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

EPTF CLL Common, Function Description

Contents

[1 Introduction 3](#_Toc248296521)

[1.1 Revision history 3](#_Toc248296522)

[1.2 How to Read this Document 3](#_Toc248296523)

[1.3 References 3](#_Toc248296524)

[1.4 Scope 3](#_Toc248296525)

[1.5 Recommended way of reading 4](#_Toc248296526)

[1.6 Typographical conventions 4](#_Toc248296527)

[1.7 Abbreviations 4](#_Toc248296528)

[1.8 Terminology 4](#_Toc248296529)

[2 General Description 4](#_Toc248296530)

[3 Common Types 4](#_Toc248296531)

[3.1 Default List 5](#_Toc248296532)

[3.2 Integer List 5](#_Toc248296533)

[3.3 Integer Array 2D 5](#_Toc248296534)

[3.4 Charstring List 5](#_Toc248296535)

[3.5 Float List 5](#_Toc248296536)

[3.6 Boolean List 5](#_Toc248296537)

[3.7 LED Colors 6](#_Toc248296538)

[3.8 Status LED 6](#_Toc248296539)

[3.9 Parameter Range Descriptor 6](#_Toc248296540)

[3.10 Parameter Range Descriptor List 7](#_Toc248296541)

[3.11 RndValues component 7](#_Toc248296542)

[3.12 Debug switch 7](#_Toc248296543)

[3.13 IndexArray 7](#_Toc248296544)

[3.14 Summary Table of all Types in Common 8](#_Toc248296545)

[4 Functional Interface 8](#_Toc248296546)

[4.1 Naming Conventions 8](#_Toc248296547)

[4.2 Public Functions 8](#_Toc248296548)

[4.2.1 Initializing 8](#_Toc248296549)

[4.2.2 Reset Range Parameter 8](#_Toc248296550)

[4.2.3 Incrementing Range Iterators 9](#_Toc248296551)

[4.2.4 Increasing the least significant range 9](#_Toc248296552)

[4.2.5 Initialize the RndValues component 9](#_Toc248296553)

[4.2.6 Get the next generated random number 9](#_Toc248296554)

[4.2.7 IndexArray functions 9](#_Toc248296555)

[4.2.8 Logging 11](#_Toc248296556)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-12-03 | PA1 | First draft version | EGBOTAT |
| 2007-12-06 | PA2 | Final version after review | EGBOTAT |
| 2008-01-28 | PA3 | Updated | ENORPIN |
| 2008-11-21 | PB1 | Additions | EGBOTAT |
| 2009-09-07 | PC1 | IndexArray | EBENMOL |
| 2009-11-04 | PC2 | EPTF\_DEBUG switch | EANDRKS |
| 2009-12-11 | PC3 | Error message checking added | ETHJGI |

## How to Read this Document

This is the Function Description for the Common of the Ericsson Performance Test Framework (TitanSim), Core Load Library (CLL). TitanSim CLL is developed for the TTCN-3 ‎[1] Toolset with TITAN ‎[2]. For more information on the TitanSim CLL please consult the Product Revision Information ‎[3].

## 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. TitanSim CLL for TTCN-3 toolset with TITAN, Reference Guide  
   http://ttcn.ericsson.se/products/libraries.shtml

## Scope

This document is to specify the content and functionality of the Common feature of the TitanSim CLL.

## Recommended way of reading

The readers are supposed to get familiar with the concept and functionalities of TitanSim CLL ‎[4]. They should get familiar with the list of acronyms and the glossary in Section ‎1.7 and ‎1.8, respectively.

## Typographical conventions

Important concepts are denoted by *italic* font wherever they are first used in the given context.

## 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

# General Description

This document specifies the Common feature of the TitanSim CLL.

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

# Common Types

The Common feature mainly serves as a type definition module for commonly used types. This chapter describes these types.

## Default List

The type

EPTF\_DefaultList

provides a container for arbitrary number of default. Its type is a ‘record of integer’.

## Integer List

The type

*EPTF\_IntegerList*

provides a container for arbitrary number of integers. Its type is ‘record of default’.

## Integer Array 2D

The type

*EPTF\_IntegerArray2D*

provides a container for arbitrary number of integers in a 2D array. Its type is ‘record of EPTF\_IntegerList’.

## Charstring List

The type

*EPTF\_CharstringList*

provides a container for arbitrary number of charstrings. Its type is ‘record of charstring’.

## Float List

The type

*EPTF\_FloatList*

provides a container for arbitrary number of floats. Its type is ‘record of float’.

## Boolean List

The type

*EPTF\_BooleanList*

provides a container for arbitrary number of booleans . Its type is ‘record of boolean’.

## LED Colors

The type

*EPTF\_LEDColors*

is an enumerated type for the known LED colors on the user interface.

Possible values:

* led\_blue
* led\_black
* led\_yellow
* led\_green
* led\_red

## Status LED

The type

*EPTF\_StatusLED*

is a record type for status LEDS with text.

Elements of the status led:

* EPTF\_LedColors color – the color of the status led
* charstring text – optional text for the status led

## Parameter Range Descriptor

The type

*EPTF\_ParamRangeDescriptor*

is a record type defining an integer parameter range.

Its elements are:

* charstring name – the name of the range
* integer baseOffset – the first value of the range
* integer count – the size of the range
* integer iterator – the actual value

## Parameter Range Descriptor List

The type

*EPTF\_ParamRangeDescriptorList*

provides a container for arbitrary number of parameter range descriptors. Its type is ‘record of EPTF\_ParamRangeDescriptor’.

## RndValues component

The type

*EPTF\_Common\_RndValues\_CT*

provides a container for the pre generated random numbers.

## Debug switch

The constant

*c\_EPTF\_Common\_debugSwitch*

can be used to guard debug code or logging in an if statement. The code guarded this way will be excluded from the compiled executable if  
–DEPTF\_DEBUG is not added to CPPFLAGS\_TTCN3 and –O2 is added to CXXFLAGS in the Makefile.

## IndexArray

The type

*EPTF\_Common\_IndexArray*

provides a record type to help quick storage and retrieval of indexes from a list. This type with the associated functions is practical when the count of the elements to be handled is less or about 10-20. The value of the indexes can be any nonnegative integer number or -1. The index value of -1 is meaning that the given index is not used (or free). If the index value is bigger than -1, then the element is used or valid (or busy). First busy index can be used to get the first valid element in the list. The constant c\_EPTF\_emptyCommon\_IndexArray can be used to create a free IndexArray.

The elements of IndexArray are:

* EPTF\_IntegerList values – list of indexes and their values
* integer firstBusy - first busy index in the list above

## Summary Table of all Types in Common

|  |  |
| --- | --- |
| Type name | Description |
| EPTF\_DefaultList | record of default |
| EPTF\_IntegerList | record of integer |
| EPTF\_IntegerArray2D | record of EPTF\_IntegerList |
| EPTF\_CharstringList | record of charstring |
| EPTF\_FloatList | record of float |
| EPTF\_BooleanList | record of boolean |
| EPTF\_LEDColors | enumerated LED colors |
| EPTF\_StatusLED | status led record |
| EPTF\_ParamRangeDescriptor | integer parameter range |
| EPTF\_ParamRangeDescriptorList | record of integer parameter range |
| EPTF\_Common\_RndValues\_CT | component |
| c\_EPTF\_Common\_debugSwitch | boolean constant |
| EPTF\_Common\_IndexArray | record of free or busy indexes |

# Functional Interface

Apart from this description a cross-linked reference guide for the TitanSim CLL Functions can be reached for on-line reading ‎[5].

## Naming Conventions

All functions have the prefix f\_EPTF\_Common\_.

## Public Functions

### Initializing

To initialize the Common feature call the function

f\_EPTF\_Common\_init()

### Reset Range Parameter

f\_EPTF\_Common\_resetParamRanges(pl\_rangeList)

This function can be used to reset range iterators to the base offset of the ranges.

### Incrementing Range Iterators

f\_EPTF\_Common\_incParamRanges(pl\_rangeList)

This function can be used to increment the least significant range iterator. It handles carry if necessary.

### Increasing the least significant range

f\_EPTF\_Common\_fillWeightedBuckets(pl\_weightList, pl\_nrOfElements, pl\_result)

This function can be used to increment the least significant range iterator. It handles carry if necessary.

### Initialize the RndValues component

f\_EPTF\_Common\_RndValues\_init\_CT()

This function can be used to initialize the RndValues component.

### Get the next generated random number

f\_EPTF\_Common\_RndValues\_getNextRndValue\_CT()

This function can be used to get the next pre generated random number. The component has to be initialized with the f\_EPTF\_Common\_RndValues\_init\_CT function before to use of this function.

### IndexArray functions

#### Obtaining a free index

f\_EPTF\_Common\_IndexArray\_getOrCreateFreeSlot()

This function can be used to get a free index (an index which content is -1) from the IndexArray record. If there is no free index, the function creates a new free index and returns it.

#### Get the content of an index

f\_EPTF\_Common\_IndexArray\_getElement()

This function can be used to retrieve the content of the specified index from the IndexArray record. If the index is not found, the returned value will be -1.

#### Set the content of an index

f\_EPTF\_Common\_IndexArray\_setElement()

This function can be used to set the content of the specified index in the IndexArray record. If any other free indexes are created their content will be -1.

#### Obtaining a free index and set its content

f\_EPTF\_Common\_IndexArray\_setNewElement()

This function can be used to get a free index (an index which value is -1) from the IndexArray record. If there is no free index, the function creates a new free index. The content of the index will be the data value given in the function. If no data is given the content will be -1.

#### Free the content of an index

f\_EPTF\_Common\_IndexArray\_freeElement

This function can be used to free the given index. The content of the given index will be -1.

#### Checking the emptiness of IndexArray record

f\_EPTF\_Common\_IndexArray\_arrayIsEmpty()

This function can be used to check whether the IndexArray record is empty (containing only -1 values).

#### Get the first busy index

f\_EPTF\_Common\_IndexArray\_getFirstBusyIdx()

This function can be used to get the first index in the record which is busy (containing nonnegative value).

#### Get the next busy index

f\_EPTF\_Common\_IndexArray\_getNextBusyIdx()

This function can be used to get the index of the first busy element which is bigger than the given index.

### Logging

#### Logging an error/warning/user messages

To log an error/warning or user message call the appropriate function:

f\_EPTF\_Common\_error(pl\_message)

f\_EPTF\_Common\_warning(pl\_message)

f\_EPTF\_Common\_user(pl\_message)

#### Determine the number of errors

The number of errors occurred is returned by the function

f\_EPTF\_Common\_nofErrorMsgs()

#### To retrieve a certain error message

A given error message text can be retrieved by the function

f\_EPTF\_ Common\_getErrorMsg(pl\_errorNum)

where pl\_errorNum gives the id of the error message. The id if the error message starts from zero, i.e. it is zero for the first error, one for the second an so on.

#### To check if an error message matches with a given pattern

To check if an error message matches with a given pattern, the function

f\_EPTF\_ Common\_checkExpectedError(pl\_expectedError, pl\_errorNum)

can be used. The pl\_expectedError is the error pattern that is matched with the occurred error with id pl\_errorNum. The id of the errors starts from zero. The function returns true if the pattern matches and false if it does not.Summary Table of all public functions for EPTF Common

Table 1 Summary of Common functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_Common\_resetParamRanges | reset range iterators |
| f\_EPTF\_Common\_incParamRanges | increment range iterator |
| f\_EPTF\_Common\_fillWeightedBuckets | increasing the least significant range |
| f\_EPTF\_Common\_RndValues\_init\_CT | initialize the RndValues component |
| f\_EPTF\_Common\_RndValues\_getNextRndValue | get the next pre generated random number |

Table 2 Summary of Common IndexArray functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_Common\_IndexArray\_getOrCreateFreeSlot | obtaining a free index |
| f\_EPTF\_Common\_IndexArray\_getElement | get the content of an index |
| f\_EPTF\_Common\_IndexArray\_setElement | set the content of an index |
| f\_EPTF\_Common\_IndexArray\_setNewElement | obtaining a free index and set its content |
| f\_EPTF\_Common\_IndexArray\_freeElement | free the content of an index |
| f\_EPTF\_Common\_IndexArray\_arrayIsEmpty | checking the emptiness of IndexArray record |
| f\_EPTF\_Common\_IndexArray\_getFirstBusyIdx | get the first busy index |
| f\_EPTF\_Common\_IndexArray\_getNextBusyIdx | get the next busy index |