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

EPTF Core Library StatHandler, Function Description

Contents

[1 Introduction 3](#_Toc485031401)

[1.1 Revision history 3](#_Toc485031402)

[1.2 How to Read this Document 3](#_Toc485031403)

[1.3 References 3](#_Toc485031404)

[1.4 Scope 4](#_Toc485031405)

[1.5 Recommended way of reading 4](#_Toc485031406)

[1.6 Typographical conventions 4](#_Toc485031407)

[1.7 Abbreviations 4](#_Toc485031408)

[1.8 Terminology 4](#_Toc485031409)

[2 General Description 5](#_Toc485031410)

[2.1 StatHandlerClient 5](#_Toc485031411)

[2.2 StatHandler Master Component 6](#_Toc485031412)

[3 Functional Interface 7](#_Toc485031413)

[3.1 Naming Conventions 8](#_Toc485031414)

[3.2 Public Functions of StatHandler 8](#_Toc485031415)

[3.2.1 Initialization 8](#_Toc485031416)

[3.2.2 Registering Stat-declared Callback Function 8](#_Toc485031417)

[3.2.3 De-registering a Stat-declared Callback 8](#_Toc485031418)

[3.2.4 Getting the Id/Index of a Stat 8](#_Toc485031419)

[3.2.5 Getting the Number of Statistics 8](#_Toc485031420)

[3.2.6 Getting the Name of a Stat 9](#_Toc485031421)

[3.2.7 Getting the Method of a Stat 9](#_Toc485031422)

[3.2.8 Getting the Stat Variable Index 9](#_Toc485031423)

[3.2.9 Getting the Aux Statistics 9](#_Toc485031424)

[3.2.10 Getting the Aux Variables 9](#_Toc485031425)

[3.2.11 Getting the Number of Data Sources 9](#_Toc485031426)

[3.2.12 Getting the Variables of a Data Source 10](#_Toc485031427)

[3.3 Public Functions of StatHandlerClient 10](#_Toc485031428)

[3.3.1 Initializing 10](#_Toc485031429)

[3.3.2 Declaring Statistics 10](#_Toc485031430)

[3.3.3 Registering a Data Source 11](#_Toc485031431)

[3.3.4 Registering a StatMeasure Statistic 13](#_Toc485031432)

[3.3.5 Registering an Aggregated Statistic 13](#_Toc485031433)

[3.3.6 Deregistering a Data Source 14](#_Toc485031434)

[3.3.7 Getting the Variable Name of a Stat 14](#_Toc485031435)

[3.3.8 Getting Auxiliary Variables of a Stat 14](#_Toc485031436)

[3.3.9 Registering a Stat Reset Callback 14](#_Toc485031437)

[3.3.10 Deregistering a Stat Reset Callback 15](#_Toc485031438)

[3.3.11 Resetting statistics 15](#_Toc485031439)

[3.4 Public UI-handling Functions 15](#_Toc485031440)

[3.4.1 Initializing for StatHandlerUI 15](#_Toc485031441)

[3.4.2 Initializing for StatHandlerClientUI 16](#_Toc485031442)

[3.4.3 Binding a Stat to a Widget 16](#_Toc485031443)

[3.5 Summary Table of all public functions for EPTF StatHandler 17](#_Toc485031444)

[3.6 Customization of the GUI 19](#_Toc485031445)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2009-02-16 | PA1 | First draft version | EGBOTAT |
| 2009-02-26 | PA2 | Modifications after review | EGBOTAT |
| 2009-03-05 | PA3 | Clarifications for standard deviation | EGBOTAT |
| 2009-03-20 | PA4 | Changes made to statistics deregistration | EGBOTAT |
| 2009-04-22 | PA5 | Added support for registering StatMeasure and aggregated statistics | EGBOTAT |
| 2009-06-02 | PA6 | PercentileP statistics added | EBENMOL |
| 2011-07-20 | PB1 | DataSource added | EJNOSVN |
| 2016-10-17 | PC1 | getAuxVariables added. | EIMRENA |
| 2016-10-17 | PC2 | Editorial changes | EIMERNA |
| 2016-12-02 | C | Updated for release | EIMRENA |
| 2017-02-23 | PD1 | Added resetting statistics. | EIMRENA |
| 2017-06-09 | D | Updated for release | EIMRENA |

## How to Read this Document

This is the Function Description for the StatHandler of the Ericsson Performance Test Framework (TitanSim), Core Library (CLL). TitanSim Core Library is developed for the TTCN-3 [1] Toolset with TITAN [2]. For more information on the TitanSim Core Library 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 113 200/6 Uen  
   User Guide for the TITAN TTCN-3 Test Executor
3. 109 21-CNL 113 512-25 Uen   
   EPTF Core Library for TTCN-3 toolset with TITAN, Product Revision Information
4. 155 17-CNL 113 512 Uen   
   EPTF Core Library for TTCN-3 toolset with TITAN, Function Specification
5. 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)
6. 18/155 16-CNL 113 512 Uen  
   EPTF CLL Variable, Function Description
7. 16/155 16-CNL 113 512 Uen  
   EPTF CLL Statistics Measure, Function Description
8. 2/155 16-CNL 113 512 Uen  
   EPTF\_CLL\_UIHandler, Function Description

## Scope

This document is to specify the content and functionality of the StatHandler feature.

## Recommended way of reading

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

## Typographical conventions

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

## Abbreviations

API Application Programming Interface

CLL Core Library

EPTF Ericsson Performance Test Framework

GUI Graphics User Interface

TitanSim New synonym for the EPTF 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 Core Library is to be supplied and supported by the TCC organization. Any Core Library development is to be funded centrally by Ericsson.

*Statistics* are temporal statistics of EPTF Variables measured by the StatMeasure feature. The value of a statistics is calculated from a single EPTF Variable, as it changes throughout the time of execution.

*Aggregated Stat* EPTF Variables calculated from other EPTF Variables (data sources). The word stat or statistics refers aggregated statistics throughout this document.

*Auxiliary Statistics A*ggregated statistics that are not declared by the user of StatHandler directly, but as a result of declaring another stat. That statistics use the calculated value of the auxiliary stat in its calculations.

*Auxiliary Variable* EPTF Variable used by aggregated statistics to store some data.

*Provider Variable* Variable that affects the value of a given EPTF Variable.

# General Description

This document specifies the StatHandler feature of the TitanSim Core Library.

StatHandler feature makes it possible to collect global, aggregated statistics via the existing EPTF Variable (see [6]) interface.

The StatHandler feature consists of master and client components. Users of the StatHandler feature can declare aggregated statistics with certain statistics methods (e.g. Sum), and register local or third party data sources (EPTF Variables). The client component serves as a control interface for the master component.

## StatHandlerClient

The primary purpose of the StatHandlerClient component is to provide a remote or local control interface for the StatHandler master component type.

The StatHandlerClient can control a default StatHandler master component, or multiple masters. If there is only a default master, its component reference shall be passed to the init function of the StatHandlerClient and the default parameter used in the controlling functions. Otherwise, the component reference of the StatHandler master shall be passed to these functions.

The data aggregation for statistics is performed using the EPTF Variable feature. The data source component provides the Variables to which the StatHandler subscribes when it receives a data source registration message from the controlling StatHandlerClient.

The StatHandlerClient can instruct the StatHandler master component to subscribe to a Variable that is either on the same component as the client, or on a different component (e.g. legacy code). The latter is called third party registration.

In case of third party registration, the controlling client is still responsible for the reset of the statistics/variable on the data source component. This can be done, for example, by setting a callback for the statistics reset event and adjusting the EPTF Variables of the data sources from within that callback function.

## StatHandler Master Component

The StatHandler master component performs the aggregation of statistics. Declared stats are stored in an internal database, the EPTF Variable storing the aggregated value is created upon declaration of the stat.

The master component subscribes to provider variables of the data sources registered by the StatHandlerClient. Whenever provider variables change, the related statistic value is calculated.

Although the StatHandler master component is configured via the client interface, it provides some public API functions for extensibility (e.g. GUI handling, initialization of statistics capture, etc.)

The StatHandler supports the following statistics methods and statistics types:

Min calculates the minimum of all the sources. The sources shall consist of a single variable with the same type as the reset value of the statistics: integer or float.

Max calculates the maximum of all the source variables. Supports integer and float data types.

Sum calculates the sum of all the data sources with type integer or float.

Mean calculates the global mean value of the sources. The supported statistics type is float, the sources shall consist of a local mean and a number-of-samples variable.

StandardDev calculates the standard deviation of the sources. The statistics type is float, the sources shall consist of a local mean, a number-of-samples and the local variable S that is part of the calculation of standard deviation performed by StatMeasure. This statistics also calculates and stores the mean (m or ) of the sources as auxiliary variable, with name <*statName>.mean*. The resulting value of the statistics is the following function:  
  
The variance (V or ) can be calculated from this value by dividing it with the number of samples (N). The standard deviation () is the square root of the variance.

GlobalAverage is the average of the data sources. Statistics type is float, sources shall consist of a sum and an element number variable.

Density is the piecewise sum of the distribution bin values of measured local density statistics. The boundaries of each data sources must be the same. The statistics type is EPTF\_IntegerList, data sources shall consist of local density and boundaries variables.

Percentile95 calculates the global 95 percentile statistics from locally measured stats. It declares two aux stats density and max, with name *<statName>.density* and *<statName>.max* respectively. Boundaries are stored in the auxiliary EPTF\_FloatList variable *<statName>.boundaries*. The type of the statistics is float, data sources shall consist of an EPTF\_IntegerList density, an EPTF\_FloatList boundaries and a float max variable.

PercentileP calculates the global pth percentile statistics from locally measured stats. It declares two aux stats density and max, with name *<statName>.density* and *<statName>.max* respectively. Boundaries are stored in the auxiliary EPTF\_FloatList variable *<statName>.boundaries*. The p value is stored in an auxiliary variable *<statName>.pValue*. The type of the statistics is float, data sources shall consist of an EPTF\_IntegerList density, an EPTF\_FloatList boundaries and a float max variable. The p value can be between 0.0 and 1.0. The statistics with value 0.5 is called median.

# Functional Interface

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

## Naming Conventions

All functions of StatHandler have the prefix f\_EPTF\_StatHandler\_ e.g. f\_EPTF\_StatHandler\_init\_CT. All functions of the StatHandlerClient have the prefix f\_EPTF\_StatHandlerClient\_ e.g. f\_EPTF\_StatHandlerClient\_declareStat.

## Public Functions of StatHandler

### Initialization

The function

f\_EPTF\_StatHandler\_init\_CT(pl\_selfName)

must be called before using the StatHandler. The EPTF self-name of the component must be supplied as an input argument.

### Registering Stat-declared Callback Function

The function

f\_EPTF\_StatHandler\_registerStatDeclaredCallbackFn(pl\_fn)

can be used to register a function reference as callback for the event when a statistics has been declared.

### De-registering a Stat-declared Callback

Statistics declared callback functions can be de-registered using the function

f\_EPTF\_StatHandler\_deregisterStatDeclaredCallbackFn(pl\_fn)

### Getting the Id/Index of a Stat

The function

f\_EPTF\_StatHandler\_getId(pl\_statName)

can be used to get the index of the statistics with name pl\_statName. The function returns the integer index or -1 in case of error, i.e. if the statistics does not exist.

### Getting the Number of Statistics

The function

f\_EPTF\_StatHandler\_getNofStats()

can be used to get the number of statistics declared.

### Getting the Name of a Stat

The function

f\_EPTF\_StatHandler\_getName(pl\_statIdx)

can be used to retrieve the name of a statistics by index.

### Getting the Method of a Stat

The function

f\_EPTF\_StatHandler\_getMethod(pl\_statIdx)

can be used to get the statistics method of an aggregated stat by its index.

### Getting the Stat Variable Index

f\_EPTF\_StatHandler\_getVarIdx(pl\_statIdx)

can be used to get the EPTF Variable index of a statistics.

### Getting the Aux Statistics

The function

f\_EPTF\_StatHandler\_getAuxStats(pl\_statIdx)

can be used to get a list of indices referring the auxiliary statistics of a stat. If the stat has no auxiliary statistics, the function returns an empty integer list.

### Getting the Aux Variables

The function

f\_EPTF\_StatHandler\_getAuxVars(pl\_statIdx)

can be used to retrieve the indices of auxiliary variables used by a statistics.

### Getting the Number of Data Sources

The function

f\_EPTF\_StatHandler\_getNofSources(pl\_statIdx)

can be used to get the number of data sources registered to a statistics.

### Getting the Variables of a Data Source

The function

f\_EPTF\_StatHandler\_getVarsOfSource(pl\_statIdx, pl\_sourceIdx)

can be used to get the indices of variables provided by a data source for a statistics.

## Public Functions of StatHandlerClient

### Initializing

The function

f\_EPTF\_StatHandlerClient\_init\_CT(pl\_selfName, [pl\_statHandler\_compRef])

must be called before using the StatHandlerClient. The parameter pl\_selfName is the EPTF self-name of the component, pl\_statHandler\_compRef is an optional component reference to the default StatHandler. If the latter parameter is not specified or is null, the StatHandlerClient will not have a default StatHandler master component, and the actual master component reference *must* be passed to the functions where it is possible.

### Declaring Statistics

The function

f\_EPTF\_StatHandlerClient\_declareStat(pl\_statName, pl\_statMethod, pl\_statResetValue, [pl\_statHandler])

can be used to declare an aggregated statistics. The statistics will be declared on the StatHandler master component referred by the parameter pl\_statHandler, or, if this parameter is omitted or null, on the default StatHandler of the client specified at initialization. The parameter pl\_statName specifies the name of the statistics to be declared, pl\_statMethod the method of the statistics aggregation, and pl\_statResetValue the reset value of the stat in EPTF\_Var\_DirectContent format. The reset value also specifies the type of the statistics, i.e. the type of variable that will contain the aggregated value.

Table 1 lists the statistics method and the statistics types supported by StatHandler (see 2.2 for details).

Table 1: list of supported statistics methods

|  |  |
| --- | --- |
| **Stat method** | **Supported stat type** |
| Min | integer, float |
| Max | integer, float |
| Sum | integer, float |
| Mean | float |
| StandardDev | float |
| GlobalAverage | float |
| Density | EPTF\_IntegerList |
| Percentile95 | float |
| PercentileP | float |

### Registering a Data Source

The following function can be used to register a data source to a statistics:

f\_EPTF\_StatHandlerClient\_registerStat( pl\_providerVarList, pl\_statName, [pl\_subscriptionMode], [pl\_wait4response], [pl\_sourceCompRef], [pl\_statHandler])

This is the generic register function of the StatHandlerClient, specific functions are listed later in this section.

The function has the following parameters:

* pl\_providerVarList: list of provider variable names.
* pl\_statName: name of the statistics for which the provider variables will be used as data source.
* pl\_subscriptionMode: EPTF Variable subscription mode for the provider variables. Optional parameter, default value: sampledAtSync.
* pl\_wait4response: boolean parameter specifying whether the function should use a semaphore to block the execution until the result of the operation is received. Optional parameter, default value: true.
* pl\_sourceCompRef: EPTF\_Var\_CT component reference of the provider variables. A value of **null** will be interpreted as **self**, i.e. this StatHandlerClient component. Optional parameter, default value: null.
* pl\_statHandler: The StatHandler master component reference that should subscribe to the provider variables. If this parameter is null, the default StatHandler component reference will be used. Optional parameter, default value: null.

The StatHandlerClient provides register functions specific to supported statistics types and methods. These functions accept a pre-defined number of Variable names as parameters instead of a list of names and also check the type of these variables with an assert operation. The following functions are recommended to use instead of the generic function:

* f\_EPTF\_StatHandlerClient\_registerStat\_IntMin:  
  register a data source for an integer statistics with aggregation method “Min”. This function needs a single variable name, the type of the variable must be integer.
* f\_EPTF\_StatHandlerClient\_registerStat\_FloatMin:  
  register a data source for a float statistics with aggregation method “Min”. This function needs a single variable name, the type of the variable must be float.
* f\_EPTF\_StatHandlerClient\_registerStat\_IntMax:  
  register a data source for an integer statistics with aggregation method “Max”. This function needs a single variable name, the type of the variable must be integer.
* f\_EPTF\_StatHandlerClient\_registerStat\_FloatMax:  
  register a data source for a float statistics with aggregation method “Max”. This function needs a single variable name, the type of the variable must be float.
* f\_EPTF\_StatHandlerClient\_registerStat\_IntSum:  
  register a data source for an integer statistics with aggregation method “Sum”. This function needs a single variable name, the type of the variable must be integer.
* f\_EPTF\_StatHandlerClient\_registerStat\_FloatSum:  
  register a data source for a float statistics with aggregation method “Sum”. This function needs a single variable name, the type of the variable must be float.
* f\_EPTF\_StatHandlerClient\_registerStat\_Mean:  
  register a data source for a float statistics with aggregation method “Mean”. This function needs two variable names as parameters: a float local mean and an integer number-of-samples.
* f\_EPTF\_StatHandlerClient\_registerStat\_StandardDev:  
  register a data source for a float statistics with aggregation method “StandardDev”. This function needs three variable names as parameters: a float local mean, an integer number-of-samples and a float variable ‘S’ that is calculated for a local standard deviation statistics.

f\_EPTF\_StatHandlerClient\_registerStat\_GlobalAverage:  
register a data source for a float statistics with aggregation method “GlobalAverage”. This function needs two variable names as parameters: a float sum and an integer element number.

* f\_EPTF\_StatHandlerClient\_registerStat\_Density:  
  register a data source for an integer-list statistics with aggregation method “Density”. This function needs two variable names as parameters: an integer-list local density and a float-list boundaries variable.
* f\_EPTF\_StatHandlerClient\_registerStat\_Percentile95:  
  register a data source for a float statistics with aggregation method “Percentile95”. This function needs three variable names as parameters: an integer-list local density, a float-list boundaries variable and a maximum variable.
* f\_EPTF\_StatHandlerClient\_registerStat\_PercentileP:  
  register a data source for a float statistics with aggregation method “PercentileP”. This function needs four variable names as parameters: an integer-list local density, a float-list boundaries variable a maximum variable and a pValue variable.

### Registering a StatMeasure Statistic

The following function can be used to register a StatMeasure stat as data source to an aggregated statistics:

f\_EPTF\_StatHandlerClient\_registerStatOfStatMeasure( pl\_measuredStatIdx, pl\_statName, [pl\_subscriptionMode], [pl\_wait4response], [pl\_statHandler])

Note: this function cannot be used for third party registration of the data source, as that must be on the same component as the StatHandlerClient.

### Registering an Aggregated Statistic

The following function can be used to register an aggregated statistics (i.e. StatHandler stat) as a data source to another stat:

f\_EPTF\_StatHandlerClient\_registerAggregatedStat( pl\_sourceStatHandler, pl\_sourceStatName, pl\_targetStatName, [pl\_subscriptionMode], [pl\_wait4response], [pl\_statHandler])

This function queries the provider variables of the source statistics, waits for the response regardless of the value of pl\_wait4response and registers these variables as the data source. The parameter pl\_wait4response is only for the response of the registration.

### Deregistering a Data Source

The following function can be used to deregister a data source from an aggregated statistics:

f\_EPTF\_StatHandlerClient\_deregisterStat(pl\_statName, [pl\_sourceCompRef], [pl\_statHandler], [pl\_providerVarList])

The parameter pl\_statName specifies the name of the aggregated statistic, pl\_sourceCompRef specifies the component providing the variables as data source, pl\_statHandler refers the StatHandler master component and pl\_providerVarList refers to the list of provider variable names as was specified at registration. The last three parameters are optional. If pl\_sourceCompRef is omitted, null or self, the self component reference will be used. If pl\_statHandler is omitted or is null, the default StatHandler master component will be used.

If parameter pl\_providerVarList is not specified or an empty list, the function deregisters all data sources provided by the source component for the given statistics.

### Getting the Variable Name of a Stat

The following function can be used to get the name of the EPTF Variable containing the aggregated value of a statistics:

f\_EPTF\_StatHandlerClient\_getStatVariableRef(pl\_statName, [pl\_statHandler])

The parameter pl\_statHandler is optional.

### Getting Auxiliary Variables of a Stat

The following function can be used to get the names of the auxiliary EPTF Variables for given statistics of the given StatHandler:

f\_EPTF\_StatHandlerClient\_getAuxVariables(in charstring pl\_statName, in EPTF\_StatHandler\_CT pl\_statHandler := null)  
runs on EPTF\_StatHandlerClient\_CT  
returns EPTF\_CharstringList

The parameter pl\_statHandler is optional.

### Registering a Stat Reset Callback

The following function can be used to register a function as a callback for statistics reset:

f\_EPTF\_StatHandlerClient\_registerResetFunction(pl\_fn)

The provided function will be called upon a statistics reset, and shall perform the reset for all data sources that were registered by this client.

### Deregistering a Stat Reset Callback

The following function can be used to deregister a reset callback function that was previously registered:

f\_EPTF\_StatHandlerClient\_deregisterResetFunction(pl\_fn)

### Resetting statistics

The following function can be used to reset given statistics on a StatHandler component:

function f\_EPTF\_StatHandlerClient\_resetStatistics(in charstring pl\_statName, in EPTF\_StatHandler\_CT pl\_statHandler := null, in boolean pl\_wait4response := true) runs on EPTF\_StatHandlerClient\_CT return boolean

The parameter pl\_statHandler is optional.

The parameters of the function:

* pl\_statName: the name of the statistics to be reset.
* pl\_statHandler: The statistics will be reset on this StatHandler component. If this parameter is *null*, the default StatHandler component reference will be used. Optional parameter, default value: *null*.
* pl\_wait4response: boolean parameter specifying whether the function should use a semaphore to block the execution until the result of the operation is received. Optional parameter, default value: *true*.

When the statistics is reset, its value is set to the reset value that was given at the declaration.

## Public UI-handling Functions

The module EPTF\_CLL\_StatHandlerUI\_Functions contains functions for handling the user interface part for both StatHandler and StatHandlerClient. This section lists the public functions contained within this module.

### Initializing for StatHandlerUI

The function

f\_EPTF\_StatHandlerUI\_init\_CT(pl\_selfName, pl\_uiParentWidgetId, pl\_uiHandler\_compRef, [pl\_loggingUi\_compRef], [pl\_localResetButtonUIVarName])

can be used to initialize a StatHandlerUI component, i.e. a StatHandler providing UI-handling capabilities.

This function has the following parameters:

* pl\_selfName: EPTF self-name of the component.
* pl\_uiParentWidgetId: the widget ID of the parent widget to which the StatHandler should add its tab and the global reset button.
* pl\_uiHandler\_compRef: component reference of the UIHandler component.
* pl\_loggingUi\_compRef: component reference of the LoggingUI component. Optional, default value is **null** for no LoggingUI.
* pl\_localResetButtonUIVarName: an optional variable name for reset button of this instance of StatHandler. If this is specified (and not empty string), the StatHandlerUI will attempt to subscribe to the variable (supposed to be at the UIHandler referred by pl\_uiHandler\_compRef) and use it as a reset button. The variable should be of integer type. If the variable just simulates a button, each time it is “pressed”, its value shall be incremented.

### Initializing for StatHandlerClientUI

The function

f\_EPTF\_StatHandlerClientUI\_init\_CT(pl\_selfName, pl\_statHandler\_compRef, pl\_uiHandler\_compRef, [pl\_loggingUi\_compRef])

can be used to initialize a StatHandlerClientUI component, i.e. a StatHandlerClient providing UI-handling capabilities.

### Binding a Stat to a Widget

The following client function can be used to instruct a StatHandler to bind its statistics variable to a widget on the GUI:

f\_EPTF\_StatHandlerClient\_bindStatToWidget(pl\_statName, pl\_widgetId, [pl\_statHandler\_compRef], [pl\_wait4response], [pl\_additionalWidget])

This function has the following parameters:

* pl\_statName: name of the statistics
* pl\_widgetId: name of the widget
* pl\_statHandler\_compRef: optional StatHandler master component reference
* pl\_wait4response: whether the client should block execution until the master component serves the request
* pl\_additionalWidget: whether the widget is additional, i.e. the statistics is already on another widget.

## Summary Table of all public functions for EPTF StatHandler

Table 2 lists the public API of the EPTF CLL StatHandler feature.

Table 2 Summary of StatHandler functions

|  |  |
| --- | --- |
| Function name | Description |
| f\_EPTF\_StatHandler\_init\_CT | Initialization function for StatHandler master component |
| f\_EPTF\_StatHandler\_registerStatDeclaredCallbackFn | Function to register a callback function for stat declaration event |
| f\_EPTF\_StatHandler\_deregisterStatDeclaredCallbackFn | Function to deregister a stat declaration callback function |
| f\_EPTF\_StatHandler\_getId | Get the ID/index of a stat by name |
| f\_EPTF\_StatHandler\_getNofStats | Get the number of declared statistics |
| f\_EPTF\_StatHandler\_getName | Get the name of a stat by its index |
| f\_EPTF\_StatHandler\_getMethod | Get the aggregation method of a statistics |
| f\_EPTF\_StatHandler\_getVarIdx | Get the EPTF Variable index of a stat |
| f\_EPTF\_StatHandler\_getAuxStats | Get auxiliary statistics of a stat |
| f\_EPTF\_StatHandler\_getAuxVars | Get auxiliary variables of a stat |
| f\_EPTF\_StatHandler\_getNofSources | Get the number of data sources registered for a stat |
| f\_EPTF\_StatHandler\_getVarsOfSource | Get the list of provider variables for a data source of a stat |
| f\_EPTF\_StatHandlerClient\_init\_CT | Initialization function for StatHandlerClient |
| f\_EPTF\_StatHandlerClient\_declareStat | Declare an aggregated stat |
| f\_EPTF\_StatHandlerClient\_registerStat | Register a data source for a stat |
| f\_EPTF\_StatHandlerClient\_registerStatOfStatMeasure | Register a StatMeasure statistic data source for a stat |
| f\_EPTF\_StatHandlerClient\_registerAggregatedStat | Register an aggregated stat as data source to another stat |
| f\_EPTF\_StatHandlerClient\_registerStat\_IntMin | Register a data source for an integer stat with Min method |
| f\_EPTF\_StatHandlerClient\_registerStat\_FloatMin | Register a data source for a float stat with Min method |
| f\_EPTF\_StatHandlerClient\_registerStat\_IntMax | Register a data source for an integer stat with Max method |
| f\_EPTF\_StatHandlerClient\_registerStat\_FloatMax | Register a data source for a float stat with Max method |
| f\_EPTF\_StatHandlerClient\_registerStat\_IntSum | Register a data source for an integer stat with Sum method |
| f\_EPTF\_StatHandlerClient\_registerStat\_FloatSum | Register a data source for a float stat with Sum method |
| f\_EPTF\_StatHandlerClient\_registerStat\_Mean | Register a data source for a Mean stat |
| f\_EPTF\_StatHandlerClient\_registerStat\_StandardDev | Register a data source for a Standard Deviation stat |
| f\_EPTF\_StatHandlerClient\_registerStat\_GlobalAverage | Register a data source for a Global Average stat |
| f\_EPTF\_StatHandlerClient\_registerStat\_Density | Register a data source for a Density stat |
| f\_EPTF\_StatHandlerClient\_registerStat\_Percentile95 | Register a data source for a Percentile 95 stat |
| f\_EPTF\_StatHandlerClient\_registerStat\_PercentileP | Register a data source for a Percentile P stat |
| f\_EPTF\_StatHandlerClient\_deregisterStat | Deregister a data source for a stat |
| f\_EPTF\_StatHandlerClient\_getStatVariableRef | Get the EPTF Variable name of an aggregated stat |
| f\_EPTF\_StatHandlerClient\_getAuxVariables | Get names of the auxiliary variables of a stat |
| f\_EPTF\_StatHandlerClient\_registerResetFunction | Register a statistics reset callback function |
| f\_EPTF\_StatHandlerClient\_deregisterResetFunction | Deregister a statistics reset callback function |
| f\_EPTF\_StatHandlerClient\_resetStatistics | Resets the statistics |
| f\_EPTF\_StatHandlerUI\_init\_CT | Initialize a StatHandlerUI component |
| f\_EPTF\_StatHandlerClientUI\_init\_CT | Initialize a StatHandlerClientUI component |
| f\_EPTF\_StatHandlerClient\_bindStatToWidget | Bind an aggregated stat to a widget on the GUI |

## Customization of the GUI

The StatHandler feature provides DataSource Client functionality. This means that a StatHandler component has several Variables, that are available to use with the help of a DataSource server, for example the UIHandler. With the call of the function f\_EPTF\_UIHandler\_createGUI(), a custom GUI is built, and the data of the features are freely to use. For more info about the DataSource feature, see the UIHandler Function Description [8]. For the list of the existing iterators and external data elements see the natural documentation.