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

EPTF CLL Semaphore, User Guide

Contents

[1 Introduction 2](#_Toc211150887)

[1.1 Revision history 2](#_Toc211150888)

[1.2 About this Document 2](#_Toc211150889)

[1.2.1 How to Read this Document 2](#_Toc211150890)

[1.2.2 References 2](#_Toc211150891)

[1.2.3 Abbreviations 3](#_Toc211150892)

[1.2.4 Terminology 3](#_Toc211150893)

[1.3 System Requirements 3](#_Toc211150894)

[2 Semaphore 3](#_Toc211150895)

[2.1 Overview 3](#_Toc211150896)

[2.2 Description of files in this feature 3](#_Toc211150897)

[2.3 Description of required files from other features 4](#_Toc211150898)

[2.4 Installation 4](#_Toc211150899)

[2.5 Configuration 4](#_Toc211150900)

[2.6 Usage 5](#_Toc211150901)

[3 Error messages 5](#_Toc211150902)

[4 Warning messages 5](#_Toc211150903)

[5 Examples 6](#_Toc211150904)

[5.1 Configuration file 6](#_Toc211150905)

[5.2 Demo Module 6](#_Toc211150906)

[5.2.1 Detailed description 6](#_Toc211150907)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-12-05 | PA1 | First draft version | <ETHJGI> |
| 2007-12-06 | PA2 | Updated after review | <ETHJGI> |
|  |  |  |  |
|  |  |  |  |

## About this Document

### How to Read this Document

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

### References

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. 198 17-CNL 113 512 Uen  
   TitanSim CLL for TTCN-3 toolset with TITAN, User Guide
6. 22/155 16-CNL 113 512  
   EPTF CLL Semaphore Function Description
7. TitanSim CLL for TTCN-3 toolset with TITAN, Reference Guide  
   <http://ttcn.ericsson.se/products/libraries.shtml>

### Abbreviations

CLL Core Load 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]

### Terminology

*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

*Semaphore* The TTCN-3 way to solve blocking scenarios and perform synchronization

## System Requirements

In order to use the Semaphore feature the system requirements listed in TitanSim CLL User Guide ‎[5] should be fulfilled.

# Semaphore

## Overview

The EPTF CLL Semaphore component is a fundamental component providing an implementation for Semaphore in a load test environment.

## Description of files in this feature

The EPTF CLL Semaphore API includes the following files:

* Semaphore
  + EPTF\_CLL\_Semaphore\_Definitions.ttcn: This TTCN-3 module contains common type definitions that should be used in all Semaphore Components.
  + EPTF\_CLL\_Semaphore\_Functions.ttcn: This TTCN-3 module contains the implementation of Semaphore main functions.

## Description of required files from other features

The Semaphore feature is part of the TitanSim EPTF Core Load Library (CLL). It relies on several features of the CLL. The user has to obtain the products/files to be found under the respective feature names:

* Base
* FreeBusyQueue

## Installation

Since EPTF CLL Semaphore 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 CLL Semaphore:

* Copy the files listed in section [‎2.2, ‎2.3] to the directory of the test suite or create symbolic links to them.
* Import the NameService demo or write your own application using NameService.
* 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 behaviour is determined via 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].

The EPTF Semaphore feature defines TTCN-3 module parameters as defined in ‎[2] clause 4. Actual values of these parameters – when there is no default value or a different from the default actual value to be used – shall be given in the [MODULE\_PARAMETERS] section of the configuration file.

The EPTF Semaphore feature defines the following module parameters:

**boolean EPTF\_CLL\_Semaphore\_Definitions.tsp\_debugVerbose\_PoolMgmt**

The debug messages of the semaphore pool can be enabled by this module parameter. Default: false.

## Usage

To use the EPTF Semaphore feature

* Extend one of your component with the EPTF\_Semaphore\_CT
* call the init function of the Semaphore to initialize the feature
* use its public functions to create/wait for/unlock a semaphore

Note, that the init function registers the cleanup function of the Semaphore, so it is not necessary to call it explicitly.

Do not access the component variables in EPTF\_Semaphore\_CT directly! Use the API functions instead.

When the f\_EPTF\_Semaphore\_waitForUnlock function is called on a locked semaphore it blocks execution until the semaphore is unlocked. To be able to unlock the Semaphore the altstep that handles the unlock event and calls the unlock function should be activated as default before calling the wait function.

The basic steps you have to do to use the Semaphore when you have to wait the response after the request was sent:

* Create a semaphore and store its id (or send with the message, and put it into the response)
* send the message of which response should be received
* in the default altstep write the handler function of the response so that it unlocks the semaphore
* In the place where you want to be sure, that the response is received call the waitForUnlock function.

# Error messages

Please note, that besides the below described error messages, error messages shown in ‎[2] or those of other used features or product may also appear.

There are no error messages defined by the Semaphore feature.

# Warning messages

Please note, that besides the below described warning messages, warning messages shown in ‎[2] or those of other used features or product may also appear.

**Semaphore is already locked: <id>**

The lock function is called for a semaphore that is already locked.

**Semaphore <idx> was not unlocked before max waiting time expired. Semaphore unlocked**

The wait function reached the max waiting time but the Semaphore was not unlocked. The semaphore is unlocked automatically to let the wait function run.

# Examples

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

* EPTF\_Semaphore\_test.ttcn

## Configuration file

The used configuration file (main.cfg) sets only the parameters in the [LOGGING] section. No special configuration is needed.

## Demo Module

The demo module EPTF\_Semaphore\_test.ttcn illustrates a typical usage of the Semaphore feature.

The usage of create/lock/unlock/wait functions are shown.

### Detailed description

In the demo two semaphore are created. The first semaphore will be unlocked and the default maxWait time (infinity) is used in the wait function. The second semaphore is not unlocked, but the maxWait time is set the 1 second. After the semaphores are created the default altstep is activated and the wait function of the first semaphore is called with infinity wait time.

The event that corresponds to a received message is t\_wait timer’s timeout event. When this event handled in the activated default altstep it will wait for the second semaphore to be unlocked. This never happens, so its wait function will exit because the maxWait time elapses. Then it unlocks the first semaphore and calls repeat. At this point as the semaphore is unlocked its waiting function continues and the verdict is set to pass.