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

EPTF CLL LGenBase, User Guide

Contents

[1 Introduction 2](#_Toc453593782)

[1.1 Revision history 2](#_Toc453593783)

[1.2 About this Document 2](#_Toc453593784)

[1.2.1 How to Read this Document 2](#_Toc453593785)

[1.2.2 Abbreviations 3](#_Toc453593786)

[1.2.3 Terminology 3](#_Toc453593787)

[1.3 System Requirements 6](#_Toc453593788)

[2 LGenBase 6](#_Toc453593789)

[2.1 Overview 6](#_Toc453593790)

[2.1.1 Burst calculation 6](#_Toc453593791)

[2.1.2 External templates 8](#_Toc453593792)

[2.2 Description of files in this feature 8](#_Toc453593793)

[2.3 Description of required features 10](#_Toc453593794)

[2.4 Installation 11](#_Toc453593795)

[2.5 Configuration 11](#_Toc453593796)

[2.6 Usage 13](#_Toc453593797)

[2.6.1 LGenBase 13](#_Toc453593798)

[2.6.2 LGenBaseStats 13](#_Toc453593799)

[2.6.3 LGenBaseStatsUI 14](#_Toc453593800)

[2.7 DTE handling in LGenBase 14](#_Toc453593801)

[3 Error messages 14](#_Toc453593802)

[4 Warning messages 21](#_Toc453593803)

[5 Examples 23](#_Toc453593804)

[5.1 Configuration file 24](#_Toc453593805)

[5.2 Demo Module 24](#_Toc453593806)

[6 References 24](#_Toc453593807)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-12-06 | PA1 | First draft version | EGBOZIE, ELSZSKU |
| 2007-12-06 | PA2 | Updated after review | EGBOZIE |
| 2008-02-01 | PA3 | LGenBaseStats feature | ELSZSKU |
| 2008-02-18 | PA4 | LGenBaseStatsUI feature | ELSZSKU |
| 2008-02-26 | PA5 | Usage of LGenBaseStats and LGenBaseStatsUI features | ELSZSKU |
| 2008-03-10 | PA6 | Required files updated | ELSZSKU |
| 2009-01-22 | PB1 | New file structure | ELSZSKU |
| 2009-02-06 | PB2 | External templates feature added | EISTFAL |
| 2009-02-26 | PB3 | Burst calculation and Phase feature | EBENMOL |
| 2009-04-23 | PB4 | External templates correction | EISTFAL |
| 2012-07-05 | PC1 | DebugLight module parameter | ELSZSKU |
| 2014-06-26 | PD1 | External template new tsp | EJNOSVN |
| 2014-09-22 | PD2 | Added section DTE handling in LGenBase | EDORNAG |
| 2014-10-09 | PD3 | Documentation numbers updated | ESZILSZ |
| 2014-11-21 | PD4 | Overview chapter updated | ESZILSZ |
| 2014-11-21 | PD5 | Undo the PD4 update and add DTE handling comment | ESZILSZ |
| 2014-12-02 | D | Updated for release | ESZILSZ |
| 2016-01-19 | PE1 | Switch for ILog feature added | EZOLZSI |
| 2016-01-28 | PE2 | Examples updated | ESZILSZ |
| 2016-03-22 | PE3 | Fixed position based substitution for external template | EZOLZSI |
| 2016-06-09 | E | Updated for release | ESZILSZ |

## About this Document

### How to Read this Document

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

### Abbreviations

CLL Core Library

EPTF Ericsson Performance Test Framework

TitanSim Ericsson Performance Test Framework

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

### Terminology

*EPTF 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.

*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 stores index cross-references to *🡪 behavior contexts* and it directly stores the *🡪 FSM contexts* of the entity. 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 function 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 the same entity type. The indices of the entities of the group make a continuous interval that does not overlap with the index 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 statements with blocking semantics.

*Event listener* is a call-back function to be dynamically registered that will be called whenever an *🡪 event reporter* calls f\_EPTF\_LGenBase\_dispatchEvent

*Reportable event* is a declared notification message of a given *🡪 behavior type* that can be distributed to *🡪 event listeners* by 🡪 *event reporters* via f\_EPTF\_LGenBase\_dispatchEvent

*Event reporter* is analtstep, or function of a given *🡪 behavior type* that eventually calls f\_EPTF\_LGenBase\_dispatchEvent

*Reported event* is an instance of a *🡪reportable event* supplied with reported arguments that is actually being distributed via f\_EPTF\_LGenBase\_dispatchEvent.

*Finite State Machine* is a formal description of *🡪 Control Logic*. It is a tabular description of what *🡪test steps* to call when a given *🡪reported event* occurs, when the entity is in a given declared 🡪 *state of the FSM*. After executing these *🡪 test steps* an optional next *🡪 FSM state* can be also defined.

*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.

*FSM state* is a locally indexed declaration of a state within an FSM table.

*FSM timer* is a locally indexed declaration of a timer within an FSM table.

*Traffic case type* is a prototype used as an input for instantiating traffic case instances on a specific entity group. It defines which *🡪 FSM table* to use, which *🡪 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 traffic case. An entity group may have more than one concurrent traffic cases.

*Group finish condition* is list of conditions that defines situations when a traffic case is over on group level, that is, no new traffic is launched on that group. Group finish conditions are evaluated whenever any entity of the group reports the event that it has just finished executing a traffic instance of that traffic case.

*Group finish actions* is list of action that is executed when any of the 🡪 group finish conditions has been met.

*User finish condition* is list of conditions that defines situations when a *🡪 traffic case* is over for a given entity, that is, the entity is not considered for selection for launching a new traffic, even if the entity returns to its idle *🡪 state*. User finish conditions are evaluated whenever any entity of the group reports finishing the traffic case, that is, returning to the “idle” state of the respective FSM context.

*User finish actions* is list of action that is executed when any of the *🡪 user finish* conditions has been met.

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

*Named functions* are functions that has an associated name (label), are declared to LGenBase metadata with that label; and can be invoked via their labels.

*External templates* are parameterized text modules are stored by LGenBase. Parameter values can be substituted dynamically during run-time.

*Template type* is a named declaration of a template assigned to a behavior type.

*Template set* is set of templates belonging to a traffic case. It is a list of <template type, external template> pairs.

## System Requirements

In order to use the LGenBase feature the system requirements listed in TitanSim CLL User Guide [4] should be fulfilled.

# LGenBase

## Overview

The EPTF CLL LGenBase component is a fundamental component providing an implementation for final state machine handling and traffic case management in a load test environment.

Using Applibs based on LGenBase the users can easily configure and manage traffic tests.

### Burst calculation

Burst calculation functions are used to adjust the rate, the load level to be achieved. For reaching the adequate rate we have to calculate the number of traffic/event to be started in a call generation event and the next call generation time. There are number of predefined burst calculating functions, but user can specify his own function too. The predefined burst functions are orig, imst, imst2, cs and poisson. The burst calculation methods then can be referred by their names or their indices.

Burst calculating functions can be assigned to a traffic case or a weighted scenario in three ways:

First it can set via the initialization of LGenBase component. In this case all of the traffic cases can be set with the same burst calculation method.

The second way for setting burst calculation is via the declaration of a traffic case or weighted scenario. The parameter *scheduler* can be used to assign a burst calculation method to a traffic case.

Users can use the function f\_EPTF\_LGenBase\_setBurstCalcAlgByIdx() or f\_EPTF\_LGenBase\_setBurstCalcAlgByName() to assign a burst calc method to a traffic case or a weighted scenario.

The original EPTF burst calculating function (orig) uses the following algorithm:

The trivial formula idealBurstSize=cpsToReach\*cpsUpdateInterval specifies the number new calls/traffics to be generated at every (cpsUpdateInterval)th second. Call generation has two main input parameters that affect the resulting CPS value: cpsToReach and cpsUpdateInterval. However, the cpsUpdateInterval is subject to rouding errors in the scheduler.

Therefore, when these values are set, the load generator calculates the parameter roundedCpsUpdateInterval as the nearest multiple of the elementary timestep greater than (or equal to) parameter cpsUpdateInterval. It calculates a burst size based on this rounded value and parameter cpsToReach.

However, when the call generation event is scheduled - relative to the snapshot of the actual component timer - the scheduler also rounds the scheduling time to the elementary timestep. Thus the scheduled interval is usually longer than needed (quantization error).

To prevent this issue, the actual interval used to schedule a call generation event is calculated by decreasing the rounded update interval with the elementary timestep multiplied by a float value greater than 0.0 and lower than 1.0. This float value gives the probability of accurate scheduling. In case of a multiplication factor of 0.99, 99% of scheduling will be accurate and the remaining 1% is scheduled later than needed.

The IMST type burst calculation function (imst) uses the following algorithm for calculating the number of traffic/event to be started and the next call generation time:

It wakes up every 10 millisec, this will be the next call generation time. At the start it sets masterCounter variable to 0. The variable moduloOperand equals 100\*1000/rate where 100 is OUR\_RESOLUTION, 1000 is TICS\_PER\_INTERVAL and rate is the rate to be achieved. Then it increases masterCounter variable by one. If masterCounter modulo moduloOperand equals 0 then one plus traffic should be started. It repeats the increase of mastercounter 1000 (TICS\_PER\_INTERVAL) times.

OUR\_RESOLUTION=100;

TICS\_PER\_INTERVAL=1000;

masterCounter=0;

moduloOperand=OUR\_RESOLUTION\*TICS\_PER\_INTERVAL/rate;

For(i=masterCounter;i< masterCounter+ TICS\_PER\_INTERVAL;i++)

{

If((i mod moduloOperand)==0);

{

trafficToStart=trafficToStart+1;

}

}

masterCounter=masterCounter+TICS\_PER\_INTERVAL;

The IMST2 type burst calculation function (imst2) does the same as the IMST type algorithm a little bit simplified and faster way.

The next call generation time is 10 millisec too. The variable moduloOperand is the same, it equals OUR\_RESOLUTION\*TICS\_PER\_INTERVAL/rate. The number of traffic should be started is the floor of (masterCounter+TICS\_PER\_INTERVAL) divided by moduloOperand. The new masterCounter equals (masterCounter+TICS\_PER\_INTERVAL) modulo moduloOperand.

OUR\_RESOLUTION=100;

TICS\_PER\_INTERVAL=1000;

masterCounter=0;

moduloOperand=OUR\_RESOLUTION\*TICS\_PER\_INTERVAL/rate;

trafficToStart=

int((masterCounter+TICS\_PER\_INTERVAL)/moduloOperand);

moduloOperand=

(masterCounter+TICS\_PER\_INTERVAL)mod moduloOperand;

### External templates

LGenBase supports loading and caching of parameterised text modules called external templates. LGenBase allows effective parameter substitution into the templates during runtime. LGenBase provides specific set of templates belonging to each traffic case. There are three different template parameter substitution modes:

* Dictionary based substitution
* Positional parameter based substitution
* Fixed position based substitution

## Description of files in this feature

The EPTF CLL LGenBase API includes the following files:

* LGenBase
  + EPTF\_CLL\_LGenBase\_Definitions.ttcn: This TTCN-3 module contains LGenBase type definitions that can be used in all components extending LGenBase.
  + EPTF\_CLL\_LGenBase\_ConfigDefinitions.ttcn: This TTCN-3 module contains common type definitions that can be used to describe or process test descriptions from configuration files without importing LGenBase-specific definitions.
  + EPTF\_CLL\_LGenBase\_PhaseDefinitions.ttcn: This TTCN-3 module contains phase related type definitions that can be used in all components extending LGenBase.
  + EPTF\_CLL\_LGenBase\_Functions.ttcn: This TTCN-3 module contains the implementation of LGenBase functions.
  + EPTF\_CLL\_LGenBase\_ConfigFunctions.ttcn contains functions to manipulate the test configurations, such as traffic cases, scenarios, FSM-s, etc.
  + EPTF\_CLL\_LGenBase\_LoggingDefinitions.ttcn This TTCN-3 module contains logging related type definitions that can be used in all components extending LGenBase.
  + EPTF\_CLL\_LGenBase\_LoggingFunctions.ttcn This TTCN-3 module contains the implementation of logging functions.
  + EPTF\_CLL\_LGenBase\_PhaseFunctions.ttcn contains functions to use phase related scenarios, manipulate phases, etc.
  + EPTF\_CLL\_LGenBase\_EventHandlingFunctions.ttcn contains functions of event dispatching.
  + EPTF\_CLL\_LGenBase\_EventHandlingExternalFunctions.cc contains external functions of event dispatching.
  + EPTF\_CLL\_LGenBase\_ExternalFunctions.ttcn contains the headers of the external functions of event dispatching.
  + EPTF\_CLL\_LGenBase\_StepFunctions.ttcn contains the functions in the test steps registered by the LGenBase.
  + EPTF\_CLL\_LGenBase\_TrafficFunctions.ttcn contains functions to manage traffic, e.g. start, stop traffic cases, scenarios, handle entity
  + EPTF\_CLL\_LGenBaseTrafficMixer\_Definitions.ttcn contains data types to describe traffic cases based on their weight factors.
  + EPTF\_CLL\_LGenBaseTrafficMixer\_Functions.ttcn contains functions to generate traffic cases based on their weight factors.
  + EPTF\_CLL\_LGenBase\_TemplateDefinitions.ttcn contains data types and constants need to handle external templates.
  + EPTF\_CLL\_LGenBase\_TemplateFunctions.ttcn contains functions for handling external templates.
* LGenBaseStats
  + EPTF\_CLL\_LGenBaseStats\_Definitions.ttcn: This TTCN-3 module contains common type definitions that should be used in all components extending LGenBaseStats.
  + EPTF\_CLL\_LGenBaseStats\_Functions.ttcn: This TTCN-3 module contains the implementation of LGenBaseStats functions.
* LGenBaseStatsUI
  + EPTF\_CLL\_LGenBaseStatsUI\_Definitions.ttcn: This TTCN-3 module contains common type definitions that should be used in all components extending LGenBaseStatsUI.
  + EPTF\_CLL\_LGenBaseStatsUI\_Functions.ttcn: This TTCN-3 module contains the implementation of LGenBaseStatsUI functions.

## Description of required features

The LGenBase feature is part of the EPTF Core Load Library (CLL). It relies on some features of the CLL. To use the LGenBase, the user has to obtain the respective files from the following features:

* Base
* Common
* FreeBusyQueue
* RandomNArray
* HashMap
* RedBlackTree
* Scheduler
* Variable
* StatHandler
* TCCUsefulFunctions

For the LGenBaseStats feature:

* Variable

For the LGenBaseStatsUI feature:

* UIHandler

## Installation

Since EPTF\_CLL\_LGenBase 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\_LGenBase:

* Copy the files of the features listed in Section ‎2.2 to the directory of the test suite or create symbolic links to them.
* If you use TITAN plugin for Eclipse, the files of the features must lay in the “CLL dir”\src\”feature” structure. You have to create a path variable with the name of CoreLibraryPath pointing to the “CLL dir” directory.
* Import the LGenBase demo or write your own application using LGenBase.
* 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].

This LGenBase 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.

This LGenBase feature defines the following module parameters:

**tsp\_EPTF\_LGenBaseDebug**

This boolean type module parameter is defined in module EPTF\_CLL\_LGenBase\_Functions. The feature will log the FSM event handling if the module parameter is true.

**tsp\_EPTF\_LGenBaseDebugTraffic**

This boolean type module parameter is defined in module EPTF\_CLL\_LGenBase\_Functions. The feature will log the traffic management if the module parameter is true.

**tsp\_EPTF\_LGenBase\_roundedCpsUpdateInterval**

This float type module parameter is defined in module EPTF\_CLL\_LGenBase\_Functions ‎[6]. The feature periodically generates traffic. This parameter defines this rate.

**tsp\_EPTF\_LGenBase\_abortStopProcess**

This float type module parameter is defined in module EPTF\_CLL\_LGenBase\_Functions ‎[6]. If the entities didn’t reply to c\_EPTF\_LGenBase\_inputName\_testMgmt\_stopTC or c\_EPTF\_LGenBase\_inputIdx\_testMgmt\_abortTC messages in time described by this module parameter, LGenBase steps to stopped or aborted state.

**tsp\_LGenBase\_nameSeparator**

This charstring type module parameter is defined in module EPTF\_CLL\_LGenBase\_Functions ‎[6]. This is used to concatenate the name of the entity group, the scenario and the traffic case in unique name of the traffic cases.

**tsp\_LGenBase\_extTemplLoadList**

This module parameter specifies the list of external template, which is loaded by initializing of the component. It shall be given a list which contains the exteranal templates’ name, file path and a flag to enable/disable automatic <LF> ─> <CR><LF> conversion.

{ {"foo\_templ", "/templates/foo.txt", false}, {"bar\_templ", "/templates/bar.txt", true} }

The default value of the module parameter is the empty list.

**tsp\_LGenBase\_templParamOpenToken**

This charstring type module parameter specifies the opening string of the template parameters. Empty string is not allowed to define. The default value of the module parameter is: ‘$(‘ .

**tsp\_LGenBase\_templParamCloseToken**

This charstring type module parameter specifies the closing string of the template parameters. Empty string is not allowed to define. The default value of the module parameter is: ‘)‘ .

**tsp\_LGenBase\_enableDefaultArithmeticOperationsInExtTemplate**

This module parameter enables the default arithmetic operations handling during external template parameter substitution. The addition, subtraction, multiplication and divison is handled on parameters if the substitution value has a valid number.

**tsp\_LGenBase\_extTemplRemoveLastNewLine**

This boolean type module parameter can be used to strip the last new line from an external template. Its default value is false.

**tsp\_EPTF\_LGenBaseDebugLightList**

The LGenBase activates the DebugLight debugging feature (see also [6]) on the FSM tables enlisted in this module parameter.

**tsp\_LGenBaseStatsUI\_columnDescriptorList**

This module parameter describes the contents of each traffic case row on the LGenBaseStatsUI GUI. The colTitle field describes the title of the specified column, the tcColId describes which data to be displayed in that column, and the enableIfAvailable describes whether the column should be writeable, if the data is writeable, or it should be read only in all cases.

**tsp\_LGenBaseStatsUI\_scColumnDescriptorList**

This module parameter describes the contents of the table in the row of the scenarios. Rather for future use.

**tsp\_EPTF\_LGenBase\_ILog\_enableILog**

This boolean type module parameter is defined in module EPTF\_CLL\_LGenBase\_ILog\_Definitions. It can be used to enable or disable the LGenBase’s ILog feature. Its default value is true, which means that the ILog feature is enabled by default.

## Usage

### LGenBase

To use the functionalities of the LGenBase feature

* extend the EPTF\_LGenBase\_CT with your component
* call the f\_EPTF\_LGenBase\_init function from the init function of your component
* use the public functions of the LGenBase

### LGenBaseStats

To use the functionalities of the LGenBaseStats feature

* extend the EPTF\_LGenBaseStats\_CT with your component
* call the f\_EPTF\_LGenBaseStats\_init function from the init function of your component
* use the variables provided by the LGenBaseStats to display and/or manage the properties of the LGenBase.

### LGenBaseStatsUI

Extend the EPTF\_LGenBaseStatsUI\_CT with your component.

The f\_EPTF\_LGenBaseStatsUI\_init function must be called before using the functionalities of the component.

Users should create the GUI which will contain the widgets created by the LGenBaseStatsUI.

When all the scenarios are ready, the f\_EPTF\_LGenBaseStatsUI\_prepareGUI function must be called. That’s all.

## DTE handling in LGenBase

DTE generated by test steps in FSM will be caught by LGenBase and the event "LGenBase: Dynamic\_test\_case\_error\_occured!" is reported for the entity. The FSM of the entity is deactivated, the entity is disabled. If the reported DTE event is not handled in the FSM, LGenBase reports a traffic error.

The error string of the DTE can be retrieved by the function f\_EPTF\_LGenBase\_get\_dte\_str (in integer pl\_eIdx, in integer pl\_fCtxIdx).

If DTE occurs, then all events are cancelled for the entity in the FSM, test steps that follow the step that caused the DTE are not executed, and the state remains the same as it was when the failing step was started.

LGenBase will not report the DTE event for failing test steps that are executed when the DTE event is processed in the FSM (that is the DTE handling part).

By default DTE handling is disabled.

# 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.

"Traffic case is running. The burst calculation method can not be set."   
The burst calculation method can not be set to a running traffic case.

"This burst calculation method cannot be found."   
The burst calculation method cannot be found in the database. Probable this method wasn’t declared before, or its name is misspelled.

"This burst calculation method is already defined."   
A burst calculation method has been declared with the same name.

Error: attempting to delete fcb. at invalid idx: [idx]"   
The index of the scenario creation callback to be deleted is invalid.

"Can not schedule traffic case timer event.”  
The f\_EPTF\_SchedulerComp\_scheduleAction function returned false, the EPTF\_Scheduler\_CT couldn’t schedule the timer to call the f\_EPTF\_LGenBase\_doTrafficCaseTimerAction.

"Invalid input index in finished event.”  
The private f\_EPTF\_LGenBase\_doTrafficCaseTimerAction function was called with invalid event parameter.

"\* must be \*”  
The specified object has been registered in wrong order. Probably other components had registered objects of the same type earlier, however the init function of the LGenBase must be called first.

"Unhandled burst management type.”  
Internal coding error, it must not happen.

"The behavior type already exists.”  
A behavior type with the same name had been registered already.

"Invalid behavior type: [name]”  
The behaviour type had not been registered.

"The step [name] already exists!”  
A test step with the same name had been registered already.

"The declared event is a timer timeout event, but the [name] name is not in the list of timers!”  
The event type in the FSM declarator is a timer timeout event, but the specified name is not in the list of timers of the FSM.

"The declared [name] event behavior name is not valid!”  
The behaviour type had not been registered.

"The declared [name] event input name is not valid!”  
The event had not been registered.

"The count of the declared states and the states in row#[nr.] of the FSM differ!”  
There are less or more FSM states than the count of the cell rows in the FSM’s specified table row.

"Invalid step function: [name]”  
The referred step function had not been registered.

"Invalid step function type: “  
There is a registered function with the same name, but it its type is not EPTF\_LGenBase\_TestStepFunction\_FT.

"Invalid step: “  
The referred test step had not been registered.

"In case of timer start step the contextArgs must be present.”  
The test step is a timer start, but the contextArgs is omit.

"The declared [name] name is not in the list of timers!”  
The referred timer name does not exist in the list of timers of the FSM.

"In case of timer cancel step the contextArgs must be present.”  
The test step is a timer cancel, but the contextArgs is omit.

"In case of timer cancel step the contextArgs must be a name of a timer.”  
The test step is a timer cancel, the contextArgs is present, but not the timerName field is chosen but the stepContextArgs.

"If the contextArgs is present, it must be an integer list. Row#[nr.]”  
The contextArgs is present, but not the stepContextArgs field is chosen but the timerName.

"Invalid next state calculation function: [name]”  
The referred next state calculation function had not been registered.

"Invalid next state calculation function type: [name]”  
There is a registered function with the same name, but it its type is not EPTF\_LGenBase\_NextStateCalc\_FT.

"Declared nextstate must be valid”  
The name of the referred state is not present in the list of states of the FSM table.

"The name of the entity type mustn't be empty.”  
The pl\_name parameter of the f\_EPTF\_LGenBase\_declareEntityType function is an empty string.

"The entity type [name] already exists.”  
An entity type with the same name had been registered already.

"Invalid behavior type: [name]”  
There is no registered behavior type with the specified name.

"The entity group name mustn't be empty.”  
The name field of the EPTF\_LGenBase\_EntityGrpDeclarator is an empty string.

"The entity type name mustn't be empty.”  
The eType field of the EPTF\_LGenBase\_EntityGrpDeclarator is an empty string.

"The entity group [name] already exists.”  
An entity group with the same name had been registered already.

"Invalid entity type [name]”  
There is no registered entity type with the specified name.

"Invalid count of entities: [count]”  
The specified count is less than zero.

"Invalid entity group name: [name]”  
There is no registered entity group with the specified name.

"Invalid entity group index: [index]”  
The index of the entity group is invalid.

"Invalid customFinishFunction: [name]”  
There is no registered function with the specified name, or its type is not EPTF\_LGenBase\_customFinishFunction\_FT.

"Invalid group finish action.”  
Internal error, it should not happen.

"Invalid entity finish action.”  
Internal error, it should not happen.

"Invalid customFinishCondition: [name]”  
There is no registered function with the specified name.

"Invalid customFinishCondition type: [name]”  
There is a registered function with the specified name, but its type is not EPTF\_LGenBase\_customFinishCondition\_FT.

"Invalid custom entity success function: [name]”  
There is no registered function with the specified name.

"Invalid custom entity success function type: [name]”  
There is a registered function with the specified name, but its type is not EPTF\_LGenBase\_evaluateSuccess4EntityAtEntityGroup2\_FT.

"Invalid FSM name: [name]”  
There is no registered FSM table with the specified name.

"The traffic case name already exists: [name]”  
A traffic case with the same name had been registered already.

"FMS name must not be empty.”  
The name of the FSM table must be specified and must not be an empty string.

"Invalid traffic case index: [index]”  
The specified index of the traffic case is invalid.

"Invalid traffic case name: [name]”  
There is no traffic case with the specified name.

"Invalid scenario type: [name]”  
There is an inconsistency error in the scenario type.

"Not weighted scenario!: [name]”  
A function related to a weighted scenario has been called, but the referred scenario is not a weighted scenario.

"Invalid scenario name: [name]”  
There is no scenario with the specified name in the specified entity group.

"Invalid scenario index: [name]”  
There is no scenario in the entity group with the specified index.

"Invalid scenario index within the group: [index]”  
There is no scenario in the entity group with the specified index.

"Error assigning the traffic case [name] to group [name]”  
There was an error when a traffic case has been created on the specified entity group.

"collecting traffic weights: TC[index] in weighted SC shall have a weight instead of cpsToReach!”  
There is a non weighted traffic case in a weighted scenario.

"initializing traffic mix for TC in weighted SC [name]: mixer has not accepted traffic mix”  
There was an error initializing the traffic mix.

"Error: attempting to delete fcb. at invalid idx:[index]”  
The index of the EPTF\_LGenBase\_scenarioCreated\_FT functions in the list of “scenario created” callbacks is invalid.

"CPS cannot set to weighted TC: [name]”  
Required call per sec rate can not be set to a weighted traffic case.

"Weight cannot set to normal TC: “  
Weight can be set to a weighted traffic case only.

"CPS cannot set to non-weighted scenario: [name]”  
Required call per sec rate can be set to a weighted scenario only.

"Weighted traffic cases can not be paused. Use disable instead.”  
Only non-weighted traffic cases can be paused.

"Entity index”  
Invalid entity index.

"FSM context index”  
Invalid FSM context index.

"FSM table index”  
Invalid FSM table index.

"referred FSM table index”  
Invalid FSM table index.

"Row index for FSM table[index]”  
Invalid row index in the FSM table.

"Column index for FSM table[index]”  
Invalid column index in the FSM table.

"FSM timer index”  
Invalid FSM timer index.

"Referred FSM context index”  
Invalid FSM context index.

"FSM timer duration”  
The defined duration is less than zero.

"Presence of target entity's reference”  
There was no target defined in an event record to be reported to an entity.

"Special behavior type in event is unknown”  
Invalid behavior type is used.

"special behaviorType must be known”  
Invalid behavior type is used.

"Behavior type idx for timeouts must be [index]”  
Invalid behavior type is used.

"assumes 3 integers in actionId[]”  
Internal error in a private function.

"expects refContext in TestStepArgs”  
The refContext field of the test step args is not present.

"expects exactly 1 refArgs in refContext in TestStepArgs”  
Starting an FSM timer the step argument must be an integer list with length 1.

"Event scheduling must be always successful”  
The scheduler couldn’t schedule the timer action.

"initial state of FSM context”  
There is no defined state in the FSM table with the specified index.

"FSM context management error”  
Invalid FSM context index or there is no FSM table index in the referred context.

"Unhandled listener type.”  
Internal error.

"Event/input index”  
Invalid input index for the referred event.

"Event target field must be omit”  
The target field of a generic event must be omit.

"The index [index] is invalid. Max: [count]”  
Removing an event listener with invalid index.

"Event target field must be present”  
The target field of an entity event must be set.

"Invalid listener. pl\_bIdx == [index] , pl\_iIdx == [index], pl\_eIdx == [index]”  
There was no registered listener listening to the specified event.

"Length of pl\_listenerArgs must be 3”  
Internal error.

"Table index referred the by FSM context is [index]”  
Invalid FSM table index. Internal error.

"Referred FSM table row index”  
Invalid FSM table row index. Internal error.

"state index referred the by FSM context”  
Invalid FSM state index. Internal error.

"Invalid timer”  
There is no timer with the specified parameters.

"Invalid timer name: [name]”  
There is no timer in the FSM table with the specified name.

"Burst PreCalc is empty.”  
Internal error.

"Different FBQ sizes.”  
Internal error.

"The default UIHandler component must not be null.”  
The UIHandler component must be a valid component reference.

"Entity group name [name] contains the separator [separator]”  
The name of the entity group must not contain the separator character defined in tsp\_LGenBase\_nameSeparator module parameter.

"Scenario name [name] contains the separator [separator]”  
The name of the scenario must not contain the separator character defined in tsp\_LGenBase\_nameSeparator module parameter.

"Invalid variable name: “  
There is no EPTF Var variable with the specified name.

"Couldn't store phase list... Possible name collision in phase list…"   
A phase declarator has already been defined with this name. Use another name for declaring a new one.

# 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.

"The running traffic case [name] can not be restored!"  
A running traffic case can not be restored.

"Abort the entities of the traffic case [name] has been timed out.”  
Not all the entities answered to the aborting message.

"Stop the entities of the traffic case [name] has been timed out.”  
Not all the entities answered to the stopping message.

"The traffic case is in invalid [state] state!”  
When the timeout period of stopping or aborting a traffic case expired, the traffic case wasn’t in stopping, aborting or finished state.

"Entity stop response of entity#[index] of traffic case [name] has arrived, but the TC is in [state] state.”  
When the specified entity answered to the stopping event, the traffic case wasn’t in stopping state.

"Entity abort response of entity#[index] of traffic case [name] has arrived, but the TC is in [state] state.”  
When the specified entity answered to the aborting event, the traffic case wasn’t in aborting state.

"Start launch called for tc [name] in state [state]”  
When the start launch command was called, the traffic case was in invalid state.

"Start launch called for the disabled tc [name]”  
When the start launch command was called, the traffic case was in disabled state.

"Start launch called for weighted scenario [name] in state [state]”  
When the start launch command was called, the weighted scenario was in invalid state.

"Start launch called for weighted scenario [name] in state [state]”  
When the start launch command was called, the weighted scenario was in disabled state.

"Pause called for tc [name] in state [state]”  
The paused command was called, but the traffic case wasn’t in running state.

"Weighted traffic cases can not be paused. Use disable instead.”  
A weighted traffic case can not be paused.

"Event reporting request to an inactive FSM! Entity == [index] fsmCtxIdx == [index]”  
An event was reported to an inactive FSM.

"start has been requested for a running FSM timer: [index] timer will be cancelled first!”  
The started FSM timer is already running.

"Cancelling not running timer:=[index]”  
There was a cancel request to a not running timer.

"StartStep called but entiy#[index] in traffic case [name], [index] is busy!”  
An entity start was called, but the entity is running or finished.

"Warning: Path of external template cannot be empty!"   
Empty string is given as template name.

"Warning: Name of external cannot be empty!"   
Empty string is given as template file path.

"Warning: Could not read file: [path]"  
Cannot access external template file on file system.

"Warning: Template set with index: [index] doesn't exist "  
Referred template set doesn’t exist.

"Warning: Template type declaration with index: [index] doesn't exist "  
Referred template type declaration doesn’t exist.

"Warning: External template with index: [index] doesn't exist "  
Referred external template doesn’t exist.

"Warning: External template with name: [name] doesn't exist "  
Referred external template doesn’t exist.

"Warning: Template set doesn’t have item with index: [index]"  
The referred template item doesn’t exist in template set.

"Warning: Template type declaration with key [behaviourType, templateName] doesn’t exist"  
The referred template type declaration doesn’t exist.

"Warning: Assignment not possible, mandatory parameter: [param] is missing from external template"  
Assign a template type to an external template did not succeed because of missing mandatory parameter(s).

"Warning: Assignment in template set already exists"  
Assign a template type to an external template did not succeed because another assignment of same template type declaration already exist in the template set.

"Warning: Actual phase is already running in this scenario"  
The phase to be started is already running. LGenBase cannot start or skip a running phase again.

"Warning: There is already a running phase in this scenario"  
Another phase is already running. LGenBase cannot start or skip a phase, if there is a running phase. Running phase should be stopped before.

"Warning: No phase is running in this scenario "  
If there is no running phase, it cannot be stopped.

# Examples

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

* EPTF\_LGenBaseDemo\_CallerDefinitions.ttcn
* EPTF\_LGenBaseDemo\_CallerFunctions.ttcn
* EPTF\_LGenBaseDemo\_Definitions.ttcn
* EPTF\_LGenBaseDemo\_Functions.ttcn
* EPTF\_LGenBaseDemo\_ResponderDefinitions.ttcn
* EPTF\_LGenBaseDemo\_ResponderFunctions.ttcn
* EPTF\_LGenBaseDemo\_SiblingFsmDefinitions.ttcn
* EPTF\_LGenBaseDemo\_SiblingFsmFunctions.ttcn
* EPTF\_LGenBaseDemo\_TemplateFunctions.ttcn
* EPTF\_LGenBaseDemo\_TransportDefinitions.ttcn
* EPTF\_LGenBaseDemo\_TransportFunctions.ttcn
* EPTF\_LGenBaseDemo\_msgReRegisterTemplate.txt
* EPTF\_LGenBaseDemo\_msgRegisterTemplate.txt

## Configuration file

The used configuration file (demo.cfg) is for the LGenBase example is placed in the demo directory.

## Demo Module

The demo module (GenFSM\_Demo.ttcn) illustrates a typical usage of the LGenBase feature and performs some tests.

The demo module (WeightedTraffic\_Demo.ttcn) illustrates some typical usage of weighted scenario.

# 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 113 200/5 Uen  
   User Guide for the TITAN TTCN-3 Test Executor
3. 155 17-CNL 113 512 Uen   
   EPTF CLL for TTCN-3 toolset with TITAN, Function Specification
4. 198 17-CNL 113 512 Uen  
   EPTF CLL for TTCN-3 toolset with TITAN, User Guide
5. 7/155 16-CNL 113 512  
   EPTF CLL LGenBase, Function Description
6. EPTF CLL for TTCN-3 toolset with TITAN, [Reference Guide](http://ttcn.ericsson.se/TCC_Releases/Libraries/EPTF_Core_Library_CNL113512/doc/apidoc/html/index.html)