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

Guideline to use the Browser-based GUI

Contents

[1 Introduction 2](#_Toc468717934)

[1.1 Revision history 2](#_Toc468717935)

[1.2 About this Document 2](#_Toc468717936)

[1.2.1 How to Read this Document 2](#_Toc468717937)

[1.2.2 Knowledge prerequisites 2](#_Toc468717938)

[1.2.3 Abbreviations 2](#_Toc468717939)

[1.2.4 Terminology 2](#_Toc468717940)

[1.3 System Requirements 3](#_Toc468717941)

[2 BrowserGUI 3](#_Toc468717942)

[2.1 How does it work 3](#_Toc468717943)

[2.2 Existing sample solution 4](#_Toc468717944)

[2.2.1 EPTF\_LoadMain.xsl 4](#_Toc468717945)

[2.2.2 main.css 4](#_Toc468717946)

[2.2.3 Javascript files 5](#_Toc468717947)

[2.2.4 External javascript libraries 5](#_Toc468717948)

[2.2.5 htmlbrowser.js 6](#_Toc468717949)

[2.2.6 Image files 6](#_Toc468717950)

[2.2.7 Logo 7](#_Toc468717951)

[2.2.8 Background images 7](#_Toc468717952)

[2.2.9 Loading image 7](#_Toc468717953)

[2.2.10 Led 7](#_Toc468717954)

[2.3 Customization 7](#_Toc468717955)

[3 Module parameters 8](#_Toc468717956)

[3.1 tsp\_EPTF\_UIHandler\_enableBrowserGUI 8](#_Toc468717957)

[3.2 tsp\_EPTF\_UIHandler\_Browser\_RemoteAddress 8](#_Toc468717958)

[3.3 tsp\_EPTF\_UIHandler\_Browser\_RemotePort 8](#_Toc468717959)

[3.4 tsp\_EPTF\_UIHandler\_Browser\_xsl2send 8](#_Toc468717960)

[3.5 tsp\_EPTF\_UIHandler\_Browser\_xsl4Tabpage 8](#_Toc468717961)

[3.6 tsp\_EPTF\_UIHandler\_Browser\_welcomeScreen 8](#_Toc468717962)

[3.7 tsp\_EPTF\_UIHandler\_Browser\_directory 9](#_Toc468717963)

[4 New features 9](#_Toc468717964)

[4.1 Menu hide 9](#_Toc468717965)

[4.2 Window resize 9](#_Toc468717966)

[4.3 Chart functions 11](#_Toc468717967)

[4.4 Open tab in new tab 12](#_Toc468717968)

[4.5 HTML insert 13](#_Toc468717969)

[5 BrowserGUI setup 14](#_Toc468717970)

[6 References 15](#_Toc468717971)

# Introduction

## Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Rev | Characteristics | Prepared |
| 2012-10-08 | PA1 | Guidline to use BrowserGUI | <EILDVNC> |
| 2012-11-19 | PA2 | HQ57615 correction | ELSZSKU |
| 2016-06-09 | B | Updated for release | ESZILSZ |
| 2016-10-12 | PC1 | Updated with charts functionality | EIMRENA |
| 2016-12-02 | C | Updated for release | EIMRENA |

## About this Document

### How to Read this Document

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

### Knowledge prerequisites

The solution uses XML schema transformations ([5]), HTML cascading style sheets ([6]), and Java scripts ([7]) as the most commonly used HTML data manipulation techniques. Most of this document assumes the knowledge of these technologies. However, some parts (e.g. changing the lookup with replacing or editing files) can be used without any special knowledge.

### Abbreviations

AFS Andrew File System

CLL Core Library

EPTF Ericsson Performance Test Framework

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.

## System Requirements

In order to use the BrowserGUI feature the system requirements listed in EPTF CLL User Guide [4] should be fulfilled. The minimum required web browser version is Mozilla Firefox 38.5.2 Extended Support Release (module name in AFS: firefox/38.5.2esr).

# BrowserGUI

The TitanSim applications have multiple solutions for handling user interactions and show statistics. The command line interface and the java based RuntimeGUI are currently in use. To follow the new technologies, a new GUI was designed and developed. This new user interface is similar to the RuntimeGUI from the GUI layout point of view, which helps the users to be familiar with it in a very short time.

The Browser-based GUI (BrowserGUI) is an HTML based feature, which uses standard technologies in order to support as many web browsers as possible (Mozilla, IE, etc.). It makes the used code understandable, if the customer is familiar with HTML, XSLT and javascript basics.

## How does it work

The main concept of this feature is to define the graphical user interface in an XML file that is generated by the server. The structure of the XML file is described by the XTDP Protocol Module.

This XML file is processed by using an XSL transformation, and the layout is formed by the client side, as it’s shown on Figure 1.

Application

XML data

XSLT,

CSS,

Static files

Browser

XHTML

Figure Main concept of the BrowserGUI behavior

When an HTTP client requests data from the application, it adds an <?xml-stylesheet XML type="text/xsl" href=…> tag to the beginning of the XML data. This XSL file transforms the XML data to an XHTML page. This is the only file required by the BrowserGUI feature for a successful data exchange. The file path and name can be set in module parameters (see 3.3, 3.4 and 3.6). If the file is unavailable for the server, it results in an error message on the client side, in the browser.

## Existing sample solution

The TitanSim framework provides a sample solution that can be used. This section gives a brief overview about the main client side components and files that this solution is built from.

### EPTF\_LoadMain.xsl

As it was mentioned earlier, the GUI is defined in an XML file, which is sent to the client, when the user first loads the web page of the application. This XML file refers to an XSL file. That XSL file defines the transformation rules, which create the HTML code of the page. Each widget type has its equivalent HTML element type. The properties of these items mostly defined in the source XML, like size, orientation or color. These properties are used during the transformation and defined in the HTML code as well.

The XSL file defines only valid, standard HTML elements that can be processed by multiple web browsers. The XSL transformation rules can be edited by the user, in order to process a new widget type or modify an existing transformation. For applying the changes only the web page needs to be reloaded.

### main.css

The generated HTML file uses this CSS file to define the style of the GUI layout. This technology makes it possible to define the visualization information separated from the XSL transformation and the javascript code.

This file defines the font types, size, color and orientation information of the elements. It is possible to create multiple CSS files for this application that gives the possibility to use different skins for the same application without changing the source code or the GUI descriptor XML.

The GUI can be visualized without this file as well, but it will contain mostly plain text, without any style information and background image and so forth. The predefined CSS file is the main.css, this file can be modified or overwrite by the user at any time, the web page needs to be reloaded in order to apply the changes.

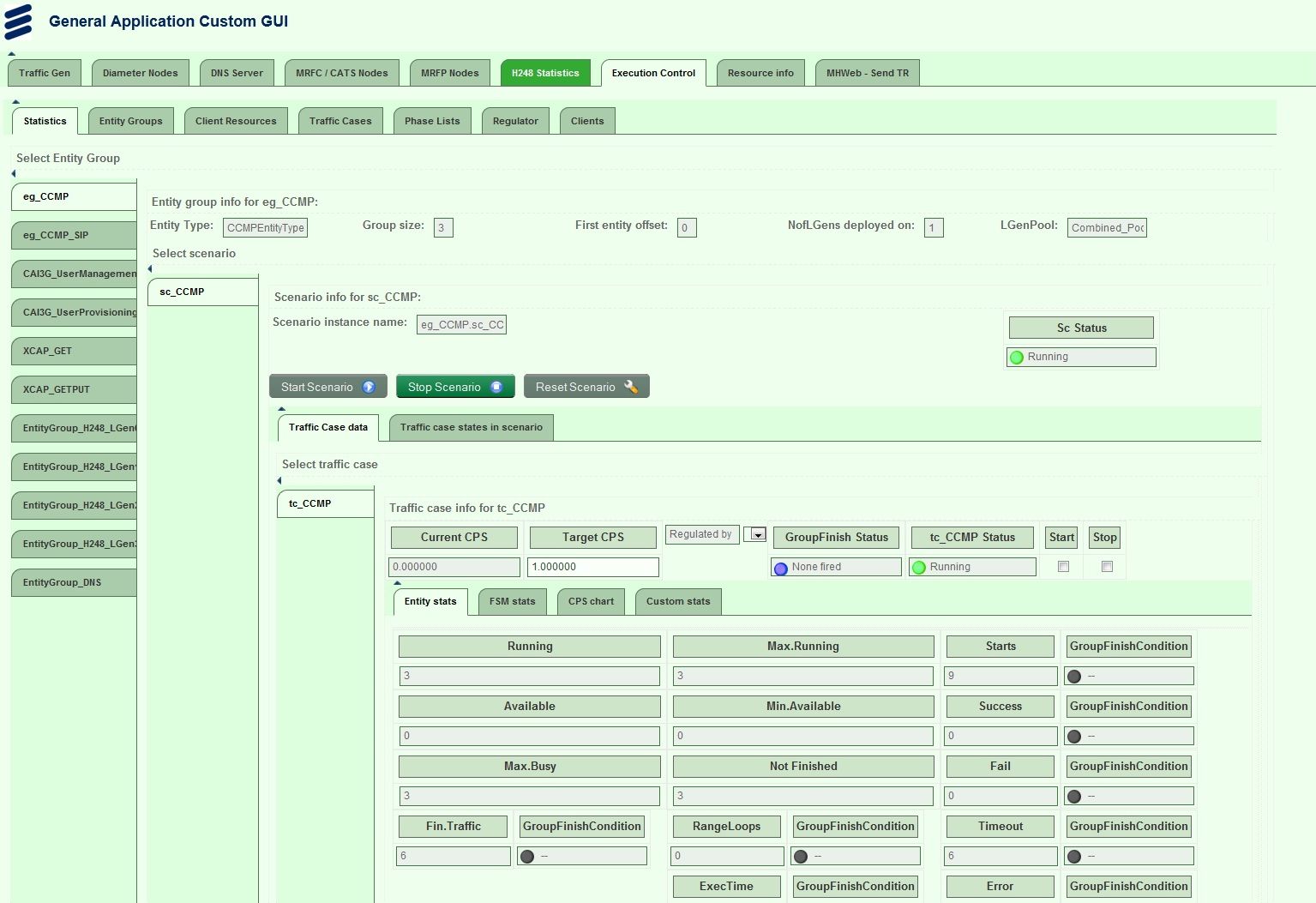


Figure Execution Control tab with green CSS style

### Javascript files

The above mentioned files are used for creating the basic GUI layout. This interface does not contain all the elements, defined in the XML file of the GUI and it is also not capable to communicate with the server side in case of any changes on the web page or handle interactions with it.

Some additional files are needed, which contain the javascript that handles the interactions with the application, visualizes the special elements and provides a refresh method in order to follow the statistics of the running tests in the application.

The BrowserGUI uses AJAX technology in order to refresh the web page in a predefined time interval and updating the server side in case of any, user indicated changes. There are elements on the GUI, like the different kind of charts that also use javascript technology.

### External javascript libraries

External javascript libraries are used to support the chart generation, event handling and AJAX technology that is used for refreshing the page.

We use jquery (jquery-2.2.3.min.js) for supporting the user interaction with the GUI and the refresh method.

### htmlbrowser.js

The htmlbrowser.js file contains the developed javascript itself, which uses the functions provided by the above mentioned libraries. This file can be edited or replaced by the user and the changes can be applied by reloading the webpage.

The main functionalities of the javascript are the following: refreshing the web page elements in every three seconds, handling the tab changes, sending the updates of the web page to the server.

The refresh and update methods are implemented as AJAX requests. The messages have JSon string format, which makes it easier to handle the content on the client side with built-in functions. The JSon string has a predefined format, which contains key-value pairs. In our feature the key is the Id of the refreshed or updated element and the value is the value of that element.

In order to decrease the size of the messages between the client and the server the refresh method requests the new values of only the visible form elements on the active tab of the page. It is important in case of the charts and tables, which contains great number of statistics.

This javascript file handles the chart creation mechanism as well. Every chart has a chart object with parameters that came from the server side in the GUI descriptor XML. When the chart is created for the first time, the client requests historical data from the server, if there is any. When the chart is refreshed only the new value and timestamp value is requested. When the chart is scrolled out from the screen, the chart object is destroyed and it is created again, when the chart becomes visible again.

The javascript handles the tab changes as well, when the user clicks on a tab in the horizontal or vertical menu. The script requests for the XML file only of that tab and not the whole page. When the proper XML file is received from the server, the client transforms it with the XSL transformation and visualizes it by applying the CSS rules as well.

### Image files

The HTML technology gives the possibility to embed images into the GUI layout for instance as background images.

The image files can be edited or replaced by the customer at any time. If you replace the files with files having the same name, nothing else has to be changed. If the new files have different names, the references have to be changed also. (Since the CSS, XSL and JS files are text files, it means a simple text replace.) To apply the changes it is enough to reload the web page.

### Logo

Any application specific logo can be added to the GUI layout, it depends on the customer needs. It can be easily inserted by editing the XSL and the CSS files to set its place and look. The actual version of the feature uses the Ericsson and TitanSim logos (the files are the ericsson\_logo\_top.png and titansim\_logo\_animated.gif).

### Background images

The images can be used as background for example for buttons, or tab menu elements. It is important to avoid the use of too many and too big pictures as their size can make the page loading mechanism slow because of the size of the files the browser downloads, when it loads the page for the first time.

The CSS file defines the position of an image in most cases, which makes it possible to store multiple background images in one file.

### Loading image

Loading images are used for elements that need noticeable time for loading and visualizing the data, like charts. The loading image is only visible until the chart is rendered to the screen. It can be changed by replacing the loading.gif file.

### Led

The leds are used to show status information, as for example the green led indicates that a test is in running state. These leds are inserted from png images, which can be changed by replacing the proper files (led\_green\_16x16.png, led\_red\_16x16.png, etc.).

## Customization

The BrowserGUI is a fully customizable feature of the TitanSim. All the files that were mentioned above can be edited or replaced by the user at any time. The technologies that were used are all based on standards in order to support multiple web browsers and make the customization process more user friendly.

The files can be changed even while the application is running. The changes appear after reloading the web page. In case of changing the module parameters, the application needs to be restarted in order to read the modified configuration files.

# Module parameters

The BrowserGUI feature has some parameters that can be set in the configuration files of the application by setting the values of the following module parameters:

## tsp\_EPTF\_UIHandler\_enableBrowserGUI

The BrowserGUI feature can be turned on or off by setting this module parameter to true or false. The default value of it is true.

## tsp\_EPTF\_UIHandler\_Browser\_RemoteAddress

modulepar charstring tsp\_EPTF\_UIHandler\_Browser\_RemoteAddress := "127.0.0.1";

This parameter defines the IP address of the application, where the BrowserGUI is available. The default is the localhost address, but it can be changed to the IP address of the host machine, where the application is running.

## tsp\_EPTF\_UIHandler\_Browser\_RemotePort

modulepar integer tsp\_EPTF\_UIHandler\_Browser\_RemotePort := 4000;

This parameter defines the port number, where the GUI is available. The default port number is 4000, but it can be changed to any valid port number as well.

## tsp\_EPTF\_UIHandler\_Browser\_xsl2send

modulepar charstring tsp\_EPTF\_UIHandler\_Browser\_xsl2send := "EPTF\_LoadMain.xsl";

This parameter defines the XSL file, which contains the XSL transformation rules that need to be used, when the client visualizes the XML file of the web page that came from the server side.

## tsp\_EPTF\_UIHandler\_Browser\_xsl4Tabpage

modulepar charstring tsp\_EPTF\_UIHandler\_Browser\_xsl4Tabpage := "EPTF\_LoadMain.xsl";

There is a possibility to define a different XSL file for transforming the XML file of the tabpages.

## tsp\_EPTF\_UIHandler\_Browser\_welcomeScreen

modulepar charstring tsp\_EPTF\_UIHandler\_Browser\_welcomeScreen := "welcome.html";

The BrowserGUI feature provides the capability for displaying a welcome screen, while the application is starting. The current welcome screen is defined in the welcome.html file. This file can be edited or replaced as well in order to change the welcome screen if needed.

## tsp\_EPTF\_UIHandler\_Browser\_directory

modulepar charstring tsp\_EPTF\_UIHandler\_Browser\_directory := "../BrowserGUI/";

The path of the files, which were mentioned in this section, has to be defined. The default is a BrowserGUI folder on the same level as the build folder of the application.

# New features

The TitanSim applications have different look, but there are some common components, which is used by every of them. This section gives a brief overview about the usage of the common features of the new web-based GUI.

## Menu hide

The layout usually has several menu items on the screen both horizontal and vertical ones. These elements can use a quite large amount from the screen in case of small display resolution or window size. The BrowserGUI has the capability to collapse these menus and increase the size of the useable part of the browser window.



Figure Menu with full size



Figure Menu after hide

## Window resize

The browser window is mostly resizable and the GUI layout adapts to this change. The BrowserGUI is configured to display a vertical scrollbar as the tables and charts of the statistics cannot fit into a browser window.

The size of the elements on the GUI is decreased, even the size of the charts as well. The GUI keeps its layout as long as it is possible.

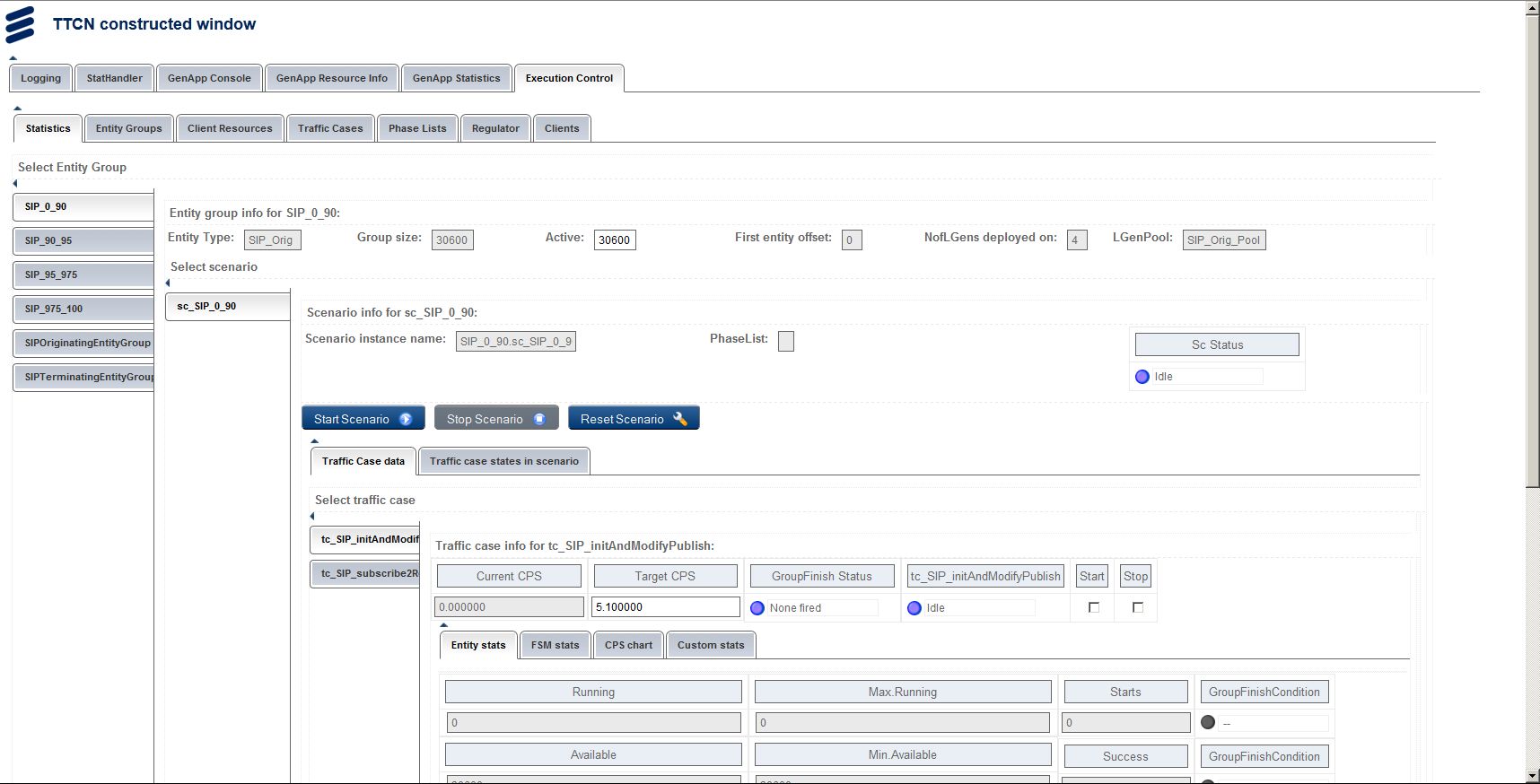


Figure Full size window

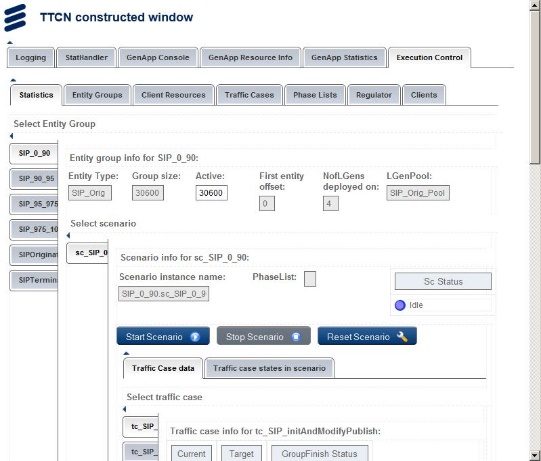


Figure Window after resize

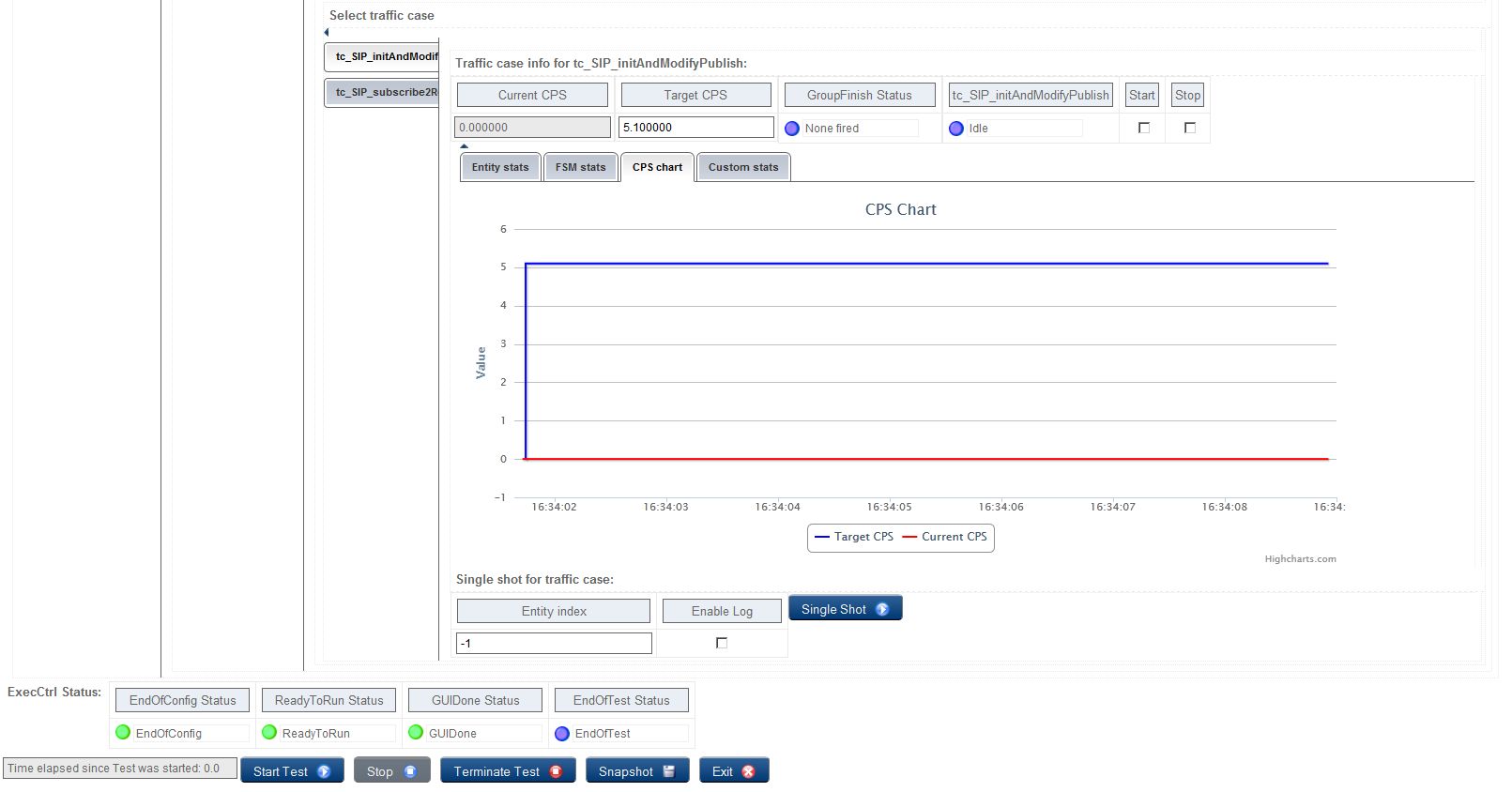


Figure Chart at full window size



Figure Chart after resize

When the horizontal scrollbar is enabled, the GUI layout does not change and the scrollbar appears, if the window width becomes smaller, than the width of the GUI layout.

## Chart functions

Line (e.g.:CPS) and bar (e.g.: density) chart types are supported. Following interactive features are provided:

* Zooming
* Toggle series (if there is more than one dataset in one chart, you can choose to hide certain dataset),
* Save as image (saving chart state as an image in PNG format)

## Open tab in new tab

The TitanSim applications mostly have difficult statistics and resource usage tabs with many values in the tables and on the charts. This results in small and unreadable tables and charts as too much information is positioned onto one page.

The BrowserGUI has the capability to open the tabs of the application in separate browser tabs by right clicking on the tab header in the menu and selecting the Open Link in New Tab option. This can make the tables and charts bigger and after this, the information in these elements is much easier to read.

This solution is the best, when hiding the menu items is not enough. The refresh method work continuously in the new browser tab as well. The difference is spectacular for instance in case of the CPS Chart tab of the Execution Control component, as it can be seen on Figure 12 and Figure 13.

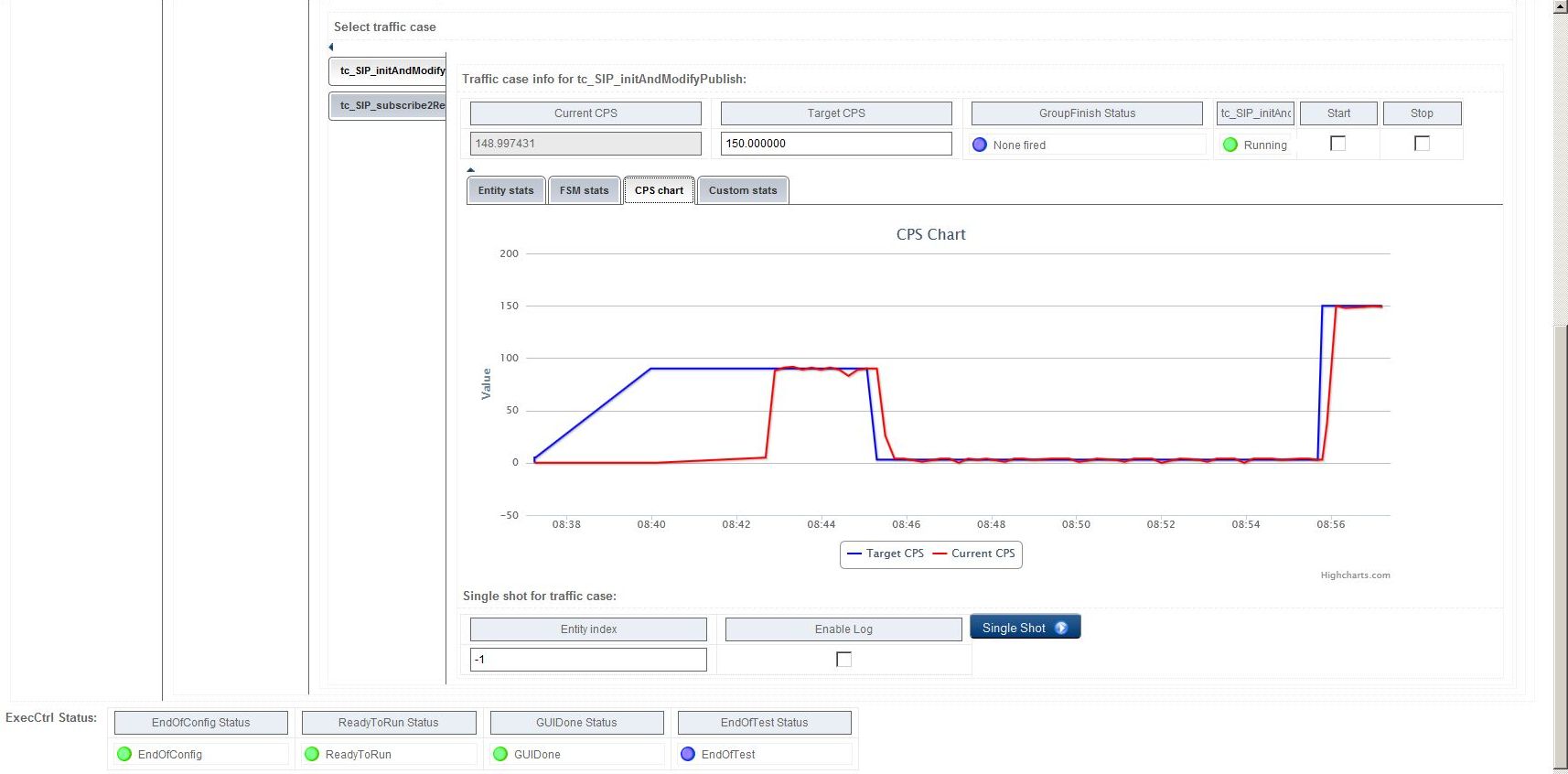


Figure Chart inside of the application window

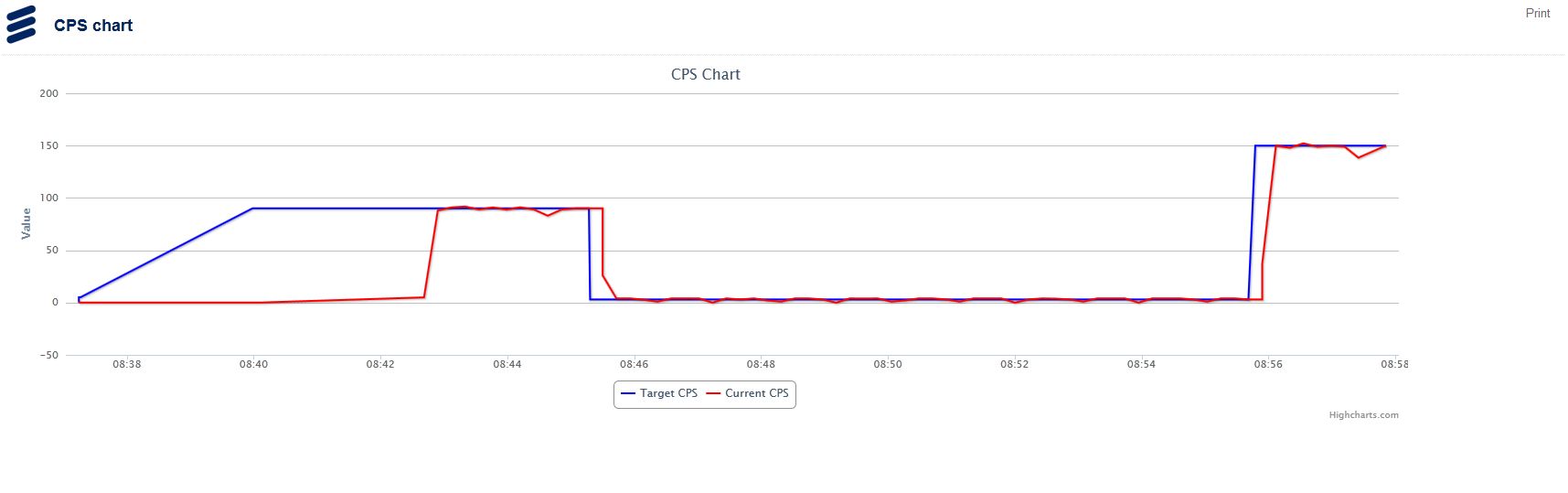


Figure Chart in separate browser tab

## HTML insert

The GUI descriptor xml file can be customized in several ways. The user can assemble the items of an own, individual user interface like a puzzle. The BrowserGUI feature provides additional puzzles parts in the form of HTML code insert.

This feature makes it possible to embed several external HTML pages or even another TitanSim application into one BrowserGUI window. These HTML pages can be visualized in separate tabs, like any other tabs.

The following code can be inserted into a hbox, or tabpage:

<htmlcode>

<value>&lt;iframe width="1100px" height="800px"

src="https://mhweb.ericsson.se/TREditWeb/faces/tredit/tredit.xhtml?create=true"&gt;&lt;/iframe&gt;</value>

</htmlcode>

The value tag cannot contain new line characters and the special values inside it, like < and >, has to be escaped until the next release of Titan. The result of this HTML code insert is depicted on Figure 14.

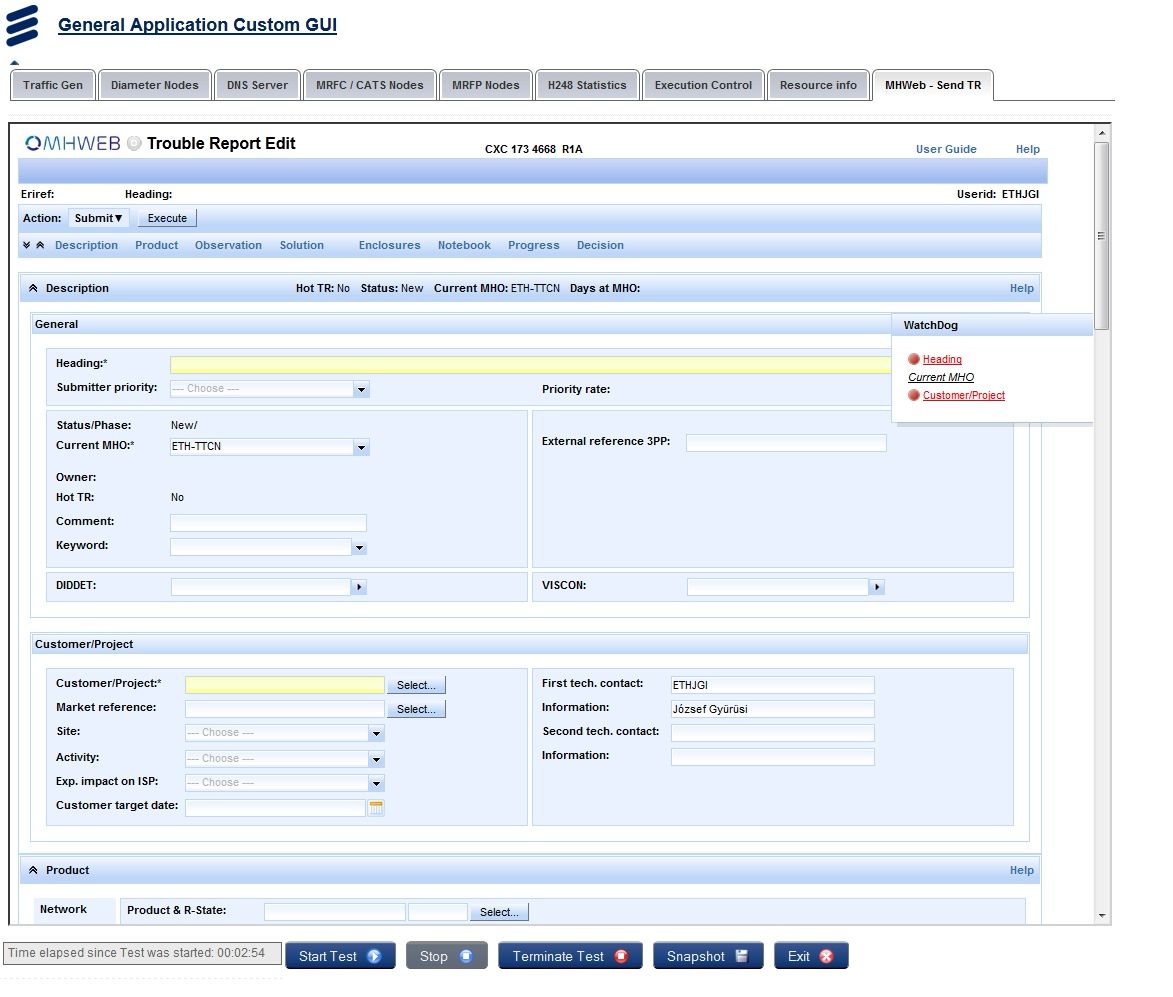


Figure MHWeb inserted into the BrowserGUI on a separate tab

# BrowserGUI setup

The initial steps to start the BrowserGUI feature:

1. The module parameter, which enables the feature, is the tsp\_EPTF\_UIHandler\_enableBrowserGUI. The default value of it is true. If the feature does not start, then check the value of this parameter.
2. Set module parameters

If the application parameters differ from the default values of the module parameters, these parameters need to be updated. The additional files can be replaced by the user or deleted as well. The XSL transformation file is required for the browser to process and visualize the web page definition and the javascript files are required for the additional features of BrowserGUI.

1. Start a web browser

See the required web browser version in 1.3

1. Type the address (<http://browsergui_address:browsergui_port>)

Make sure that the firewall settings are correct, if you try to use the feature from another machine, than the host of the application.

# 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. [Extensible Markup Language (XML)](http://www.w3.org/TR/xml/)
6. [Cascading Style Sheets](http://www.w3.org/Style/CSS/)
7. [JavaScript](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55755)