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

EPTF CLL SMacro, User Guide

Contents

[1 Introduction 2](#_Toc405379499)

[1.1 Revision History 2](#_Toc405379500)

[1.2 About this Document 2](#_Toc405379501)

[1.2.1 How to Read this Document 2](#_Toc405379502)

[1.2.2 Abbreviations 2](#_Toc405379503)

[1.2.3 Terminology 2](#_Toc405379504)

[1.3 System Requirements 2](#_Toc405379505)

[2 SMacro 3](#_Toc405379506)

[2.1 Overview 3](#_Toc405379507)

[2.2 Description of Files in this Feature 3](#_Toc405379508)

[2.3 Description of Required Files from Other CLL Features 3](#_Toc405379509)

[2.4 Description of Required External Features outside CLL 3](#_Toc405379510)

[2.5 Installation 4](#_Toc405379511)

[2.6 Configuration 4](#_Toc405379512)

[2.7 Usage 4](#_Toc405379513)

[2.8 How to Define, Redefine, or Undefine a String Macro 4](#_Toc405379514)

[2.9 How to Resolve a String Macro 5](#_Toc405379515)

[2.10 How to Calculate Macro Value with Custom Function 6](#_Toc405379516)

[2.11 How to Escape Characters in TTCN-3 Code 7](#_Toc405379517)

[2.11.1 How to Escape with log2str Function 7](#_Toc405379518)

[2.11.2 How to Escape String Templates Which Contains Macros as Parameter 7](#_Toc405379519)

[2.12 How to Calculate Mathematical Expressions 8](#_Toc405379520)

[2.12.1 How to Use Built-in EVAL Macro 8](#_Toc405379521)

[2.12.2 How to Use Automatic Evaluation (autoEVAL) 9](#_Toc405379522)

[3 Error Messages 9](#_Toc405379523)

[4 Warning Messages 9](#_Toc405379524)

[5 Examples 10](#_Toc405379525)

[6 References 10](#_Toc405379526)

# Introduction

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Rev** | **Characteristics** | **Prepared** |
| 2014-11-19 | PA1 | First draft version | EZOLZSI |
| 2014-12-02 | A | Updated for release | ESZILSZ |

## About this Document

### How to Read this Document

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

### Abbreviations

CLL Core Library

EPTF Ericsson Load Test Framework, formerly TITAN Load Test Framework

SMacro String Macro feature

TitanSim Ericsson Load Test Framework, formerly TITAN Load Test Framework

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

### Terminology

*TitanSim Core Library (CLL)* is that part of the TitanSim software that is totally project independent. (That is, which is not protocol-, or application-dependent). The EPTF CLL is to be supplied and supported by the TCC organization. Any EPTF CLL development is to be funded centrally by Ericsson

## System Requirements

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

# SMacro

## Overview

The SMacro feature is developed to replace the former external template handling mechanism in CLL LGenBase.

String Macros are necessary when someone needs to replace certain patterns in a text. These patterns are called String Macros and the string itself is termed as String Template.

Examples:

|  |  |
| --- | --- |
| String Template | “Username is: $SIGNUM” |
| String Macro | SIGNUM |
| Value of String Macro | “ethjgi” |
| Resolved string template | “Username is: ethjgi” |

## Description of Files in this Feature

The EPTF CLL SMacro feature includes the following files:

* EPTF\_CLL\_SMacro\_Definitions.ttcn: this TTCN-3 module contains common type definitions used by the SMacro Component
* EPTF\_CLL\_SMacro\_Functions.ttcn: this TTCN-3 module contains the implementation of SMacro functions
* EPTF\_CLL\_SMacro\_ExternalFunctions.cc: this TTCN-3 module contains external functions for SMacro

## Description of Required Files from Other CLL Features

The EPTF SMacro feature depends on the following CLL features:

* Base
* Common
* FreeBusyQueue
* HashMap
* Logging

## Description of Required External Features outside CLL

* TCCUsefulFunctions\_CNL113472 (TCCConversion\_Functions)

## Installation

Since EPTF\_CLL\_SMacro 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 noted otherwise in the respective sections, the following are needed to use EPTF\_CLL\_SMacro:

* Copy the files which belong to the CLL features listed in section 2.2 and 2.3 to the directory of the test suite or create symbolic links to them.
* Import the SMacro test or write your own application using EPTF SMacro.
* 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 through the run-time configuration file. This is a simple text file which contains various sections. The usual suffix of configuration files is .cfg. For further information on the configuration file, see ‎[2].

The EPTF SMacro feature does not define any module parameters.

## Usage

To use the EPTF SMacro feature, do the following:

* Extend your component with the EPTF\_SMacro\_CT component
* Call the init function (f\_EPTF\_SMacro\_init\_CT) of the SMacro to initialize the feature
* Use its public functions to handle String Macros or String Templates
* Call f\_EPTF\_Base\_cleanup\_CT function before your component terminates, or call f\_EPTF\_Base\_stop function if the execution should be stopped (for example, because of an error the execution is forced to stop immediately)

## How to Define, Redefine, or Undefine a String Macro

After the initialization you should define the needed String Macros by calling the f\_EPTF\_SMacro\_define function.

Example: defining the USER macro with value “ethjgi”

f\_EPTF\_SMacro\_define(“USER”, “ethjgi”);

var charstring my\_stringTemplate := “Username is: $USER”;

The value of the resolved String Template is:   
“Username is: ethjgi”;

It is possible to redefine macro value by calling the f\_EPTF\_SMacro\_define function again.

Example: re-defining the USER macro with value “ezolzsi”

f\_EPTF\_SMacro\_define(“USER”, “ezolzsi”);

my\_stringTemplate := “Username is: $USER”;

The value of the resolved String Template is:   
“Username is: ezolzsi”;

It is possible to undefine macro value by calling the f\_EPTF\_SMacro\_undefine function.

Example: Undefining the USER macro

f\_EPTF\_SMacro\_undefine(“USER”);

my\_stringTemplate := “Username is: $USER”;

The value of the resolved String Template is:   
“Username is: $USER”;

## How to Resolve a String Macro

A String Template which contains any number of String Macro can be resolved by calling the f\_EPTF\_SMacro\_resolve function.

Example: resolving the previously re-defined USER macro which has the value “ezolzsi”

var charstring my\_stringTemaplate := “Username is: $USER”;  
  
var charstring my\_resolvedStringTemplate := f\_EPTF\_SMacro\_resolve(my\_stringTemplate);

The value of my\_resolvedStringTemplate is:   
“Username is: ezolzsi”;

In the example above the f\_EPTF\_SMacro\_resolve function was called without automatic evaluation since the default value of parameter pl\_autoEVAL is false.

## How to Calculate Macro Value with Custom Function

It is possible to calculate the macro value with registered macro calculator function.

Example: defining the SUM macro to calculate its value by concatenating its two arguments and inserts + sign between them. First we have to define the macro to have calculator function name as its value.

f\_EPTF\_SMacro\_define(“SUM”, “f\_calc\_sum”);

The function that calculates the value of SUM macro is:

function f\_calc\_sum(  
 in EPTF\_CharstringList pl\_args,  
 in EPTF\_IntegerList pl\_userArgs := {}) {

return pl\_args[0]&”+”&pl\_args[1];

}

This macro calculator function needs to be registered with the name given in the define function.

Example: registering the macro calculator function with name “f\_calc\_sum”:

f\_EPTF\_SMacro\_registerCalcFn(  
 pl\_functionName := “f\_calc\_sum”,  
 pl\_macro\_function := refers(f\_calc\_sum)  
)

In this way when the f\_EPTF\_SMacro\_resolve function is called, the value of the SUM macro is calculated with f\_calc\_sum function which concatenates its two arguments and puts a “+” sign between them.

Example: resolving the String Template when the macro calculator function is registered:

var charstring my\_stringTemplate :=  
”How much is $(SUM, \“31”\, \“11”\)?”

var charstring my\_resolvedStringTemplate := f\_EPTF\_SMacro\_resolve(my\_stringTemplate);

The value of my\_resolvedStringTemplate is:   
“How much is 31+11?”

It is possible to deregister the macro calculator function by calling f\_EPTF\_SMacro\_deregisterCalcFn function:

f\_EPTF\_SMacro\_deregisterCalcFn(“f\_calc\_sum”);

Example: resolving the String Template when the macro calculator function is deregistered:

my\_resolvedStringTemplate := f\_EPTF\_SMacro\_resolve(my\_stringTemplate);

The value of my\_resolvedStringTemplate is:  
”How much is f\_calc\_sum?”

## How to Escape Characters in TTCN-3 Code

String Macro parameters are enclosed in quotation marks. Since String Template is a TTCN-3 charstring the quotation marks needs to be escaped in TTCN-3 code.

Example:

var charstring my\_stringTemplate :=

"MACRO1 value: $(MACRO1, \"parameter1\", \"parameter2\")

MACRO2 value: $(MACRO2, \"parameter3\")"

As it can be seen in the example above each quotation mark which encloses the macro parameters are escaped with “\” character.

### How to Escape with log2str Function

In order to create the same string template (see in the example above) without having unreadable code, the log2str built-in TITAN function has to be used.

Example:

var charstring my\_stringTemplate :=

"MACRO1 value: $(MACRO1, "&log2str("parameter1")&", "&log2str("parameter2")&")

MACRO2 value: $(MACRO2, "&log2str("parameter3")&")";

The log2str function does the escaping automatically on its charstring parameter. Its return value is concatenated with other the parts of the String Template.

### How to Escape String Templates Which Contains Macros as Parameter

It could happen that a macro which has parameters is a parameter of another macro. In that case the parameters need to be escaped in each level.

Example: MACRO2 will be the second parameter of MACRO1

var charstring my\_stringTemplate :=   
"My string template: $(MACRO1, \"parameter1\", \"$(MACRO2, \\\"parameter3\\\")\")"

As it can be seen in the example, each already escaped quotation mark is escaped two times again. Instead of doing this manually in TTCN-3 code, it is recommended to use the log2str built in TITAN function.

Example: same string template with log2str

var charstring my\_stringTemplate :=  
"My string template: $(MACRO1, "&log2str("parameter1")&","&log2str("$(MACRO2, "&log2str("parameter3")&")")&")"

In the next escape level it looks like:

Example: parameter3 of the previous example is MACRO2 again

var charstring my\_stringTemplate :=   
"My string template: $(MACRO1, \"parameter1\", \"$(MACRO2, \\\" $(MACRO2,\\\\\"parameter3\\\\\") \\\")\")"

As it can be seen, the parameter of MACRO2, which is the MACRO2 itself, is escaped again.

Example: previous string template with log2str

var charstring my\_stringTemplate :=  
"My string template with log2str: $(MACRO1, "&log2str("parameter1")&","&log2str("$(MACRO2, "&log2str("$(MACRO2, "&log2str("parameter3")&")")&")")&")"

It can be made more readable, if separate variables are defined for each String Macro parameter and those variables are used in the argument of the log2str function.

## How to Calculate Mathematical Expressions

If the String Template contains mathematical expressions which need to be evaluated you should use the built-in EVAL macro. Further details can be found in [6].

### How to Use Built-in EVAL Macro

EVAL macro is defined by the init function (f\_EPTF\_SMacro\_init\_CT), therefore there is no need to register it.

Example:

var charstring my\_stringTemplate := "Result of 2+4/2\*3-1 is: $(EVAL, \"2+4/2\*3-1\" )."

var charstring my\_resolvedStringTemplate := f\_EPTF\_SMacro\_resolve(my\_stringTemplate);

The value of my\_resolvedStringTemplate is:   
“Result of 2+4/2\*3-1 is: 7.”

The mathematical expression which is the first parameter of the EVAL macro was passed to the registered EVAL calculator function which did the calculation.

### How to Use Automatic Evaluation (autoEVAL)

After each String Macro is resolved in the String Template it is possible to call the built-in EVAL macro automatically. For this the pl\_autoEVAL parameter of the f\_EPTF\_SMacro\_resolve function should be set to true.

Example:

var charstring my\_stringTemaplate := "Result of $(SUM, \“31”\, \“11”\) is: 31+11"

var charstring my\_resolvedStringTemplate := f\_EPTF\_SMacro\_resolve(  
 pl\_stringTemaplate := my\_stringTemplate,  
 pl\_autoEVAL := true);

In the first round (in the background) the f\_EPTF\_SMacro\_resolve function only resolves the SUM macro, then the string template looks like the following:

“Result of 31+11 is: 31+11.”

Then the autoEVAL feature calculates both expressions and finally the value of my\_resolvedStringTemplate is:

“Result of 42 is: 42.”

# Error Messages

**Note:** besides the below described error messages, error messages shown in ‎[2] or those of other used features or product may also appear.

"(EVAL: Error: Division by zero.)"

It is not a classical error, since the execution is not stopped and there is not ERROR message in the logs. Only the above string is printed into the String Template as a resolved value. It occurs when the divisor is zero in case of division or remainder.

# Warning Messages

**Note:** besides the below described warning messages, warning messages shown in ‎[2] or those of other used features or product may also appear.

f\_EPTF\_SMacro\_define: Warning: macro name <macro name> is invalid, should contain only characters 'a..z', 'A..Z', '\_', '0..9'. Macro definition ignored.

If you try to define (f\_EPTF\_SMacro\_define) a macro with invalid name you will get this warning message and the macro will be ignored. List of the allowed characters can be found in [6].

# Examples

The “test” directory of the deliverable contains the following example:

EPTF\_SMacro\_Test\_Testcases.ttcn

# 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 113 200/4 Uen  
   User Guide for the TITAN TTCN-3 Test Executor
3. 109 21-CNL 113 512-20 Uen   
   EPTF CLL for TTCN-3 toolset with TITAN, Product Revision Information
4. 155 17-CNL 113 512 Uen   
   EPTF CLL for TTCN-3 toolset with TITAN, Function Specification
5. 198 17-CNL 113 512 Uen  
   EPTF CLL for TTCN-3 toolset with TITAN, User Guide
6. 36/155 16-CNL 113 512  
   EPTF CLL SMacro, Function Description
7. 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)