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

EPTF CLL Statistics Capture Control, User Guide

Contents

[Introduction 2](#_Toc422736843)

[1.1 Revision history 2](#_Toc422736844)

[1.2 About this Document 2](#_Toc422736845)

[1.2.1 How to Read this Document 2](#_Toc422736846)

[1.2.2 Abbreviations 3](#_Toc422736847)

[1.2.3 Terminology 3](#_Toc422736848)

[1.3 System Requirements 4](#_Toc422736849)

[1.4 Release Information 4](#_Toc422736850)

[2 Statistics Capture Control 4](#_Toc422736851)

[2.1 Overview 4](#_Toc422736852)

[2.2 Basic Usage 5](#_Toc422736853)

[2.2.1 Basic Configuration 5](#_Toc422736854)

[2.3 Extended Usage 5](#_Toc422736855)

[2.3.1 Capture Control Modes 5](#_Toc422736856)

[2.3.2 Extended Configuration 8](#_Toc422736857)

[2.4 Description of files in this feature 9](#_Toc422736858)

[2.5 Description of required files from other features 9](#_Toc422736859)

[2.6 Installation 10](#_Toc422736860)

[2.7 Configuration 11](#_Toc422736861)

[2.7.1 Module parameters of the EPTF\_CLL\_StatCapture 11](#_Toc422736862)

[2.7.2 Modules EPTF\_CLL\_StatCaptureControl 12](#_Toc422736863)

[3 Error messages 14](#_Toc422736864)

[4 Warning messages 15](#_Toc422736865)

[5 Examples 15](#_Toc422736866)

[5.1 Configuration file 16](#_Toc422736867)

[5.2 Demo Modules 17](#_Toc422736868)

[5.2.1 Basic Usage 17](#_Toc422736869)

[5.2.2 Extended Usage 18](#_Toc422736870)

[6 References 20](#_Toc422736871)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2007-11-16 | PA1 | First draft version | <EANDDAR> |
| 2007-12-06 | PA2 | Updated after revision | <EANDDAR> |
| 2008-01-02 | PA3 | The functionality is extended with a new (optional) component EPTF\_StatCapControl\_CT and belonging functionality | <EANDDAR> |
| 2010-08-24 | PB1 | Minor changes | <EJNOSVN> |
| 2010-10-25 | PC1 | HM83202 – separators in stat names substituted; stat header written after new stat(s) added | EGBOTAT |
| 2014-07-04 | PD1 | Updated section Module parameters of the EPTF\_CLL\_StatCapture | EDNISND |
| 2014-12-03 | D | Updated for release | ESZILSZ |
| 2015-05-20 | PE1 | Graphics replaced with new ones | ESZILSZ |
| 2015-06-19 | E | Updated for release and graphics changed back to the old ones. | ESZILSZ |

## About this Document

### How to Read this Document

This is the User Guide for the Statistics Capture Control of the Ericsson Performance Test Framework (TitanSim), Core Library (CLL). EPTF Core Library is developed for the TTCN-3 [1] Toolset with TITAN [2]. This document should be read together with the Function Description of the Statistics Capture Control feature [5] . For more information on the EPTF Core Library, see the Product Revision Information, the User Guide [6] and the Function Specification [5] of the TitanSim. Additionally, to understand the functionality of this feature, read the documentation of the Variable [7] and the Statistics Measure [8] features.

### Abbreviations

CLL Core 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

*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 EPTF Core Library is to be supplied and supported by the TCC organization. Any EPTF Core Library development is to be funded centrally by Ericsson

*Variables* contain values, from which *🡪 TitanSim Statistics* can be calculated automatically on changes or periodically.

*Statistics Measure* is a feature, which is responsible for creating *🡪 TitanSim Statistics* and updating their value when required.

*Statistics* are TTCN-3 variables. The values of such variables are automatically and periodically updated in the background.

*Capture Group* is a concept for grouping Statistics with the same *🡪 Sampling Time* together, capturing referenced variables at sampling intervals and printing contents of the group to the log files assigned to the group.

*Sampling Time* is defined per *🡪 Capture Group*, in seconds. It is the time period, at which the Statistics contained by a capture group are sampled and printed out to the log files.

*Capture Control Mode* is a functionality which allows choosing between several capture control modes for starting and stopping statistics capture, such as: *🡪 Manual*, *🡪 Load-based* and *🡪 Complete* *Capture Control* modes.

*Manual Capture Control* is a *🡪 Capture Control Mode that* starts/stops the statistics capture according to a manual user interaction (for example a button pressed on the TitanSim GUI).

*Load-based Capture Control* is a *🡪 Capture Control Mode*, which makes it possible to start/stop capture according to load stability measured in the system. The load Variable can be any user-defined load-related parameter supported by the TitanSim tool (for example the cps generated by the tool, CPU load of the IUT, etc.).

*Complete Capture Control* is a *🡪 Capture Control Mode*, which starts statistics capture automatically when the test execution is started and automatically stops it when the test execution is finished.

## System Requirements

In order to use the Statistics Capture Control feature the system requirements listed in EPTF Core Library User Guide [4] should be fulfilled.

Additionally, this feature also uses the functionalities of the TCC Useful Function product [7].

## Release Information

This document contains actualized information on the TitanSim Release R7A01. The new functionalities compared to the release R2A02 concerning the EPTF Statistics Capture Control feature are the following:

* The feature is extended with the client component. The existing component behaves like before and adds server functionalities.
* The changes are backward-compatible to the initial release R2A01 and tested according to this requirement.

# Statistics Capture Control

## Overview

The EPTF CLL Statistics Capture Control feature consists of two fundamental components providing an implementation for statistics capture control in a load test environment.

Statistics are calculated belonging to EPTF Variables ‎[9] by the EPTF Statistics Measure Feature [8]. For each EPTF Variable, several statistics can be defined containing different type of values: for example variable content, minimum, maximum, delta, etc. Statistics can be reset independently, without changing the original Variable value.

These Statistics are captured by the EPTF Statistics Capture Control feature. Statistics can be grouped to capture groups, defining a certain sampling period, when the capture group is printed to file. Each capture group can be printed to logfiles of a format defined by the user. The feature implements a mechanism to handle large logfiles for meeting the requirements of a typical load test use case.

## Basic Usage

The EPTF Statistics Capture Control feature is implemented by more components. The previously described functionality is implemented on the *EPTF\_StatCapture\_CT* component that provides the statistics capture functionality and offers an API for starting/stopping statistics capture. This component itself can be used as a stand-alone application for statistics capture and statistics capture control. The following figure (see ‎2.2.1) shows the test configuration. In this case the demo application is based on the EPTF\_StatCapture\_CT and the two belonging modules only (i.e. the optional modules are not needed).

### Basic Configuration

MTC

EPTF\_StatCapture\_demo\_CT

extends EPTF\_StatCapture\_CT

The *EPTF\_StatCapture\_demo\_CT* is the user application, it extends the *EPTF\_StatCapture\_CT* component. Statistics Capture is carried out by the functionality of the *EPTF\_StatCapture\_CT* component and it is controlled by function calls according to its API. That means, that the statistics capture can be started or stopped by calling the belonging API functions in the user application.

## Extended Usage

The second component implemented in the EPTF Statistics Capture Control feature is the *EPTF\_StatCaptureControl\_CT,* it extends the *EPTF\_StatCapture\_CT* component and it is an optional, add-on component, that provides extended functionality by implementing *automatic* statistics capture control based on external variables (see ‎2.3.2), for example defining certain capture start/stop criteria based on load stability in the system or user interaction.

### Capture Control Modes

There are three different Capture Control Modesoffered by the*EPTF\_StatCaptureControl\_CT*(they can be selected by the configuration file): Manual, Load-based and Complete Capture Control Mode.

#### **Manual** (or *User*) **Capture Control Mode**

If this capture control mode is selected by the user, statistics capture can be started and stopped any time (after the execution has been started) by user interaction (for example a button is pressed on the TitanSim GUI). An EPTF Variable of type boolean should be defined in the configuration file, which shows user interaction (value change from false->true initiates start capture). By default, this control mode is selected.

#### **Load-based** **Capture Control Mode**

If this capture control mode is chosen by the user, statistics capture is controlled according to load stability and user-specified timer intervals (module parameters:*tsp\_EPTF\_StatCaptureControl\_loadStableToStart*, *tsp\_EPTF\_StatCaptureControl\_loadStableCapturePeriod*and the *tsp\_EPTF\_StatCaptureControl\_loadCaptureDelay*in theconfig file, see Section ‎2.7.2).

For convenience reasons, in the further description the *\** abbreviation is used instead of the string “*tsp\_EPTF\_StatCaptureControl\_*”*.*

The Load-based mode continuously monitors an EPTF Variable which signals whether the load is stable in the system. Note that this Variable is a boolean variable (specified by the user with the *\*loadVariable* module parameter) and it can be any regulated load-related parameter supported by the TitanSim tool, for example cps generated by the tool, or CPU load of the IUT, etc.

The Statistics Capture is started after the load has been stable for the time specified by the user in the *\*loadStableToStart* module parameter. If the load remains stable for a time specified by the *\*loadStableCapturePeriod* module parameter, then the captured data during this period are considered valid and the capture of information is stopped.

After a successful capture, the user can define a delay to start a new capture by the *\*loadCaptureDelay* module parameter. Note that only a fixed delay interval can be specified via this parameter. By default, it is set to 0.0.

After the delay period is over, the *\*loadStableToStart* timer is started again to examine the load stability and to initiate a new capture.

Otherwise, if the load becomes unstable during the *\*loadStableCapturePeriod,* the capture is stopped andthe captured data so far during this capture data collection period becomes obsolete and have to be discarded (this is signalled in the capture file as well). In this case, the *\*loadStableToStart* timer is started again to initiate a new capture.

The maximum number of capture periods (i.e. the number of start capture-stop capture sequences) initiated (in Load-based mode) can be defined via the *\*nrOfCaptures* module parameter. This counter is increased independently of the success of capture periods. Once the maximum number is reached, no more capture periods can be initiated. The load generation is allowed to be stopped before the maximum number of capture periods is reached (see Figure).

The example scenario depicted in the figure shows the timers and the statistics capture control according to the load Variable.

***Test execution***

***capture***

t2

started

t1

t3

t1

t1

***load***

***timers***

t2

t2   
stopped

t2

stopped

started

stopped

started

Invalid

capture

Valid

capture

Figure: Example of Load-based Capture Control

stopped

Execution

stopped

t1 := *\*loadStableToStart*

t2 := *\*loadStableCapturePeriod*

t3 := *\*loadCaptureDelay*

*\*nrOfCapturePeriods := 5*

nrOfCapturePeriods = 3

Valid

capture

Execution

started

Figure: Example of Load-based Capture Control

#### **Complete** (or *Execution*) **Capture Control Mode**

If this capture control mode is selected, the capture of statistics is automatically started as soon as the test execution is started and automatically stopped when the test execution has been finished.

### Extended Configuration

Load\_CT

Variable

MTC

EPTF\_StatCaptureControl\_demo\_CT  
extends EPTF\_StatCaptureControl\_CT,

EPTF\_NS\_Client\_CT

Figure 1

User\_CT

Variable

Execution\_CT

Variable

EPTF Name Service (NS)

Server\_CT

In this configuration, the user application runs on the *EPTF\_StatCaptureControl\_demo\_CT,* which extends the *EPTF\_StatCaptureControl\_CT* component. This latter contains additional functionality to the *EPTF\_StatCapture\_CT* component (extends it), for example the automatic capture control functionality. Thus the starting or stopping of statistics can be bound to the fulfillment of certain conditions for values of Variables defined by the user, and the statistics capture can be controlled automatically.

The Capture Control Mode can be selected via the configuration file and the Variables belonging to the Capture Control Modes can be described here as well with the owner component name and their variable name. Note that all these Control Variables must be of type Boolean.

For example, the *Execution\_CT* component contains a boolean Variable which signals that the load test execution has been started or stopped. The *User\_CT* component contains a boolean Variable which shows whether the user has pressed the “Start Capture”/ “Stop Capture” button on the TitanSim GUI. The *Load\_CT* component contains a boolean Variable which shows whether load is stable or not. The demo application automatically controls statistics capture according to the selected Capture Control Mode and the values of these Variables.

For accessing these Variables in the user application, an *EPTF Name Service Server* must be running in the system, and the demo application must extend a belonging EPTF Name Service Client *EPTF\_NS\_Client\_CT* component.

## Description of files in this feature

The EPTF CLL Statistics Capture Control API includes the following files:

* StatCapture
  + EPTF\_CLL\_StatCapture\_Definitions.ttcnpp: This TTCN-3 module contains common type definitions that should be used in all Statistics Capture Controlling Components.
  + EPTF\_CLL\_StatCapture\_Functions.ttcnpp: This TTCN-3 module contains the implementation of Statistics Capture Control functions.
  + EPTF\_CLL\_StatCapture\_External.cc: This TTCN-3 module contains external functions for Statistics Capture Control.
* StatCaptureControl (extends StatCapture)
  + EPTF\_CLL\_StatCaptureControl\_Definitions.ttcnpp: This TTCN-3 module is optional, it contains common type definitions that should be used in all Statistics Capture Controlling Components if *automatic* statistics capture control is needed.
  + EPTF\_CLL\_StatCaptureControl\_Functions.ttcnpp: This TTCN-3 module is optional, it contains the implementation of Statistics Capture Control functions for *automatic* statistics capture control.

**Note:** the StatCaptureControl component and the belonging modules are optional. If the user application only needs programmable statistics capture control, these files are not necessary to include. This case the user application should extend the EPTF\_StatCapture\_CT component and include the files listed in the *StatCapture* bullet point.  
  
On the other hand, if automatic statistics capture control functionality is needed, the user application should include all files of the EPTF CLL Statistics Capture Control feature (both *StatCapture* and *StatCaptureControl* bullet points), and extend the EPTF\_StatCaptureControl\_CT component.

## Description of required files from other features

The Statistics Capture Control feature is part of the TitanSim EPTF Core Library (CLL). It relies on several features of the CLL and the TCC Useful Functions. The user has to obtain the products/files to be found under the respective feature names:

* Base
* Common
* FBQScheduler
* FreeBusyQueue
* HashMap
* Scheduler
* StatMeasure
* Variable
* NameService (for the optional StatCaptureControl)

From the TCC Useful Functions ‎[8]:

* + TCCFileIO.cc;
  + TCCFileIO\_Functions.ttcn;
  + TCCMaths.cc;
  + TCCMaths\_Functions.ttcn;
  + TCCMaths\_GenericTypes.ttcn;

## Installation

Since EPTF CLL Statistics Capture Control 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\_StatCapture:

* Copy the files listed in section [‎2.4, ‎2.5] to the directory of the test suite or create symbolic links to them.
* Import the Statistics Capture Control demo or write your own application using StatCapture.
* 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 [‎2.7].

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

### Module parameters of the EPTF\_CLL\_StatCapture

**tsp\_EPTF\_StatCapture\_def\_capture file**

This charstring type module parameter is defined in module EPTF\_CLL\_StatCapture\_Functions. It is used to define the name of the default capture file, where those capture groups, which are created without any assigned logfiles are dumped. The default value for this parameter is set to “DefaultFile\_<selfname>\_<component id>\_<date>\_<time>.txt” where selfname is the test component self name and the component id the component identifier.

**tsp\_EPTF\_StatCapture\_max\_file\_size**

This integer type module parameter is defined in module EPTF\_CLL\_StatCapture\_Functions. It is used to define the maximum file size of the logfiles, where the capture is dumped. By default, this file size is set to 1 GB (1000000000 is the default value). (Note that 2 GB can be a critical size for some OS).

**tsp\_EPTF\_StatCapture\_max\_nrof\_files**

This integer type module parameter is defined in module EPTF\_CLL\_StatCapture\_Functions. It defines the maximum number of files that can be opened for one data stream. That means, if a logfile is full, how many new files can be opened after to continue dumping. By default, this parameter is set to 100.

**tsp\_EPTF\_StatCapture\_fileFormat**

This enumerated type module parameter is defined in module EPTF\_CLL\_StatCapture\_Functions. It defines the output file format for the logfiles. Possible values are: *readable* or *gnuplot*. By default, this parameter is set to the *readable* format. Both formats are aimed to be post-processed easily. The *readable* format is easier to read and it is inherited from a previous project and the *gnuplot* format is aimed for use with Gnuplot.

**tsp\_EPTF\_StatCapture\_captureGroups**

This module parameter is of type EPTF\_StatCapture\_CaptureGroupsConfig, and it is defined in module EPTF\_CLL\_StatCapture\_Functions. With this module parameter it is possible to define Capture Groups (and belonging statistics) from the configuration file. In this case, to process this parameter, the function *f\_EPTF\_StatCapture\_parseConfigGroup()* should be called *after* the local Variables (for which Statistics are captured) are available/created/subscribed for on the user application component.   
By default, this parameter is set to an empty record.

**tsp\_EPTF\_StatCapture\_headerSeparator**

This module parameter specifies the separator character (or string) that the StatCapture is use when writing a statistics header line containing the name of the stats. The default value is a single space.

**tsp\_EPTF\_StatCapture\_statNameSeparatorSubstitute**

This module parameter specifies the character or string that is to be substituted instead of the separator character in statistics names when writing a header line. The default value is a single underscore (“\_”).

### Modules EPTF\_CLL\_StatCaptureControl

**tsp\_EPTF\_StatCaptureControl\_captureControlMode**

This enumerated type module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It defines the desired Capture Control Mode for automatic statistics capture control. Possible values are: *manual, loadBased, complete.* By default, the parameter is set to *manual* control mode*.*

**tsp\_EPTF\_StatCaptureControl\_nrOfCaptures**

This integer type module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It specifies the number of periods of starting-stopping capture, and its value is only considered if the *loadBased* Capture Control Mode has been selected. Its default value is set to 100.

**tsp\_EPTF\_StatCaptureControl\_loadStableToStart**

This float type module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It specifies the time interval, for which the load Variable must be stable in order to start capture. By default, it is set to 10.0 (seconds).

**tsp\_EPTF\_StatCaptureControl\_loadStableCapturePeriod**

This float type module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It specifies the time interval, for which statistics capture is carried out. By default, it is set to 600.0 (seconds).

**tsp\_EPTF\_StatCaptureControl\_loadCaptureDelay**

This float type module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It defines the time delay, which is started after a successful capture. By default, its value is set to 0.0 (seconds).

**tsp\_EPTF\_StatCaptureControl\_userVariable**

This module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It is an internal record type containing the following fields of type charstring: {componentName, varName}. In the *componentName* field the owner component`s name should be specified and to the varName field the name of the capture controlling EPTF Variable should be assigned.

E.g. tsp\_EPTF\_StatCaptureControl\_userVariable := {componentName := "User\_CT", varName := "UserVar"}.

**tsp\_EPTF\_StatCaptureControl\_loadVariable**

This module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It is an internal record type containing the following fields of type charstring: {componentName, varName}. In the *componentName* field the owner component`s name should be specified and to the varName field the name of the capture controlling EPTF Variable should be assigned.

E.g. tsp\_EPTF\_StatCaptureControl\_loadVariable := {componentName := "Load\_CT", varName := "LoadStableVar"}.

**tsp\_EPTF\_StatCaptureControl\_executionVariable**

This module parameter is defined in module EPTF\_CLL\_StatCaptureControl\_Functions. It is an internal record type containing the following fields of type charstring: {componentName, varName}. In the *componentName* field the owner component`s name should be specified and to the varName field the name of the capture controlling EPTF Variable should be assigned.

E.g. tsp\_EPTF\_StatCaptureControl\_executionVariable := {componentName := "Execution\_CT", varName := "ExecutionVar"}.

# 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.

**New StatCapture group name is not unique.**

The group name assigned to create a new capture group is already in use. Group names should be unique for each capture group.

**No output file defined for Capture Group.**

This error message is printed out, when the capture group is created with no logfile parameter assigned and also no default logfile is set in the config file.

**Group does not exist.**

This error message states that the input parameter of a function refers to a non-existent capture group.

**Statistics index inside the group does not exist**

This error message states that the input parameter of a function refers to a non-existing statistics inside a capture group.

**Cannot dump to file**

This error message occurs, when writing or appending to a file is not possible.

**Cannot open logfile**

This error message occurs, when opening a logfile is not possible.

**Cannot create Statistics, StatID already exists!**

This error message occurs, when the statistics ID to be created already exists.

**The same statistics already added to the group!**

This error message states that the statistics with the same statID is already added to the group.

**Statistics with periodic reset cannot be added to more capture groups.**

This error message occurs, when a statistics with periodic reset enabled is added to more than one capture group. This is not allowed as it would lead to side effects when resetting the statistics.

# Warning messages

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

**Group to be deleted does not exist.**

This warning message states that the input parameter of the group deletion function refers to a non-existent capture group.

**Group is not found for name <groupname>**

This warning message states that the input parameter (groupname) is not valid.

**No capture data defined for this capture group**

This warning message occurs, when the capture of an empty capture group should be started.

**Group does not exist.**

This warning message states that the input parameter of a function refers to a non-existent capture group. In this case, it is only a warning, as it does not have any harmful effect (for example when deleting a non-existent capture group).

**File size too low. Logstring cannot be dumped.**

This warning message states that the config file parameters regarding maximum file size and maximum number of files are set too low.

**The last file <file descriptor> is full, capture finished. Data is not logged to file.**

This warning message occurs, when dumping a capture stream has filled the maximum number of files (defined in the config file). From this timepoint, no more capture is possible for that stream.

# Examples

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

* main.cfg
* EPTF\_StatCapture\_demo.ttcn
* EPTF\_StatCaptureControl\_demo.ttcn

## Configuration file

The used configuration file (main.cfg) is for the Statistics Capture Control example is placed in the demo directory.

In case the tsp\_EPTF\_StatCapture\_def\_capture\_file module parameter is left empty, each capture group created with an empty log file list is captured to a file called “Default\_Capture\_File.txt”. By setting the module parameter to a filepath, the default capture file can be changed.

[MODULE\_PARAMETERS]

*//StatCapture config parameters*

tsp\_EPTF\_StatCapture\_def\_capture\_file := "DefaultFile"

tsp\_EPTF\_StatCapture\_max\_file\_size := 1000000 //1 MByte

tsp\_EPTF\_StatCapture\_max\_nrof\_files := 100 //maximum 100 files for each // stream

tsp\_EPTF\_StatCapture\_fileFormat := gnuplot //gnuplot/readable

tsp\_EPTF\_StatCapture\_captureGroups :=

{{groupName := "configgroup1",

timerPeriod := 1.0,

logFileNames := {"configFirstfile"},

statistics := { { varName := "var1",

statType := content,

statName := "content",

periodicReset := false,

activeStat := true

},

{ varName := "var1",

statType := delta,

statName := "delta",

periodicReset := false,

activeStat := true

}

}

},

{{groupName := "configgroup2",

timerPeriod := 1.0,

logFileNames := {"configSecondfile"},

statistics := { { varName := "var2",

statType := deltaSum,

statName := "deltaSumReset",

periodicReset := true,

activeStat := true

},

{ varName := "var2",

statType := EPS,

statName := "EPS",

periodicReset := false,

activeStat := true

}

}

}

}

*//StatCaptureControl config parameters*

tsp\_EPTF\_StatCaptureControl\_captureControlMode := loadBased; //manual/loadBased/complete

tsp\_EPTF\_StatCaptureControl\_nrOfCaptures := 4; // 4 periods of //start-stop capture

tsp\_EPTF\_StatCaptureControl\_loadStableToStart := 5.0;//the load //Variable must be stable for this long to start capture

tsp\_EPTF\_StatCaptureControl\_loadStableCapturePeriod:= 10.0; //10 //seconds long measurement

tsp\_EPTF\_StatCaptureControl\_loadCaptureDelay := 5.0; //5 //seconds long delay after successful capture

tsp\_EPTF\_StatCaptureControl\_userVariable := {componentName := "User", varName := "UserVar"};

tsp\_EPTF\_StatCaptureControl\_loadVariable := {componentName := "Load", varName := "LoadVar"};

tsp\_EPTF\_StatCaptureControl\_executionVariable := {componentName := "Execution", varName := "ExecutionVar"};

//StatCaptureControl Demo config parameters

tsp\_StatCaptureControl\_demo\_user\_modify\_timer := 20.0;

tsp\_StatCaptureControl\_demo\_load\_modify\_timer := 30.0;

tsp\_StatCaptureControl\_demo\_execution\_start\_timer := 1.0;

tsp\_StatCaptureControl\_demo\_execution\_stop\_timer := 149.0; //relative time from execution start

## Demo Modules

### Basic Usage

The demo module EPTF\_StatCapture\_demo.ttcn illustrates the basic usage of the EPTF Statistics Capture Control feature (see ‎2.2).

Two variables are created by the user (EPTF\_Var). For the first variable, two statistics are defined, one representing the content value, the second one representing the delta of that variable (EPTF\_StatMeasure). The second variable is a float variable, its deltaSum is measured in a third statistics.

**Note:** this statistics is set to periodicreset = true, which means its value is reset at sampling intervals of the capture group it belongs to.

var integer vl\_var1ID, vl\_var2ID:= -1;

var integer vl\_stat1ID, vl\_stat2ID, vl\_stat3ID := -1;

f\_EPTF\_Var\_newInt("var1",10,vl\_var1ID);

vl\_stat1ID := f\_EPTF\_StatMeasure\_newStat(vl\_var1ID, content);

var EPTF\_StatCapture\_tStatistics stat1 := f\_EPTF\_StatCapture\_createStatforCapture(vl\_stat1ID, "mystat1name");

vl\_stat2ID := f\_EPTF\_StatMeasure\_newStat(vl\_var1ID, delta);

var EPTF\_StatCapture\_tStatistics stat2 := f\_EPTF\_StatCapture\_createStatforCapture(vl\_stat2ID, "mystat2name");

f\_EPTF\_Var\_newFloat("var2",10.0,vl\_var2ID);

vl\_stat3ID := f\_EPTF\_StatMeasure\_newStat(vl\_var2ID, deltaSum, true);

var EPTF\_StatCapture\_tStatistics stat3 := f\_EPTF\_StatCapture\_createStatforCapture(vl\_stat3ID, "mystat3name");

The Statistics references are created for Capture. Then, they are added to Capture groups, for example:

f\_EPTF\_StatCapture\_addNewGroup("group2", 1.0,{fd2}, {stat3}, v\_groupidx2, group2statidxList);

The capture is started, and in the altstep statement, the underlying Variable values are modified during runtime. The Statistics values are refreshed automatically, and captured by the StatCapture component. Their values are printed to file at regular sampling intervals.

f\_EPTF\_StatCapture\_startCaptureAll();

…

alt {

…

[ ] t\_adjust.timeout{

f\_EPTF\_Var\_adjustContent(vl\_var1ID, {intVal:= vl\_now});

f\_EPTF\_Var\_adjustContent(vl\_var2ID, {floatVal:= int2float(vl\_now)}); vl\_now := vl\_now \* 2; t\_adjust.start; repeat;}

[ ] t\_wait.timeout {}

}

setverdict(pass);

f\_EPTF\_StatCapture\_stopCaptureAll();

f\_EPTF\_Base\_cleanup();

**Note**: at the end of the demo application, the Base cleanup function is called (not the StatCapture cleanup function!). As we have registered the StatCapture cleanup function to the Base component in our EPTF\_StatCapture\_init function with the command f\_EPTF\_Base\_RegisterCleanup(refers(f\_EPTF\_StatCaptureControl\_cleanup));, the Base component looks after the clean-up of all components in the demo system.

### Extended Usage

The demo module EPTF\_StatCaptureControl\_demo.ttcn shows the extended usage of the EPTF Statistics Capture Control feature (see ‎2.2) and illustrates the automatic capture control functionality with Capture Control Modes.

The demo testcase runs on the mtc\_test component, which first creates the components as described in the extendeD configuration section (see ‎2.3.2), and starts them with their behavior.

function f\_EPTF\_mtc\_test\_Behaviour\_1(in charstring pl\_selfName) runs on mtc\_test\_CT {

f\_init\_mtc\_test\_CT(pl\_selfName);

var NS\_test\_CT vl\_NS\_Server := NS\_test\_CT.create;

vl\_NS\_Server.start(f\_EPTF\_NS\_main\_CT("NameServiceServer"));

var User\_test\_CT vl\_User\_test := User\_test\_CT.create;

vl\_User\_test.start(f\_EPTF\_User\_test\_Behaviour\_1("User",

vl\_NS\_Server));

…

}

The StatCaptureControl component calls its init function, registers itself into the Name Service Server and creates Statistics, groups them into Capture Groups and defines belonging logfiles to the groups. Additionally, the values of Statistics are regularly updated for simulating a working system. Note that this behavior code is almost identical to the basic usage demo. The important difference is that in the init function of the StatCaptureControl\_test\_CT we call the *EPTF\_StatCaptureControl\_init* function instead of the *EPTF\_StatCapture\_init* function. This means, that we prepare for listening to the controls such as execution, load stability and user interaction.

function f\_init\_StatCaptureControl\_test\_CT(in charstring pl\_selfName, in EPTF\_NS\_CT pl\_NSCompRef) runs on StatCaptureControl\_test\_CT {

if (v\_StatCaptureControl\_test\_initialized) {

return; // already initialized

}

f\_EPTF\_StatCaptureControl\_init(pl\_selfName, pl\_NSCompRef);

v\_StatCaptureControl\_test\_initialized := true;

log("----StatCaptureControl test INIT DONE----");

}

In the meanwhile, the control components (User\_CT, Load\_CT, Execution\_CT) are also started. In the demo application, these components simulate a real system, where the User Variable is set according to a real button click event, etc. Therefore, with belonging configuration parameters, the timers for changing control Variable values on components are read in from the configuration file (for example when to start/stop test execution, when the load should become stable/unstable, when the user should press the start/stop button). These components read in the configuration file parameters and change control Variable values accordingly. They all call the Base cleanup function when terminated, except the Execution component; this one calls the *f\_EPTF\_Base\_stopRemote(mtc)* function with *mtc* as parameter. This means, if the test execution has finished, all components are stopped, which is the simulated and expected behavior of any TitanSim application with automatic Statistics Capture Control.

# 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 Core Library for TTCN-3 toolset with TITAN, Function Specification
4. 198 17-CNL 113 512 Uen  
   EPTF Core Library for TTCN-3 toolset with TITAN, User Guide
5. 15/155 16-CNL 113 512  
   EPTF Core Library Statistics Capture Control Function Description
6. EPTF Core Library for TTCN-3 toolset with TITAN, [Reference Guide](http://ttcn.ericsson.se/TCC_Releases/Libraries/EPTF_Core_Library_CNL113512/doc/apidoc/html/index.html)
7. 18/198 16-CNL 113 512 Uen  
   EPTF Core Library Variable User Guide
8. 16/198 16-CNL 113 512 Uen  
   EPTF Core Library Statistics Measure User Guide