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

EPTF CLL Command Line Interface, User Guide

Contents

[1 Introduction 2](#_Toc342136690)

[1.1 Revision history 2](#_Toc342136691)

[1.2 About this Document 2](#_Toc342136692)

[1.2.1 How to Read this Document 2](#_Toc342136693)

[1.2.2 References 2](#_Toc342136694)

[1.2.3 Abbreviations 3](#_Toc342136695)

[1.2.4 Terminology 3](#_Toc342136696)

[1.3 System Requirements 3](#_Toc342136697)

[2 The Command Line Interface 3](#_Toc342136698)

[2.1 Overview 3](#_Toc342136699)

[2.2 Description of files in this feature 3](#_Toc342136700)

[2.3 Description of required files from other CLL features 3](#_Toc342136701)

[2.4 Description of required external features outside CLL 4](#_Toc342136702)

[2.5 Installation 4](#_Toc342136703)

[2.6 Configuration 4](#_Toc342136704)

[2.6.1 tsp\_EPTF\_CLL\_CLI\_ShowWelcome 5](#_Toc342136705)

[2.6.2 tsp\_EPTF\_CLI\_Functions\_debug 5](#_Toc342136706)

[2.6.3 tsp\_EPTF\_CLI\_Client\_Functions\_debug 5](#_Toc342136707)

[3 Usage 5](#_Toc342136708)

[3.1 Using the functional interface 5](#_Toc342136709)

[3.1.1 Initialization 5](#_Toc342136710)

[3.1.2 Registering a command 5](#_Toc342136711)

[3.1.3 The help of a command 6](#_Toc342136712)

[3.1.4 The command handler 6](#_Toc342136713)

[3.1.5 Sending messages to the display terminal 6](#_Toc342136714)

[3.2 Using the terminals 6](#_Toc342136715)

[3.2.1 CLI built-in commands 7](#_Toc342136716)

[4 Error messages 8](#_Toc342136717)

[5 Warning messages 8](#_Toc342136718)

[6 Examples 9](#_Toc342136719)

[6.1 Example Configuration 9](#_Toc342136720)

[6.2 Demo Module 9](#_Toc342136721)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2010-03-30 | PA1 | First draft version | ETHJGI |
| 2012-09-19 | PB1 | Alias support | EBALLUG |

## About this Document

### How to Read this Document

This is the User Guide for the Command Line Interface 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 Command Line Interface 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 v4.1.2 (2009-07)  
   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. 29/155 16-CNL 113 512  
   EPTF CLL Command Line Interface, Function Description
7. TitanSim CLL for TTCN-3 toolset with TITAN, Reference Guide  
   <http://ttcn.ericsson.se/products/libraries.shtml>
8. 198 17-CNL 113 320 Uen   
   Telnet Test Port for TTCN-3 Toolset with TITAN, User Guide

### Abbreviations

CLI Command Line Interface

CLL Core Load Library

EPTF Ericsson Performance Test Framework, formerly TITAN Load Test Framework

TitanSim Ericsson Performance 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

*Command Line Interface* is a user interface accessible via telnet protocol where the behavior of TitanSim can be changed using commands

## System Requirements

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

# The Command Line Interface

## Overview

The EPTF CLI implements the functionality to be able to control TTCN3 code via a telnet interface by issuing commands on the connected terminal.

## Description of files in this feature

The EPTF CLL CLI includes the following files:

* EPTF\_CLL\_CLI\_Definitions.ttcn: This TTCN-3 module contains common type definitions used by the CLI feature.
* EPTF\_CLL\_CLI\_Functions.ttcn: This TTCN-3 module contains the implementation of CLI functions.

## Description of required files from other CLL features

The EPTF CLI feature depends on the following features of EPTF:

* EPTF\_CLL\_Base\_Definitions
* EPTF\_CLL\_Variable\_Definitions,
* EPTF\_CLL\_Semaphore\_Definitions
* EPTF\_CLL\_Common\_Definitions
* EPTF\_CLL\_Logging\_Definitions

## Description of required external features outside CLL

* TCCUsefulFunctions\_CNL113472
* TELNETasp\_CNL113320 test port

## Installation

Since EPTF\_CLL\_CLI 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\_CLI:

* Copy the files listed in section [‎2.2] and [‎2.3] to the directory of the test suite or create symbolic links to them.
* Import the CLI demo or write your own application using EPTF CLI.
* 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 [‎2.6].

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

The EPTF CLI feature uses the Telnet test port. This test port requires test port parameters to configure it. See the Telnet test port user guide ‎[8] for more details. Note that the CTRL\_MODE parameter of the telnet test ports used by the CLI should be set to ‘server’ to be able to connect to them via a telnet client.

The EPTF CLI feature defines the following module parameters.

### tsp\_EPTF\_CLL\_CLI\_ShowWelcome

This is a boolean parameter which can be used to enable a welcome message when connected to the CLI terminals. Default value: false (disabled).

### tsp\_EPTF\_CLI\_Functions\_debug

Module parameter to switch on debug logging in CLI. Default: false (disabled).

### tsp\_EPTF\_CLI\_Client\_Functions\_debug

Module parameter to switch on debug logging in CLI\_Client. Default: false (disabled).

# Usage

## Using the functional interface

To be able to use the CLI feature at least one component in the system has to extend the EPTF\_CLI\_CT component. This component creates the telnet ports for the normal and the display terminal. In addition to the CLI component the CLI command can be registered in a component that extends the EPTF\_CLI\_Client\_CT component.

### Initialization

Before using the functions of these components they have to be initialized by the init functions of the corresponding CLI components.

To be able to use the command line interface terminal in case of error, it is recommended to place the CLI component to the MTC since this component terminates last.

### Registering a command

New command can be registered for a given terminal (i.e. CLI component) from CLI Clients (see ‎[6]). The command names has to be unique on the CLI used and should not contain spaces.

The help message given during registration should be a short description of the command because this help string will be printed when the “help <command>” is issued on the terminal. More detailed help can be printed by implementing the handling of the “help” argument for the command. In this case the detailed help is returned by executing “<command> help” on the terminal.

If the CLI Client init function is called with CLI component set to “null” and no CLI was given in the register function, or the CLI Client init is not called at all or cleanup has been called, it is not possible to register commands, the CLI is disabled.

### The help of a command

Detailed help about a command can be printed by ‘<command> help’ if the handler function of the command handles the ‘help’ as argument.

### The command handler

The command handler function registered for a command can modify the parameter containing the result of the command. This result will be printed on the terminal on which the command was executed. If the result is not changed, “<command> done.” will be printed. Also the function has to return an error code. This error code notifies the CLI about the success of the command. If the command was handled successfully the function should return zero, otherwise a positive number. Positive numbers are treated as erroneous exit codes and will appear on the CLI terminal automatically.

If the handler function of a command has to wait for a message on a test port, it is not recommended to call altsteps that contain test port receive statements in the handler function. Instead the Semaphore feature of the CLL can be used to block the execution until the response arrives. When the response is handled in a default altstep it should also unlock the semaphore. This has to be done to avoid deadlocks on the test port.

### Sending messages to the display terminal

If messages should be sent to the terminal periodically, there is a display terminal provided by the CLI. Messages to the display terminal can be sent from the CLI Client. The display terminal can be used to prevent flooding the normal terminal. It is possible to enter the same commands on the display terminal as on the normal terminal.

## Using the terminals

The CLI provides two terminals: a normal terminal, and a display terminal to show periodic printouts. The terminals can be accessed by a telnet client on the ports configured. Commands defined in the CLI can be executed on these terminals.

If the execution of the commands take some time, it is possible that the result of a command issued later will appear before the results of commands issued earlier are printed on the terminal.

### CLI built-in commands

The CLI defines built-in commands to get basic help, exit from the terminal and stop the test execution. The built-in commands cannot be redefined i.e. to register commands with the same names on the CLI client.

#### The help command

The help command can be used to print a general help message and list all available commands. To get help about a specific command, the name of the command should be passed to the help command: ‘help <command>’.

The help command will show the basic help message specified in the command registration function. To get more detailed help, the command handle should handle an argument like “help” and print out a more detailed information about the usage of the command.

#### The exit and quit commands

The exit and quit commands can be used to close the terminal session. The test is not stopped. It is possible to reconnect the terminal later.

The quit command name is not case sensitive.

#### The stop and stopAll commands

The stop and stopAll commands stop the execution of the test case. All components (that uses the Base feature of the CLL) will be stopped and the test is terminated if the stop or stopAll commands are executed on the terminal.

The stop command name is not case sensitive.

#### The alias command

To create an alias to a command the alias command can be used. There is an example for simple usage in the first line below. ‘Help’ command can be substituted with ‘h’ from then on. Command parts (ex. arguments) can also be aliased, but must be signed with special character tags ($) in case of usage. In the third line we have an alias both for help (h) and stop (s). As ‘s’ is signed to be substituted before command execution, thus the line will retrieve help message of command stop.   
Note that there is no need of encapsulating ‘h’ between ‘$’ tags, because the first word of the command is auto substituted if it is an alias.

Example:

alias help h  
alias stop s  
h $s$

#### The unalias command

Unalias removes a registered alias.

Example: unalias h

#### The lsalias command

Lsalias prints information about a registered alias.

Example: lsalias h

#### The allalias command

Allalias lists all registered aliases in the system.

# 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.

The CLI does not have any error messages.

# 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.

**WARNING: Invalid ASP received from port <telnetPort>. Ignored.**

An unexpected. message arrived on the telnet port, and it is ignored. This warning is printed when two or more clients connect to the same terminal.

**WARNING: Invalid message received on CLI telnet port. Ignored.**

An unexpected message arrived on the telnet port. It is ignored.

**WARNING: f\_EPTF\_CLI\_Client\_sendCommandDisplay: Cannot send text to display: command not found: <commandName>**

The display message could not be shown. The command name does not registered, it has no display.

# Examples

## Example Configuration

The following shows example test port configuration settings to configure the Command Line Interface telnet ports:

\*.EPTF\_CLI\_TELNET\_PCO.CTRL\_PORTNUM := "17100"

\*.EPTF\_CLI\_TELNET\_PCO.CTRL\_LOGIN\_SKIPPED := "yes"

\*.EPTF\_CLI\_TELNET\_PCO.CTRL\_USERNAME\_CLIENT := "ttcn"

\*.EPTF\_CLI\_TELNET\_PCO.CTRL\_PASSWORD\_CLIENT := "ttcn"

\*.EPTF\_CLI\_TELNET\_PCO.CTRL\_SERVER\_PROMPT := "TTCN> "

\*.EPTF\_CLI\_TELNET\_PCO.CTRL\_MODE := "server"

Similar settings with different CTRL\_PORTNUM value can be used to configure the display terminal:

\*.EPTF\_CLI\_displayTELNET\_PCO.CTRL\_PORTNUM := "17101"

\*.EPTF\_CLI\_displayTELNET\_PCO.CTRL\_LOGIN\_SKIPPED := "yes"

\*.EPTF\_CLI\_displayTELNET\_PCO.CTRL\_USERNAME\_CLIENT := "ttcn"

\*.EPTF\_CLI\_displayTELNET\_PCO.CTRL\_PASSWORD\_CLIENT := "ttcn"

\*.EPTF\_CLI\_displayTELNET\_PCO.CTRL\_SERVER\_PROMPT := "TTCN> "

\*.EPTF\_CLI\_displayTELNET\_PCO.CTRL\_MODE := "server"

If the TELNET test port is used to connect to these ports, the CTRL\_MODE has to be set to “client”, CTRL\_PORTNUM and CTRL\_SERVER\_PROMPT has to be exactly the same as given here and the CTRL\_DETECT\_SERVER\_DISCONNECTED has to be set to “yes”.

## Demo Module

The demo module in the demo directory illustrates a typical usage of the EPTF CLI feature. It demonstrates how to register a command, how a command handler should work, how to use the display terminal etc.