60000

80000

100000

120000

140000

160000

100

1000

10000

100000

500000

**Sockets**

**msg/s**

IPL4

IPL2

Figure 2 Performance changes increasing the IP addresses

0

20000

40000

60000

80000

100000

120000

140000

160000

180000

100

1000

10000

100000

500000

**Sockets**

**msg/s**

IPL4

IPL2

Figure 3 Performance changes by increasing the amount of ports

#### Supported protocols

* EPTF\_CommPort\_IPL4\_CT
* UDP
* TCP
* SSL
* SCTP
* EPTF\_CommPort\_IPL2\_CT
* UDP

#### Required rights

*EPTF\_CommPort\_IPL4\_CT*

This transport type does not require special rights.

*EPTF\_CommPort\_IPL2\_CT*

Since the EPTF\_CommPort\_IPL2\_CT is based on the LANL2 testport, its use requires root permissions.

### IPL2 Routing

IPL2 Common Transport provides routing table support. The host routing table can be read into TTCN3 structure by the f\_EPTF\_CommPort\_IPL2\_getHostRoutingTable function. This function works only on Linux platform, and returns the routing table of the host where the PTC that calls this function is running. On Solaris machines empty routing table is returned.

The returned routing table, or custom routing table can be set into the database of IPL2 Common Transport by the function f\_EPTF\_CommPort\_IPL2\_setRouteTable. This function clears the previous entries and sets the new routing table. When the Common Transport is initialized, the host routing table is automatically loaded into the database.

Custom entries can be added to existing routing table database by the function f\_EPTF\_CommPort\_IPL2\_addRoute. The …setRouteTable and …addRoute functions normalize the routing table automatically by calling the f\_EPTF\_CommPort\_IPL2\_normalizeRoutingTable function. This means that the routing table will contain entries sorted by the networks mask starting from the smallest network.

The IP address of the next hop can be determined by the f\_EPTF\_CommPort\_IPL2\_getNextHop function. This function uses the routing table database to find out the next hop IP address. The routing table should be normalized.

### Getting the next-hop MAC for an IP address in IPL2

The MAC address for the next hop of an IP address can be determined by the f\_EPTF\_CommPort\_IPL2\_ARP\_getMac function. This MAC address is where the Ethernet packet should be sent towards the target IP address. This function maintains a MAC address database and uses ARP protocol to find out the MAC address for the next hop of a given IP address. APR requests are sent out by the LANL2 test port. In order to be able to send out the request, the LANL2 test port has to be configured. The eth\_interface\_name test port parameter has to be set to the interface on which the ARP messages will be sent out. The routing table can be used to determine the interface name towards an IP address. Additionally, the packet\_filter also has to allow the arp protocol.

Only one ARP request is sent out to determine the MAC address of an IP address. Successive calls which require the MAC for the same IP will use the cached value. This means that changes in routing table during execution will be ignored and MAC address stored for an IP will be the same during execution.

If a lot of IP addresses are configured on the same interface, it is recommended to define a sub-network and a common gateway for these addresses using the same interface. This optimizes the speed and memory usage of IPL2 Common Transport, because otherwise it should store the next-hop MAC addresses for each and every IP address, because the routing table tells that that these IP addresses are on the same network, and can be connected directly. Actually each MAC address will be the same for all of these IP addresses.

However, when the routing table is updated with a gateway through which these IP addresses can be reached will result in storing only the MAC address of the common gateway (which should be the same MAC address as for the IP addresses). It is not necessary to update the host routing table, just the routing table database of the IPL2 Common Transport. For example for the IP address range 192.168.10.1-192.168.10.255 the subnet 192.168.10.0 could be used with selecting one IP from the range as the gateway, e.g. 192.168.10.1. The subnet mask would be 255.255.255.0. The …addRoute function called with these parameters can be used to add the new route to the database.

If there is no route to the target IP address null MAC address is returned. Messages over L2 will not be sent in this case.

IP addresses created by the LAN2 Common Transport (by listen or connect functions), can be used as target addresses for determining the next-hop MAC address. This means that IPL2 Common Transport will reply ARP-requests for these IP addresses automatically.

When messages are sent over the L2 layer the next MAC address for the destination IP address is automatically determined based on the host routing table when the module parameter tsp\_EPTF\_CommPort\_IPL2\_dstMacAddr is set to its default value (null address). If this module parameter specifies a non-null MAC address, that MAC address is used as where the messages should be sent. In this case ARP discovery of IP addresses are turned off.

Source MAC addresses are determined automatically by the LANL2 test port. But for MAC discovery to work, the source MAC address has to be specified in the tsp\_EPTF\_CommPort\_IPL2\_srcMacAddr module parameter. This parameter is used in the ARP messages as the source MAC address.

**Note:** the source MAC address specified in the tsp\_EPTF\_CommPort\_IPL2\_srcMacAddr or used in the f\_EPTF\_CommPort\_IPL2\_ARP\_getMac function should be set according to the eth\_interface\_name test port parameter used by the LAN2 test port. Setting the source MAC address incorrectly will result in erroneous network operation and may affect other network connections by for example closing them down temporarily by diverting ARP replies to the wrong address.

### IPsec handling

With the Transport IPsec handler, the IPsec support is provided for the IPL4 transport. If the Common Transport is used, IPL4 should be specified for the transport type.

The IPsec is only available if the IPsecHandler is initialized in Transport. This can be done by calling its initializer function f\_EPTF\_Transport\_IPsecHandler\_init\_CT.

To create a secured TCP connection, the following steps should be followed:

1. Allocate SPI
2. Create security policy
3. Create security association
4. Create TCP connection with IPL4 Transport connect
5. Send messages with IPL4 Transport send
6. Close the connection with IPL4 Transport disconnect, or Transport reports the connection closed event
7. Remove the security association
8. Remove the security policy

Any message sent after SP/SA is created will be encrypted until SP/SA is deleted. All the SP/SA configuration data should be consistently set on both sides of the communication, otherwise the messages will not be sent or received correctly.

All the messages sent by the Transport before the SP/SA is created or after SP/SA is deleted, will not be encrypted.

To remove all SA/SP data from the kernel, the deleteSA/SP functions can be used respectively.

The IPsec data is logged into a file if IPsec logging is enabled when a SA is created.

To enable IPsec logging:

1. Start the IPsec logging server
2. Specify the server component reference to the initializer function of Transport IPsec Handler.

The logging server can be started like this:

var EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_CT vl\_loggingServer := EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_CT.create;

vl\_loggingServer.start(f\_EPTF\_CLL\_Transport\_IPsecHandler\_Logging\_Server\_behaviour("./", "IPsec.log"));

In this case the server will log the IPsec data into the IPsec.log file in the current directory.

Then this component reference must be passed to the IPsec init function:

f\_EPTF\_Transport\_IPsecHandler\_init\_CT("TransportIPsecHandler",vl\_loggingServer);

*Permissions:*

Since the IPsec handler uses the kernel with the XFRM API to create secured connections, its use requires root permissions.

# DTE handling

When the Transport executes the message handler functions registered by f\_EPTF\_CommPort\_IPL4\_setReceive, f\_EPTF\_CommPort\_IPL4\_send or EPTF\_CLL\_TransportIPL4\_Functions.f\_EPTF\_Transport\_registerMsgCallback and a dynamic test case error occurs, then the execution of the component will not stop due to the error. However, Transport will print out a warning message about the error.

By default DTE handling is disabled.

# Error messages

* The parent connection ID not found in the hashmap
* The connection ID not found in the hashmap
* Unknown SCTP event received
* LGenType not found in the hashmap
* Buffer contains invalid data
* … The specified … transport type is invalid.

All functions of the transport API have an argument which declares the transport type to be used (IPL4, IPL2, ALL, etc.) If a function implemented in the IPL4 transport got for example IPL2 as this argument, the function throws this error message. This message is thrown too if the argument inappropriate in the specified function, such as transport type is "ALL" in a send function.

* Function … is not implemented in transport type …

There can be functions which are not implemented in all transport types. When the user calls a function of common transport layer of this kind width the transport type argument where the function is not implemented, this error message is thrown.

* LGenType must be specified.

The LGenType argument of the registration function of the message length calculating callback must not be empty string.

* The incoming even type is unknown. Event: …

This error can occur only if the version of the transport feature is older, than the version of the Socket\_API. It should be updated.

* The received asp type is unknown. ASP: …

This error can occur only if the version of the transport feature is older, than the version of the Socket\_API. It should be updated.

* unsupported protocol …

The specified protocol is not supported in the given transport type.

# Warning messages

* The parent connection ID not found in the hashmap
* The connection ID not found in the hashmap
* Unknown SCTP event received
* LGenType not found in the hashmap
* Buffer contains invalid data
* no message handler is set for LGen …  
  There is no registered message processing function for the LoadGen type, which is associated to the connection where an incoming packet arrived.
* no LGen is set for connection  
  A connection should be opened only if there had been message length calculating function registered for that LoadGen type.
* UDP packet length is greater than (65535 - 8).
* f\_EPTF\_SchedulerComp\_scheduleAction failed, could not schedule retransmission of TCP segment.  
  Internal error.
* No route to host:…
* TCP MSS is greater than the receive window size, please reconfigure.
* Buffer is full, clearing buffer.

# Examples

The “demo” directory of the deliverable contains the following examples:

* Transport\_Demo.cfg
* Transport\_Demo.ttcn

## Configuration file

The used configuration file (Transport\_Demo.cfg) is for the Statistics Replay example is placed in the demo directory.

## Demo module

The demo module (Transport\_Demo.ttcn) illustrates a typical usage of the Transport feature.

# References

1. ETSI ES 201 873-1 v3.2.1 (2007-02)  
   The Testing and Test Control Notation version 3. [Part 1: Core Language](http://www.etsi.org/deliver/etsi_es/201800_201899/20187301/03.02.01_60/es_20187301v030201p.pdf)
2. 1/198 17-CRL 113 200/6 Uen  
   User Guide for the TITAN TTCN-3 Test Executor
3. 109 21-CNL 113 512-26 Uen   
   EPTF Core Library for TTCN-3 toolset with TITAN, Product Revision Information
4. 155 17-CNL 113 512 Uen   
   EPTF Core Library for TTCN-3 toolset with TITAN, Function Specification
5. 198 17-CNL 113 512 Uen  
   EPTF Core Library for TTCN-3 toolset with TITAN, User Guide
6. 25/155 16-CNL 113 512 Uen  
   EPTF Core Library Transport, Function Description
7. EPTF Core Library for TTCN-3 toolset with TITAN, [Reference Guide](http://ttcn.ericsson.se/TCC_Releases/Libraries/EPTF_Core_Library_CNL113512/doc/apidoc/html/index.html)
8. 109 21-CNL 113 472-31 Uen  
   TCC Useful Functions for TTCN-3 Toolset with TITAN,  
   Product Revision Information
9. ETH/RZ-2009:0001  
   [Applib Writers' Guide](http://anon.ericsson.se/eridoc/erl/objectId/09004cff84854ced?docno=1/00021-2/LXE108975&action=current&format=msw8)