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

EPTF CLL for TTCN-3 toolset with TITAN, User Guide

Contents

[1 Introduction 2](#_Toc245801113)

[1.1 Revision information 2](#_Toc245801114)

[1.2 How to Read this Document 2](#_Toc245801115)

[1.3 Scope 2](#_Toc245801116)

[1.4 Recommended way of reading 3](#_Toc245801117)

[1.5 Abbreviations 3](#_Toc245801118)

[1.6 Terminology 3](#_Toc245801119)

[2 TitanSim Core Load Library (TitanSim CLL) 6](#_Toc245801120)

[2.1 Introduction 6](#_Toc245801121)

[2.2 Base 7](#_Toc245801122)

[2.3 Central Scheduling 7](#_Toc245801123)

[2.4 Common 7](#_Toc245801124)

[2.5 Execution Control 7](#_Toc245801125)

[2.6 FreeBusyQueue 7](#_Toc245801126)

[2.7 HashMap 8](#_Toc245801127)

[2.8 HostAdmin 8](#_Toc245801128)

[2.9 LGenBase 8](#_Toc245801129)

[2.10 LoadRegulator 8](#_Toc245801130)

[2.11 Logging 8](#_Toc245801131)

[2.12 NameService 8](#_Toc245801132)

[2.13 PTCDeployment 9](#_Toc245801133)

[2.14 RandomNArray 9](#_Toc245801134)

[2.15 RedBlackTree 9](#_Toc245801135)

[2.16 Rendezvous 9](#_Toc245801136)

[2.17 RingBuffer 9](#_Toc245801137)

[2.18 Scheduler 9](#_Toc245801138)

[2.19 Semaphore 10](#_Toc245801139)

[2.20 StatCapture 10](#_Toc245801140)

[2.21 StatHandler 10](#_Toc245801141)

[2.22 StatMeasure 10](#_Toc245801142)

[2.23 StatReplay 10](#_Toc245801143)

[2.24 Time Profile Editor 10](#_Toc245801144)

[2.25 Transport 11](#_Toc245801145)

[2.26 UIHandler 11](#_Toc245801146)

[2.27 Variable 11](#_Toc245801147)

[3 Installation and Configuration 11](#_Toc245801148)

[4 References 12](#_Toc245801149)

# Introduction

## Revision information

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-06-20 | PA1 | First draft version | EGBOZIE |
| 2007-06-25 | PA2 | Updated after review | QTAMBUT |
| 2007-08-07 | A | Closing FOA | EGBOZIE |
| 2007-12-06 | PB1 | Updated version for Release 2 | EANDDAR |
| 2008-02-04 | PB2 | Updated version for Release 2.1 | ETHGASZ |
| 2008-02-04 | PB3 | Updated after review | ETHGASZ |
| 2008-02-04 | PB4 | Updated version for Release 2.2 | ETHGASZ |
| 2008-03-28 | PB5 | Updated version for Release 2.3 | ETHGASZ |
| 2009-03-05 | PC1 | Updated version for Release 3 | ETAMBUT |
| 2009-11-04 | PD1 | EPTF\_DEBUG switch | EANDRKS |

## How to Read this Document

This is Users Guide for the Ericsson Performance Test Framework (TitanSim), Core Load Library (CLL). TitanSim CLL is developed for the TTCN-3 Toolset with TITAN ‎[2]. This document should be read together with Product Revision Information ‎[3] and the Functional Specification ‎[4].

## Scope

This document is the User Guide for the TitanSim Core Load Library (CLL). Under the collective term “TitanSim” we mean several products:

* “Turn-key applications”, like “TitanSim for IMS I&V”
* Building blocks of such applications, that is, Application Libraries, such as SIP Application Library
* The application independent CLL.

The *technology* under the hood of TitanSim has been called historically the *Ericsson Performance Test Framework* (EPTF), hence the acronym “EPTF” in many of the identifiers of the API.

This document focuses on understanding the CLL. It contains the description of CLL features and references to their detailed User Guides. The demo applications are located in the product releases VOB along the CLL itself.

## Recommended way of reading

The readers are supposed to get familiar with the scope and goals of the project from ‎[6] and also the glossary therein. The next step is to read the Quick Study ‎[7] and its glossary. Finally they should get familiar with the list of acronyms and the glossary in Section ‎1.5 and ‎1.6 respectively.

## Abbreviations

API Application Programming Interface

EPTF Ericsson Load Test Framework, formerly TITAN Load Test Framework

FOA First Office Application

IUT Implementation Under Test

LGen Load Generator PTC

PTC Parallel Test Component

SUT System Under Test

TLTF See EPTF

## Terminology

*TitanSim Application Library* is that part of the TitanSim software that is (at-least partially) project dependent. (I.e., which is protocol-, and/or application-dependent). Any TitanSim application library is to be supplied and supported *either* by the customer project *only*; or by the customer project *together* with the TCC organization. Any TitanSim application library development is to be funded by the ordering projects.

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

*Application Library (AppLib)* is that part of the TitanSim software that is although protocol, or application-area dependent, but can be *reused* across many TitanSim applications. AppLibs are dependent on the CLL.

*Control Logic* isthat part of the TitanSim software that is specific to a particular TitanSim application program. It is usually not directly reusable and is built upon the CLL and several AppLibs.

*Entities* are “things/objects” that are simulated by the LGen. Such “things” are usually the “users”, “calls”, “servers”, “terminations”, or anything whose external behavior is to be simulated by the LGen. Usually an LGen simulates more than one entity.

*Entity context* is an index-able generic data record describing an entity. It serves as a hub connecting all information related to the entity together by referring the respective associated *🡪 behavior contexts* and *🡪 FSM contexts* via their indices. The entity contexts are identified via their indices with respect to the entity context database stored in the v\_LGenBase\_entities component variable.

*Behavior context* is an index-able (set of) data record(s) that stores data and state information of a given entity with respect to a given *🡪 behavior type.* Storage for behavior contexts is to be provided by the respective AppLibs.

*Behavior type* is a concept for collectively referring a set of *🡪 behavior contexts, 🡪FSM inputs, 🡪 test steps,* andfunctions realizing methods conforming to given function signatures prescribed by the LGenBase feature. Behavior types are to be declared dynamically during run-time to the LGenBase by the component-type initialization founction of some AppLib. Usually an AppLib product has only one behavior type, but it is permitted for AppLibs to declare more than one behavior types, if necessary. Behavior types are identified by their indices determined by the order of declaration and the declarations are stored in v\_LGenBase\_behaviorTypes

*Entity type* is a named list of references to declared *🡪 behavior types*. As such, it declares what type of behavior contexts are available on entities of this type.

*Entity group* A group of entities with homogenous entity type. The indices of the entities of the group make a continuous interval that does not overlap with the indices range of any other entity group.

*Test step* Test steps are functions whose signature conforms to a given function type. They *must* not contain TTCN-3 statement with blocking semantics.

*FSM table* is a formal specification of the communication rules of Finite State Machines. These indexable records are stored in v\_LGenBase\_fsmTables. FSM tables realizing *🡪 traffic cases* must obey certain rules.

*Traffic case type* is a prototype used as an input for instantiating traffic case instances on a specific entity group. It defines what *🡪 FSM table* to use, what *🡪 entity type* must be used by the *🡪 entity group* where the *🡪 traffic case* is to be deployed

*Traffic case* is a behavior, defined by an FSM table, that is executed on a given entity group. All entities of the entity group participate in executing the the traffic case. An entity group may have more than one concurrent traffic cases.

*Scenario* is a collection of traffic cases deployed on an entity group

*TitanSim Variables* are enhanced component variables that make it possible to access the values of other Variables in remote compoents

*TitanSim Statistics* are Statistics of TitanSim Variables. Possible Statistics are among others minimum, maximum, mean, standard deviation, etc. and also the content of a Variable can be a Statistics itself. The values of such Statistics are automatically and periodically updated in the background.

*Generated load of a traffic case* is defined as the *aggregate* load generated by *🡪 traffic case instances* of the given *🡪 traffic case*

*Generated load of the traffic mix* is defined as the *aggregate* load generated by the individual*🡪 traffic cases* of the given *🡪 traffic mix*

*Programming style, event driven* A programming style not-so “natural” for most people. It is atypical for function tests. It focuses on one or more threads of execution, however, *not* in a sequential manner. Rather it defines states for execution where the program code wait for an event (any event) to happen, then based on the event it executes a state transition to a new state and continues to wait for another event.   
The flow of events *cannot be* understood easily just by reading the source code sequentially, therefore it is less natural for many programmer.

*Programming style, procedural* It is typical for functions tests. It focuses on a single thread of execution in a sequential manner. The “natural” programming style for most programmers, since the flow of events can be understood easily just by reading the source code sequentially.

*Traffic case instance* is a specific execution instance of the behavior (message sequence) defined by the *🡪 traffic case*

*Traffic generator PTC* It is a PTC that actually generate the messages toward the SUT during the performance test campaign

*Traffic group* is a named instance of a *🡪 traffic mix*. Such traffic group may have *🡪 TitanSim Parameters* that are common to each traffic cases within the group that realize the *🡪 traffic mix*. Such traffic group may have (potentially aggregate) *🡪 TitanSim Statistics* that provides (potentially aggregate) information about the execution of the *🡪 traffic cases* within the group. Note, however, that the *🡪 traffic cases* may have individual parameters and statistics, as well.

*Traffic mix* is a specific combination of *🡪 traffic cases* that are to be executed concurrently by TitanSim. Each traffic case has a corresponding *🡪 traffic weight assigned.* The traffic mix has a load-level target that is an aggregate of the individual load-levels of the individual *🡪 traffic cases*, whose load levels are the result of the execution of their *🡪 traffic case instances*

# TitanSim Core Load Library (TitanSim CLL)

## Introduction

TitanSim CLL is a framework containing set of generic features and functions needed when implementing TTCN-3 traffic generators. It provides built-in functionality/libraries for:

* handling a generic, customizable graphical user interface
* infrastructure for handling of parameters and statistics along the test configuration (including the visualization of parameters/statistics and functions/altsteps for handling user interactions)
* ready-to-use functions and methods for effective handling of:
  + Pool/queue variables
  + Scheduling (mass timer infrastructure)
* customisable, ready-to-use TTCN-3 components (and related functions/altsteps) covering common scenarios, like execution control, load regulation, central scheduling, etc.
* API to extend the library with project-specific features

### Detailed Feature Documentation

The CLL features are shortly explained further in this section. The detailed documentation for each feature can be found under the delivered directory *EPTF\_Core\_Library\_CNL113512/doc* with the following naming convention: *EPTF\_CLL\_<FeatureName>\_FD* for the Function Description and *EPTF\_CLL\_<FeatureName>\_UG* for the User Guide, respectively.

## Base

The EPTF Base component is a fundamental component that handles common tasks (e.g. managing cleanup functions, graceful termination, etc.) that should be implemented in all EPTF components. All EPTF components should extend this component (i.e. explicitly) or a component that extends this component (i.e. implicitly).

See ‎[8] for more details.

## Central Scheduling

The EPTF Central Scheduling feature makes it possible to write load test cases using similar methodology than in function test. Load testing means executing several traffic cases in parallel.

See ‎[31] for more details.

## Common

The EPTF Common feature contains common type and function definitions used by other EPTF features.

See ‎[9] for more details.

## Execution Control

The aim of the EPTF ExecCtrl feature is to provide common framework for configuring and handling multiple Load Generators in one test scenario.

See ‎[10] for more details.

## FreeBusyQueue

The EPTF Free Busy Queue feature provides dynamic memory allocation for the TTCN-3 language in an efficient way.

See ‎[11] for more details.

## HashMap

The EPTF HashMap feature provides access to a well-tested, industry standard *GCC* HashMap implementation. HashMap is a hashed associate container that associates object of type key with object of type data.

See ‎[12] for more details.

## HostAdmin

The aim of the EPTF Host Admin feature is to provide processor and memory usage measurement for a given host it is started on. Remote measurements are also possible using the feature by subscribing to EPTF Variables (see ‎2.27).

See ‎[13] for more details.

## LGenBase

The EPTF LGenBase feature is the basis for all load generators (LGens) components of TitanSim, i.e. this feature should be used to write efficient load generators.

See ‎[14] for more details.

## LoadRegulator

The EPTF Load Regulator feature makes it possible to regulate the CPS of a load generator based on the load of the SUT.

See ‎[15] for more details.

## Logging

The aim of the EPTF Logging feature is, to use the library provided logging framework for event-class based per-PTC log control. The feature manages the logging database, defines default log classes and functions without the extension of GUI.

See ‎[16] for more details.

## NameService

The EPTF Name Service feature makes it possible to register and query component references by name.

See ‎[18] for more details.

## PTCDeployment

The EPTF PTCDeployment implements the functionality that allows components to select a host, where new PTCs can be deployed according to specified tasks (called roles) in the PTCD component.

See ‎[19] for more details.

## RandomNArray

The aim of the EPTF Random N Array feature is to provide dynamic memory allocation for the TTCN-3 language in an efficient way. The EPTF Random N Array feature makes it possible to manage lists over ‘record of’ data structures easily using index-arithmetics.

See ‎[34]**Error! Reference source not found.** for more details.

## RedBlackTree

The EPTF Red Black Tree feature is a self-balancing binary search tree. It can be used to store indexes into an associated array, similar to the EPTF Free Busy Queue (see ‎2.6).

See ‎[20] for more details.

## Rendezvous

The aim of the EPTF Rendezvous feature is, to make a *Rendezvous service*. Rendezvous service provides a generic solution for synchronization among/between various entities either locally and remotely.

See ‎[30] for more details.

## RingBuffer

The EPTF Ring Buffer feature provides a general ring buffer (or circular buffer) implementation. Ring buffer is a [data structure](http://en.wikipedia.org/wiki/Data_structure) that uses a single, fixed-size [buffer](http://en.wikipedia.org/wiki/Buffer_%28computer_science%29) as if it were connected end-to-end. This structure can be easily used for buffering [data streams](http://en.wikipedia.org/wiki/Data_stream).

See ‎[21] for more details.

## Scheduler

The EPTF Scheduler feature makes it possible to schedule events (actions) that should happen at a given time measured from the start of the component or relative to the schedule of the event. The Scheduler is efficient, it uses only two TTCN-3 timers to implement scheduling.

See ‎[22] for more details.

## Semaphore

The EPTF Semaphore feature makes it possible to implement blocking functions and to perform synchronization in TTCN-3.

See ‎[23] for more details.

## StatCapture

The aim of the EPTF Statistics Capture feature is to group Statistics into capture groups and to print capture group contents regularly at sampling intervals to predefined capture files.

See ‎[24] for more details.

## StatHandler

StatHandler feature makes it possible to collect global, aggregated statistics via the existing EPTF Variable interface.

See ‎[33] for more details.

## StatMeasure

The aim of the EPTF Statistics Measure feature is to create Statistics (stand-alone or belonging to a Variable) and update their value when required.

See ‎[25] for more details.

## StatReplay

The aim of the EPTF Statistics Replay feature is to display a given Statistics from a given capture group on the chart in a slider window in the GUI.

See ‎[26] for more details.

## Time Profile Editor

The EPTF CLL Time Profile Editor is an application which allows the user to create and edit *time profile*s used by the EPTF CLL Execution Control ‎[10] component.

See ‎[32] for more details.

## Transport

The EPTF Transport Control feature makes it possible to

* Route the incoming and outgoing messages between the components
* Store messages and information in a FreeBusyQueue

Handle the UDP and IPL4 testports dynamically.

See ‎[29] for more details.

## UIHandler

With the EPTF UIHandler feature, the user can manage display and manipulation of subscribed Variables on the runtime GUI. Apart from this, it is even possible to change the runtime GUI. The user can add, remove, enable or disable GUI elements.

See ‎[27] for more details.

## Variable

The EPTF Variable feature makes it possible to access component variables in remote components and to create automatically calculated Variables triggered by the refresh of other Variables.

See ‎[28] for more details.

# Installation and Configuration

Since the EPTF CLL 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 the EPTF CLL features:

* Copy the files of the specific feature from the released *EPTF\_Core\_Library\_CNL113512/src* directory of the test suite or create symbolic links to them.
* Import the released demo application from *EPTF\_Core\_Library\_CNL113512/demo* or write your own application using the given CLL feature.
* Import required files from *EPTF\_Core\_Library\_CNL113512/src* from other EPTF CLL features as explained in the User Guide of the feature (see *EPTF\_Core\_Library\_CNL113512/doc).*
* Create Makefile or modify the existing one. For more details see the relevant section of the User Guide of the feature from *EPTF\_Core\_Library\_CNL113512/doc.*
* Edit the configuration file according to your needs, see the User Guide of the specific *EPTF\_Core\_Library\_CNL113512/doc* for configuration and module parameters.

Some of the features of the Core Library are using constants defined during compilation. These constants can be defined with the EPTF\_DEBUG switch which can be set in the CPPFLAGS section of the Makefile. If this macro is defined the application will run in debug mode otherwise not.

# References

See Section ‎2.1.1 for the location of feature documentations in the delivery VOB.

1. ETSI ES 201 873-1 v3.2.1 (2007-02)  
   The Testing and Test Control Notation version 3. Part 1: Core Language
2. 1/198 17-CRL 113 200 Uen  
   User Guide for the TITAN TTCN-3 Test Executor
3. 109 21-CNL 113 512-2 Uen   
   TitanSim CLL for TTCN-3 toolset with TITAN, Product Revision Information
4. 155 17-CNL 113 512 Uen   
   TitanSim CLL for TTCN-3 toolset with TITAN, Function Specification
5. TitanSim CLL for TTCN-3 toolset with TITAN, Reference Guide  
   <http://ttcn.ericsson.se/products/libraries.shtml>
6. 127 02-FCPCA 101 124 Uen  
   Project Specification for TitanSim R2
7. 100 50 FCPCA 101 124 Uen  
   Quick Study on TitanSim R2 development
8. 4/198 17-CNL 113 512 Uen  
   EPTF CLL Base, User Guide
9. 5/198 17-CNL 113 512 Uen  
   EPTF CLL Common, User Guide
10. 19/198 17-CNL 113 512 Uen  
    EPTF CLL ExecCtrl, User Guide
11. 6/198 17-CNL 113 512 Uen  
    EPTF CLL Free Busy Queue, User Guide
12. 1/198 17-CNL 113 512 Uen  
    EPTF CLL HashMap, User Guide
13. 20/198 17-CNL 113 512 Uen  
    EPTF HostAdmin, User Guide
14. 7/198 17-CNL 113 512 Uen  
    EPTF CLL LGenBase, User Guide
15. 21/198 17-CNL 113 512 Uen  
    EPTF CLL LoadRegulator, User Guide
16. 8/198 17-CNL 113 512 Uen  
    EPTF CLL Logging, User Guide
17. 9/198 17-CNL 113 512 Uen  
    EPTF CLL LoggingUI, User Guide
18. 10/198 17-CNL 113 512 Uen  
    EPTF CLL NameService, User Guide
19. 3/198 17-CNL 113 512 Uen  
    EPTF CLL PTCDeployment, User Guide
20. 12/198 17-CNL 113 512 Uen  
    EPTF CLL RedBlackTree, User Guide
21. 13/198 17-CNL 113 512 Uen  
    EPTF CLL RingBuffer, User Guide
22. 14/198 17-CNL 113 512 Uen  
    EPTF CLL Scheduler, User Guide
23. 22/198 17-CNL 113 512 Uen  
    EPTF CLL Semaphore, User Guide
24. 15/198 17-CNL 113 512 Uen  
    EPTF CLL StatCapture, User Guide
25. 16/198 17-CNL 113 512 Uen  
    EPTF CLL StatMeasure, User Guide
26. 17/198 17-CNL 113 512 Uen  
    EPTF CLL StatReplay, User Guide
27. 2/198 17-CNL 113 512 Uen  
    EPTF CLL UIHandler, User Guide
28. 18/198 17-CNL 113 512 Uen  
    EPTF CLL Variable, User Guide
29. 25/198 17-CNL 113 512 Uen  
    EPTF CLL Transport, User Guide
30. 23/198 17-CNL 113 512 Uen  
    EPTF CLL Rendezvous, User Guide
31. 24/198 17-CNL 113 512  
    EPTF CLL Central Scheduling, User Guide
32. 26/198 17-CNL 113 512 Uen  
    EPTF CLL Time Profile Editor, User Guide
33. 27/198 17-CNL 113 512 Uen  
    EPTF CLL Stat Handler, User Guide
34. 28/198 17-CNL 113 512 Uen  
    EPTF CLL Random N Array, User Guide