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

EPTF CLL Logging, Function Description

Contents

[1 Introduction 3](#_Toc310943673)

[1.1 Revision history 3](#_Toc310943674)

[1.2 How to Read this Document 3](#_Toc310943675)

[1.3 References 3](#_Toc310943676)

[1.4 Scope 4](#_Toc310943677)

[1.5 Recommended way of reading 4](#_Toc310943678)

[1.6 Typographical conventions 4](#_Toc310943679)

[1.7 Abbreviations 4](#_Toc310943680)

[1.8 Terminology 4](#_Toc310943681)

[2 General Description 4](#_Toc310943682)

[3 Functional Interface 6](#_Toc310943683)

[3.1 Naming Conventions 6](#_Toc310943684)

[3.2 Format of the Logged Events 6](#_Toc310943685)

[3.3 Managing logging 6](#_Toc310943686)

[3.3.1 Initialization of Logging 6](#_Toc310943687)

[3.3.2 Register a log selection type 7](#_Toc310943688)

[3.3.3 Register a new logging Mask for a new component type 7](#_Toc310943689)

[3.3.4 Logging an Event 7](#_Toc310943690)

[3.3.5 Enable/disable EPTF Logging of all features on the current component 9](#_Toc310943691)

[3.3.6 Enable/disable EPTF Logging of the given feature on the current component 9](#_Toc310943692)

[3.3.7 Enable/disable an EPTF Logging event class of the given feature on the current component 9](#_Toc310943693)

[3.3.8 Check if user log is enabled for one of the given event classes 9](#_Toc310943694)

[3.3.9 Check if user log is enabled for the given event class 10](#_Toc310943695)

[3.3.10 Check if component log mask is enabled for the given event class 10](#_Toc310943696)

[3.3.11 Set an error message pattern to expect as the first error occurred 10](#_Toc310943697)

[3.3.12 Determine the number of errors 10](#_Toc310943698)

[3.3.13 To retrieve a certain error message 10](#_Toc310943699)

[3.3.14 To check if an error message matches with a given pattern 11](#_Toc310943700)

[3.4 Client/server functionality 11](#_Toc310943701)

[3.4.1 Initialization of LoggingServer 11](#_Toc310943702)

[3.4.2 Initialization of LoggingClient 11](#_Toc310943703)

[3.4.3 DataSource iterators and elements 12](#_Toc310943704)

[3.5 Obsolete client/server functionality using LoggingUI 12](#_Toc310943705)

[3.5.1 Initialization of LoggingUI 12](#_Toc310943706)

[3.5.2 Initialization of LoggingUIClient 12](#_Toc310943707)

[3.5.3 Enabling global logging 13](#_Toc310943708)

[3.5.4 Disabling global logging 13](#_Toc310943709)

[3.5.5 Enabling component type logging 13](#_Toc310943710)

[3.5.6 Disabling component type logging 13](#_Toc310943711)

[3.5.7 Enabling component type mask logging 13](#_Toc310943712)

[3.5.8 Disabling component type mask logging 14](#_Toc310943713)

[3.6 Summary Table of all public functions for EPTF Logging 14](#_Toc310943714)

[3.7 Summary Table of all public functions for EPTF LoggingUI 15](#_Toc310943715)

[3.8 Table of obsolete functions for EPTF Logging 15](#_Toc310943716)

# 

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-12-05 | PA1 | First draft version | EBENMOL |
| 2008-11-19 | PB1 | Added new functions, merged LoggingUI into Logging | EGBOTAT |
| 2009-12-11 | PC1 | Error message checking added | ETHJGI |
| 2011-08-08 | PD1 | Client/server functionality | ELSZSKU |
| 2011-11-11 | PE1 | Register a log selection type | ELSZSKU |

## How to Read this Document

This is the Function Description for the EPTF Logging 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>
6. 4/155 16-CNL 113 512 Uen  
   EPTF CLL Base, Function Description
7. 2/155 16-CNL 113 512 Uen  
   EPTF CLL UIHandler, Function Description

## Scope

This document is to specify the content and functionality of the EPTF Logging 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. Moreover, whenever a concept is mentioned that has a special meaning as described in the Glossary (Section 1.8) of this document, then these occurrences are marked with an initial arrow, e.g., 🡪 *TitanSim Core (Load) Library(CLL).*

## Abbreviations

CLL Core Load Library

EPTF Ericsson Load Test Framework, formerly TITAN Load Test Framework

MTC Main Test Component

PTC Parallel Test Component

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 EPTF Logging feature of the TitanSim CLL. The EPTF Logging feature consists of the following sub-features:

* Logging with component type EPTF\_Logging\_CT
* LoggingUI with component type EPTF\_LoggingUI\_CT
* LoggingUIClient with component type EPTF\_LoggingUIClient\_CT

Throughout this document, EPTF Logging generally refers to the basic logging component, i.e. the Logging sub-feature.

The EPTF **Logging** sub-feature makes it possible to

* Declare logging classes per components (e.g. separate log masks and classes for each component type in a component extend hierarchy),
* Provide controllable error, warning and debug logging trough provided log functions,
* Switch the logging on/off from TTCN per component type (only the current instance) or logging class,
* Switch the logging on/off from the UI if LoggingUI and LoggingUIClient are used. In this case, logging for component types can also be turned on or off globally or per instance,
* Automatically remove the debug logging from the compiled executable. This is performed via the constant folding of TITAN.
* Test if the expected error message pattern matches with the occurred errors. This can be used in negative testing.

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.

To be able to use EPTF Logging, the user component should extend the EPTF\_Logging\_CT component call its init function.

The default logging classes and the related functions are obsolete. The new functions have the suffix V2 in their name, and should be used instead of the old ones. The user should specify their own logging classes for the new functions. It is recommended to implement simple wrapper functions with the user defined class IDs for component types using EPTF Logging to make the feature easier to use.

The **LoggingUI** and **LoggingUIClient** sub-features make it possible to

* Put logging masks on the GUI
* Manage the change of logging masks
* Handle global logging masks

The aim of LoggingUI is to provide a user interface for the logging variables/masks defined in the EPTF Logging feature. With the help of this sub-feature, users can manage the logging of components and component types from the runtime GUI or CLI. It also makes possible to set the global logging masks of the components.

To be able to use EPTF LoggingUI, the user should extend one LoggingUI component and call its init function (typically this can be performed by the MTC). The components that will log should extend LoggingUIClient component and call its init function.

# 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 of the Logging, LoggingUI and LoggingUIClient sub-features have the prefix [f\_EPTF\_Logging\_](http://mwlx122.eth.ericsson.se:8080/EPTF_CORE_REFERENCE_GUIDE/R2/files/Logging/EPTF_CLL_Logging_Functions-ttcnpp.html#f_EPTF_Logging_enableLocal), f\_EPTF\_LoggingUI\_ and f\_EPTF\_LoggingUIClient respectively.

## Format of the Logged Events

In general, EPTF Logging logs the component type, the log class and the log message. A typical example:

“MyComponent”:“Debug”: “Initialization finished.”

## Managing logging

### Initialization of Logging

Before using the EPTF Logging functions the

f\_EPTF\_Logging \_init\_CT()

function should be called. This initializes the EPTF Logging feature.

If the logging functions are called before initialization of the Logging feature, the message parameter will be logged verbatim, i.e. the Feature will not log the component type and logging class of the log, the log will have typically the following format:

“Initialization started.”

Without initialization, the logging is always performed without checking any log masks.

### Register a log selection type

The logging classes are grouped by several categories. The topmost category is the selection type.

There are two predefined selection types, the EPTF\_CLL, and EPTF\_User types. The features of the Core Load Library register their logging classes obviously under the EPTF\_CLL type. Other features, such as application libraries register their logging classes under the EPTF\_User type. But users can register new selection types if they want to separate their logging classes. The f\_EPTF\_Logging\_registerSelection returns the index of the new selection. This index must be used in the following type and mask registration functions.

### Register a new logging Mask for a new component type

f\_EPTF\_Logging\_registerComponentMasks (pl\_componentTypeName, pl\_eventClassNames, pl\_selection)

Call this function to register the logging classes and component mask of your component type. The name of the component type (*pl\_componentTypeName)* and the new logging Mask (*pl\_eventClassNames*) should be specified. The latter can be an empty list { } if the component has no logging classes; in this case only the component local/global logging can be enabled or disabled and the log will not contain the logging class string, e.g.:

“MyComponent”: “Connecting to peer.”

The parameter pl\_selection is optional with default value EPTF\_Logging\_user. *This parameter must not be specified from user code.* EPTF features specify this parameter as EPTF\_Logging\_CLL to distinguish Core log from user log.

The integer returned by the function should be stored in a component variable and passed to the logging functions as a parameter.

The component logging mask and all logging class masks are created with enabled state.

### Logging an Event

EPTF Logging has multiple log functions that can be used to log an event, as described in subsections. All of these functions have the following format:

function f\_EPTF\_Logging\_...V2(  
 in charstring pl\_message,  
 in integer pl\_EPTF\_Logging\_maskId,  
 in EPTF\_IntegerList pl\_event\_classIdxList)  
runs on EPTF\_Logging\_CT

The parameters have the following meaning:

* pl\_message: the string to log
* pl\_EPTF\_Logging\_maskId: the component mask ID, as returned by f\_EPTF\_Logging\_registerComponentMask. The event will not be logged if logging was disabled for the component globally or locally.
* pl\_event\_classIdxList: a list of integer indexes. The indexes index elements in the list of strings that were passed to f\_EPTF\_Logging\_registerComponentMask as parameter pl\_eventClassNames. The event will be logged if logging for the component is enabled and at least one of the log classes referred to the class index list is enabled.

#### Logging an Error

The function f\_EPTF\_Logging\_errorV2 can be used to log an event with TITAN event type ERROR\_UNQUALIFIED.

Note that this function does not stop the execution. If the execution cannot continue because of the error, the user should call f\_EPTF\_Base\_stop (see [6]) after logging the error.

If the execution can still continue after logging, it should be considered to log the event as a warning instead 3.3.3.2).

#### Logging a Warning

The function f\_EPTF\_Logging\_warningV2 can be used to log an event with TITAN event type WARNING\_UNQUALIFIED.

#### Logging a Debug Message

The function f\_EPTF\_Logging\_debugV2 can be used to log an event with TITAN event type USER\_UNQUALIFIED. This function is not complied into the final executable if the flag –DNEDEBUG is added to CPPFLAGS\_TTCN3 and –O2 to CXXFLAGS in the Makefile.

Take care that if creating the log string pl\_message is in performance critical sections of the code, it should be guarded with the following statements:

if(c\_EPTF\_Common\_debugSwitch) {  
...  
}

Any functions called within the above `if` statement (or when creating the log string inline) must not have any side-effects.

#### Logging an Operational Event

The function f\_EPTF\_Logging\_operationalLogV2 can be used to log an event with TITAN event type USER\_UNQUALIFIED. Unlike the debug log described in 3.3.3.3, this log can not be optimized out from the compiled executable, but can still be disabled via the log classes or the component mask.

### Enable/disable EPTF Logging of all features on the current component

The functions

f\_EPTF\_Logging\_enableAllLocal() and f\_EPTF\_Logging\_disableAllLocal()

enables and disables EPTF Logging of all features on the current component.

### Enable/disable EPTF Logging of the given feature on the current component

The functions

f\_EPTF\_Logging\_enableLocal(pl\_compTypeId) and f\_EPTF\_Logging\_disableLocal(pl\_compTypeId)

enables and disables EPTF Logging of the given feature (*pl\_compTypeId*) on the current component.

### Enable/disable an EPTF Logging event class of the given feature on the current component

The functions

f\_EPTF\_Logging\_enableLocalMask(pl\_compTypeId, pl\_eventClass) and f\_EPTF\_Logging\_disableLocalMask(pl\_compTypeId, pl\_eventClass)

will enable and disable the given EPTF logging event class on this component.  The parameter *pl\_eventClass* is the index of the event class defined by <tsp\_EPTF\_UserEventClassPrefixList> eg.: Error, Warning...

### Check if user log is enabled for one of the given event classes

The function

f\_EPTF\_Logging\_isEnabledList(pl\_EPTF\_Logging\_maskId, pl\_event\_classIdxList)

will check if user log is enabled for one of the given event classes. The parameter *pl\_event\_classIdxList* specifies a list of event classes.

### Check if user log is enabled for the given event class

The function

f\_EPTF\_Logging\_isEnabled(pl\_EPTF\_Logging\_maskId, pl\_event\_classIdx)

will check if user log is enabled for the given event class specified by the parameter *pl\_event\_classIdx*.

### Check if component log mask is enabled for the given event class

The function

f\_EPTF\_Logging\_maskIsEnabled(pl\_EPTF\_Logging\_maskId, pl\_event\_classIdx)

will checks if the given component user-log mask is enabled for the given user event class.

### Set an error message pattern to expect as the first error occurred

The function

f\_EPTF\_Logging\_setExpectedErrorMsg(pl\_expectedError)

can be used to set an error pattern that will be check during cleanup automatically. If the first error message was the one that matches to this pattern, the verdict of the test case is set to pass automatically. If it does not match, then the verdict of the test case is set to fail.

An example for the expected error pattern that checks the end of the error message can be:

“\*Some error occurred”

This function can be used in negative testing.

### Determine the number of errors

The number of errors occurred is returned by the function

f\_EPTF\_Logging\_nofErrorMsgs()

### To retrieve a certain error message

A given error message text can be retrieved by the function

f\_EPTF\_Logging\_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\_Logging\_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.

## Client/server functionality

In order to manage centrally the logging on the distributed components, Logging provides client/server functionality with the EPTF\_LoggingServer\_CT and EPTF\_LoggingClient\_CT component types.

Since the supported use-case to manage runtime the logging is the use of GUI, both component types use the DataSource feature as their interface.

### Initialization of LoggingServer

First f\_EPTF\_LoggingServer\_init\_CT must be called.

The parameter pl\_selfName is the EPTF self name of the component instance.

The parameter pl\_sourceCompRef is the DataSource server component instance. If the parameter is not set, LoggingServer won't publish its data as DataSource client.

### Initialization of LoggingClient

Before calling any LoggingClient function, f\_EPTF\_LoggingClient\_init\_CT must be called.

The parameter pl\_selfName is the EPTF self name of the component instance.

The parameter pl\_loggingServer is a previously initialized LoggingServer component instance.

The parameter pl\_sourceCompRef is the DataSource server component instance. If the parameter is not set, LoggingClient won't publish its data as DataSource client.

In the f\_EPTF\_LoggingClient\_init\_CT the client component automatically connects to the server component.

### DataSource iterators and elements

For detailed list of iterators and external data elements see the ApiDoc of the Core Load Library.

## Obsolete client/server functionality using LoggingUI

### Initialization of LoggingUI

For using the EPTF LoggingUI functions the

f\_EPTF\_LoggingUI\_init\_CT(pl\_selfName, pl\_subscriber,pl\_tabboxName)

function should be called. This function initializes the main LoggingUI component. There always should be one main LoggingUI component. This component manages the global logging masks, and creates the user interface for Logging. LoggingUI automatically handles LoggingUIClient requests after initialization.

The parameter pl\_selfName is the EPTF self name of the component instance.

The UIHandler master component (see [7]) reference shall be passed to this function via parameter pl\_subscriber (this can be **self** in case the UIHander is on the same PTC as the LoggingUI or **mtc** if both of these component types are extended by the MTC).

A tabbox should be defined (*pl\_tabboxName*), where the LoggingUI layout will be created.

### Initialization of LoggingUIClient

For using the EPTF LoggingUIClient functions the

f\_EPTF\_LoggingUIClient\_init\_CT(pl\_selfName, pl\_loggingUI\_main, pl\_subscriber)

function should be called. This function initializes the LoggingUIClient component. Components using the graphical visualization of logging should extend LoggingUIClient.

Parameter pl\_selfName is the EPTF self name of the component instance. The component reference to the main LoggingUI component should be passed to this function via pl\_loggingUI, and the reference to the main UIHandler component (see [7]) via pl\_subscriber.

### Enabling global logging

For enabling global logging of all components

f\_EPTF\_LoggingUI\_enableAllGlobal()

function should be called. This function enables the logging on all components.

### Disabling global logging

For disabling global logging of all components

f\_EPTF\_LoggingUI\_disableAllGlobal()

function should be called. This function disables the logging on all components.

### Enabling component type logging

For enabling logging of a component type

f\_EPTF\_LoggingUI\_enableGlobal(pl\_compTypeId)

function should be called. This function enables the logging of a component type specified by its ID (*pl\_compTypeId*).

### Disabling component type logging

For disabling logging of a component type

f\_EPTF\_LoggingUI\_disableGlobal(pl\_compTypeId)

function should be called. This function disables the logging of a component type specified by its ID (*pl\_compTypeId*).

### Enabling component type mask logging

For enabling logging of a component type’s mask

f\_EPTF\_LoggingUI\_enableGlobalMask(pl\_compTypeId, pl\_eventClass)

function should be called. This function enables the logging of a component type mask. The component type is specified by its ID (*pl\_compTypeId*), the logged mask is specified by its maskID (*pl\_eventClass*).

### Disabling component type mask logging

For disabling logging of a component type’s mask

f\_EPTF\_LoggingUI\_disableGlobalMask(pl\_compTypeId, pl\_eventClass)

function should be called. This function disables the logging of a component type mask. The component type is specified by its ID (*pl\_compTypeId*), the logged mask is specified by its maskID (*pl\_eventClass*).

## Summary Table of all public functions for EPTF Logging

Table 1: Summary of public EPTF Logging functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_Logging \_init\_CT | Initializes the EPTF Logging feature |
| f\_EPTF\_Logging\_registerComponentMasks | Registers a new logging Mask for a new component type |
| f\_EPTF\_Logging\_errorV2 | Log an event with event type ERROR\_UNQUALIFIED. |
| f\_EPTF\_Logging\_warningV2 | Log an event with event type WARNING\_UNQUALIFIED. |
| f\_EPTF\_Logging\_debugV2 | Log an event with event type USER\_UNQUALIFIED. Can be optimized out. |
| f\_EPTF\_Logging\_operationalLogV2 | Log an event with event type ERROR\_UNQUALIFIED. |
| f\_EPTF\_Logging\_enableAllLocal | Enables EPTF Logging of all features on the current component |
| f\_EPTF\_Logging\_disableAllLocal | Disables EPTF Logging of all features on the current component |
| f\_EPTF\_Logging\_enableLocal | Enables EPTF Logging of the given feature on the current component |
| f\_EPTF\_Logging\_disableLocal | Disables EPTF Logging of the given feature on the current component |
| f\_EPTF\_Logging\_enableLocalMask | Enables the given EPTF logging event class on this component |
| f\_EPTF\_Logging\_disableLocalMask | Disables the given EPTF logging event class on this component |
| f\_EPTF\_Logging\_isEnabledList | Checks if user log is enabled for one of the given event classes |
| f\_EPTF\_Logging\_isEnabled | Checks if user log is enabled for the given event class |
| f\_EPTF\_Logging\_maskIsEnabled | Checks if the given component user-log mask is enabled for the given user event class |

## Summary Table of all public functions for EPTF LoggingUI

Table 2: Summary of LoggingUI functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_LoggingUI\_init\_CT | Function to initialize main LoggingUI component |
| f\_EPTF\_LoggingUIClient\_init\_CT | Function to initialize LoggingUI Client component |
| f\_EPTF\_LoggingUI\_enableAllGlobal | Function to enable EPTF Logging of all features on all components. |
| f\_EPTF\_LoggingUI\_disableAllGlobal | Function to disable EPTF Logging of all features on all components. |
| f\_EPTF\_LoggingUI\_enableGlobal | Function to enable EPTF Logging on the current component type |
| f\_EPTF\_LoggingUI\_disableGlobal | Function to disable EPTF Logging on the current component type |
| f\_EPTF\_LoggingUI\_enableGlobalMask | Function to enable an EPTF Logging an event class on the current component type |
| f\_EPTF\_LoggingUI\_disableGlobalMask | Function to disable an EPTF Logging an event class on the current component type |

## Table of obsolete functions for EPTF Logging

The following functions are kept for backward compatibility. Do not use these functions when developing new functionality.

Table 3: Summary of obsolete EPTF Logging functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_Logging \_log | Logs one charstring argument if one of the classes is enabled for the given feature |
| f\_EPTF\_log | Logs one charstring argument if one of the classes is enabled for the given feature by default logging class |
| f\_EPTF\_Logging\_error | Logging the default logging class called “Error” |
| f\_EPTF\_Logging\_warning | Logging the default logging class called “Warning” |
| f\_EPTF\_Logging\_debug | Logging the default logging class called “Debug” |
| f\_EPTF\_Logging\_debugM | Logging the default logging class called “DebugM” |
| f\_EPTF\_Logging\_debugV | Logging the default logging class called “DebugV” |
| f\_EPTF\_Logging\_debugLevelM | Logging default logging classes "Debug" and "DebugM” simultaneously |
| f\_EPTF\_Logging\_debugLevelV | Logging default logging classes "Debug", "DebugM” and "DebugV” simultaneously |