Graphical User Interface Manual

GIT for Industry

Version: 1.0RC5
Date: 27th November 2015
Author: Smart DCC Ltd.
Classification: DCC Public
## Document Control

### Revision History

<table>
<thead>
<tr>
<th>Revision Date</th>
<th>Summary of Changes</th>
<th>Changes Marked</th>
<th>Version Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>29/10/2015</td>
<td>First version.</td>
<td>N/A</td>
<td>1.0RC4</td>
</tr>
<tr>
<td>27/11/2015</td>
<td>Update</td>
<td>N/A</td>
<td>1.0RC5</td>
</tr>
</tbody>
</table>

### Reviewers

<table>
<thead>
<tr>
<th>Name</th>
<th>Title / Responsibility</th>
<th>Release Date</th>
<th>Version Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gonçalo Gouveia</td>
<td>Critical Software Project Engineer</td>
<td></td>
<td>1.0RC5</td>
</tr>
<tr>
<td>Karim Kanso</td>
<td>Critical Software Senior Engineer</td>
<td></td>
<td>1.0RC5</td>
</tr>
</tbody>
</table>

### Approvals

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Title / Responsibility</th>
<th>Release Date</th>
<th>Version Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>António Alves</td>
<td></td>
<td>Critical Software Project Manager</td>
<td>27/11/2015</td>
<td>1.0RC5</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction ................................................................................................................... 5
  1.1 Objective ................................................................................................................ 5
  1.2 Document structure ............................................................................................... 5

2 Overview ......................................................................................................................... 5
  2.1 High Level Architecture ........................................................................................ 5
     2.1.1 VSIS-Core ...................................................................................................... 6
     2.1.2 VSIS-Comms ................................................................................................. 6
     2.1.3 Simulation ....................................................................................................... 6
     2.1.4 GFI-Testing-Tool .......................................................................................... 6
     2.1.5 Test Library .................................................................................................... 6
     2.1.6 Messages Database ....................................................................................... 7
     2.1.7 GFI-Testing-Tool GUI .................................................................................. 7
  2.2 Reference Testbed ..................................................................................................... 7
  2.3 Inputs and Outputs.................................................................................................... 7
     2.3.1 Test Scenario File ........................................................................................... 8
     2.3.2 Test Properties File ....................................................................................... 8
     2.3.3 Test Report Files ........................................................................................... 8
     2.3.4 Console Output ............................................................................................... 8
     2.3.5 Execution Log ................................................................................................. 9

3 Environment Start-up ...................................................................................................... 9
  3.1 Zigbee HAN network connection .......................................................................... 9
  3.2 System Start-up ....................................................................................................... 9
  3.3 Running the GFI Standalone Application ................................................................ 9

4 GFI Testing Tool GUI Operation .................................................................................... 10
  4.1 Running the GFI Testing Tool application ............................................................ 10
  4.2 Work Environment description ............................................................................. 11
     4.2.1 Explorer ....................................................................................................... 11
     4.2.2 Console / Log Output / Report ..................................................................... 12
     4.2.3 Edit / View Pane .......................................................................................... 12
  4.3 Step-by-step Example ............................................................................................ 12
     4.3.1 Test Creation Wizard ................................................................................... 12
     4.3.2 Edit Modes ................................................................................................... 17
     4.3.3 Test Execution .............................................................................................. 20
     4.3.4 Results Analysis ........................................................................................... 24
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.5 Other Features</td>
<td>30</td>
</tr>
<tr>
<td>Glossary</td>
<td>33</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Objective
This document is the Graphical User Interface Manual for the GFI testing tool. It is intended to provide detailed technical information for standard and advanced user operation and configuration.

1.2 Document structure
Section 1 (Introduction) presents a general description of this document’s contents.
Section 2 (Overview) presents a general description of the framework.
Section 3 (Environment Start-up) presents a description of the environment start-up.
Section 4 (GFI Testing Tool GUI Operation) presents a detailed description of the Test creation, execution and analysis process using the graphical user interface.
Section 5 (Glossary) presents the definitions and acronyms used throughout the document.

2 Overview
GIT for Industry (GFI) is a software tool, provided by Smart DCC, for anybody that wishes to check whether their interpretation and implementation of the Great Britain Companion Specification for smart meters (GBCS) is consistent with Smart DCC’s. GFI v1.0 release candidate 3 supports all GBCS v0.8.1 Use Cases over a ZigBee HAN. In addition to the library of Use Cases, GFI allows end users to create or extend existing GBCS Use Cases. This manual provides detailed technical information for advanced operation of the tool.

2.1 High Level Architecture
GFI is a testing tool and systems’ validation competence centre for GBCS smart meters. In the core of the tool lays a message oriented infrastructure, where the message field is the most elementary entity. In a simplified overview, the tool executes elementary operations over message fields and messages, namely sets, checks, gets, sends and waits, among a few others. Therefore, a fundamental concept that should always be kept in mind is the concept of message and respective fields.

Messages and respective fields are defined in a database, automatically generated from message specifications. This database creates an abstraction layer between the engine of the testing tool (and the tests themselves) and the protocols and interfaces, through which messages are sent and received, leveraging a high level of decoupling between Tests, communication protocols and transmission medium.

To execute and produce the respective reports, modify and/or create Use Cases, a number of steps need to be taken. The following Sections provide detailed technical information for standard and advanced User operation. Also a top level architecture overview is provided in the next Section, describing each one of the main modules that form the testing tool.

Figure 1 provides a high level overview of the system architecture. Besides the GFI Testing Tool Graphical User Interface there are six main modules (shown in purple): GFI-Testing-Tool application interface, VSIS-Core, VSIS-Comms, Simulation manager, Test Library and Messages Database.
2.1.1 VSIS-Core
This is the central element of the framework, provided as a library. This is where all the logic and functionalities of the testing tool are concentrated. Generically, this module is responsible for:

- The creation of the test environment and all the necessary resources for the test execution,
- The execution of the test,
- Generating all the log information,
- Generating all the execution reports.

2.1.2 VSIS-Comms
This is a library responsible for handling the communications that are external to the application, namely the routing of GBCS messages from the KRP to the devices and vice-versa, using the CHF ZigBee Adapter when the test is executed in a real scenario (with non-emulated Devices).

2.1.3 Simulation
The Simulation Manager is responsible for handling the routing of GBCS messages from the KRP to the Devices and vice-versa when the test is executed in an emulated scenario (with emulated Devices).

2.1.4 GFI-Testing-Tool
This is the highest layer of the test framework, responsible for establishing the connections between Core, Comms, Simulation Manager and the Test Library. This tool may be invoked either by the Graphical User Interface (which generates the command to be executed) or directly by the command line.

2.1.5 Test Library
This is the library containing all the Use Cases specified by GBCS and available for the Tester to use in the construction of Tests.
2.1.6 Messages Database
This is a database containing all the messages that provide support for the execution of the Use Cases. For instance, this database contains the General Ciphering Message and General Signing Message as well as all the other messages specified by GBCS.

2.1.7 GFI-Testing-Tool GUI
The Testing Tool's Graphical User Interface, providing functionality for editing, executing and analysing user-implemented Test procedures.

2.2 Reference Testbed
The GFI framework operates as a Home Area Network (HAN) and emulates Remote Parties, ACB and CHF. It creates a ZigBee network and allows Devices to join. Through this network it communicates with Devices in order to execute Tests. In these communications, messages are exchanged using the GBCS protocol: commands are sent and the relevant Devices’ responses and alerts are gathered to produce the Test Reports and to verify the Devices’ conformance with the protocol. The reference Testbed is presented in Figure 2.

Note: The GFI testing tool only performs conformance tests against the GBCS v0.8.1 protocol, not functional tests. Although some minimal functional tests can be implemented and are in fact supported, that is not the purpose of the tool.

![Testbed Diagram](image)

**Figure 2 – GFI Testbed diagram**

2.3 Inputs and Outputs
Table 1 presents the inputs and outputs produced by the testing tool. A short description for each artefact is also presented.

<table>
<thead>
<tr>
<th>File</th>
<th>I/O</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;scenario-file&gt;.xml</td>
<td>Input</td>
<td>Configuration file</td>
<td>The scenario configuration file required by the tool. Specifies which equipment to use, the device</td>
</tr>
</tbody>
</table>
### File I/O Type Description

<table>
<thead>
<tr>
<th>File</th>
<th>I/O</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;properties-file&gt;.xml</td>
<td>Input</td>
<td>Test execution file</td>
<td>The execution file required by the tool. Specifies which Test Cases should be executed and the inputs for each Test Case.</td>
</tr>
<tr>
<td>&lt;test-report&gt;.html</td>
<td>Output</td>
<td>Test report file</td>
<td>The final execution report produced by the tool, in HTML format. Highlights all the relevant actions executed in the test as well as the results of each Test Case, Test Case iteration and Test Case step.</td>
</tr>
<tr>
<td>&lt;execution-log&gt;.csv</td>
<td>Output</td>
<td>Execution log file</td>
<td>The full execution output in comma-separated values format. Contains the full detail available from the tool’s execution. Every single action (set, check, send, wait) performed by the tool is recorded in this file.</td>
</tr>
</tbody>
</table>

**Table 1 – Input and Output artefacts**

The following Sections contain a description for each one of these artefacts and the role they play in the system.

#### 2.3.1 Test Scenario File

To setup a working environment for a Test, all the configurations should be defined in the scenario file <scenario-file>.xml. In this file, the settings for the produced outputs, test scheduler, emulators, equipment and codecs used may be tweaked to cope with environment needs. For example, a Test can be executed in a scenario with real ESME/GSME devices or in a scenario with emulated devices. This configuration is defined in this file.

#### 2.3.2 Test Properties File

The Test Cases that constitute the Test should be defined in the Test Properties file <test-properties>.xml. After the Test specific configurations at the top of the file, information regarding each Test Case (directly mapped to a SMETS Use Case) should be added, namely input parameters and expected outputs.

#### 2.3.3 Test Report Files

Upon Test execution, a formatted HTML report file is generated featuring all the sets, checks, action prints and exchanged messages with a human-readable appearance. This file (along with the Execution Log) may be considered the final output and contains the overall PASS/FAIL information as well as detailed PASS/FAIL information for each expected result.

#### 2.3.4 Console Output

All the relevant actions performed during Test execution are outputted to the console. The detail level of this execution output is maximum by default.
2.3.5 Execution Log

As for the console output, the tool also produces an execution log, with a configurable level of detail defined in the Scenario Configuration file \(<\text{scenario-file}.xml\). The higher the value chosen for the detail level, the more information will be recorded in the log. It should always be the highest possible value (default) for the sake of record keeping. Filters may be applied in the Log view pane for detailed analysis.

3 Environment Start-up

The contents of this user manual are based on the following assumptions:

- The user has access to a machine with the GFI testing tool installed and working.
- The user has access to the system console and permissions to fully operate in the test environment workspace.

3.1 Zigbee HAN network connection

To allow the communications between the testing tool and the metering devices a HAN network needs to be set up. For this purpose a ZigBee USB stick should be connected to a USB 2.0 port. The purpose of this network is described in Section 2.2.

The configurations of this network connection should be defined in the Scenario configuration as described in Section 4.3.1.

3.2 System Start-up

The operating system setup is a Linux Ubuntu distribution with a default user. This user has the required profile for the GFI testing tool, which logs in automatically in the UI desktop and has the following credentials:

- Username: gfi
- Password: gfi

3.3 Running the GFI Standalone Application

In versions 1.0 RC1 and 1.0 RC2, the Testing Tool’s GUI was built as an Eclipse CDT (C/C++ Development Tooling) plugin. Therefore, Users without any background in Eclipse were advised to read the online documentation, namely the “Workbench User Guide” (which is available at http://help.eclipse.org/luna/index.jsp). Recently, the GUI has evolved to a standalone application built to run in Ubuntu 14.04. In spite of this change, the concepts and philosophy of the application are the same, so the recommendation is still relevant.

The Test environment displays the newly created Tests and shows only the relevant menus and options which allow for the user to create new test procedures.
4 GFI Testing Tool GUI Operation

The GFI test environment includes a navigation pane (Explorer) where tests and campaigns are listed; three output panes for the console buffer, for the generated log with filtering capabilities and for the HTML report; and a working area for test case editing and report analysis.

Although this is the default setup, windows and panes can be moved freely, allowing, for example for the last report to be placed within the Edit / View area, side-by-side with other reports to perform a comparison.

The following sections will detail each of these components.

4.1 Running the GFI Testing Tool application

To start the GUI application, use the GFI shortcut on the desktop. Upon start-up, the tests and campaigns created previously will be shown on the Explorer pane.
4.2 Work Environment description

4.2.1 Explorer

The Explorer shows the available Tests and Campaigns. In the context of the GFI GUI, a Test is a procedure enclosing a sequence of test cases and a Campaign is a set of executions of a specified Test. A Campaign is created every time a Test is run.

Test

A Test is composed of two files: The Properties and the Scenario file.

The Properties file contains the information of all test cases in the sequence – configurations, input/expected parameters values, as well as items that will be featured in the test report (such as the Test Id, Test Name and Test Purpose).

The Scenario file holds the setup for the various configurations needed to run the tool (like the output console and log levels, schedulers, remote parties and metering devices emulators, SMETS object storage information and message codecs).

Campaign

A Campaign is generated when a Test is executed. It contains a Test section, a set of Runs and a Metrics file.

The test section contains a copy of the test files at the date and time of the first Run (i.e. when the campaign was created). These files are a snapshot of the original test and are, hence, not editable.

A Run contains the execution Report and Log files for analysis. Future Runs of this Campaign (generated by running this Campaign and not the original Test) will use these same test files. A new Run will always be created at the time/date of a Campaign execution.

A Metrics file is also created on Campaign generation and updated on each Campaign execution. It contains statistics on the Campaign’s Runs PASS/FAIL report information.

See also Figure 27 – Multiple Runs for a depiction of Tests and Campaigns in the application’s Explorer pane.
4.2.2 Console / Log Output / Report
Upon a given test execution, the console feed will be shown in the console pane. In the same way, the execution log and the report will be available for analysis in their respective panes. See Section 4.3.4 for more information about the execution log and report.

Figure 5 – Console

4.2.3 Edit / View Pane
On the working area it is possible to edit the Test properties and scenario through the Basic (or Advanced) editor by double clicking the respective entry in the Explorer. Also, the execution output reports and the log files may be viewed in this area for detailed analysis. Although panes can be moved freely within the Application Workbench, last execution items (logs / reports) will be shown within the bottom pane right after being generated, whereas files opened by the user (by double-clicking on its item in the Explorer pane) will be shown within the Edit / View Pane as shown in Figure 29.

4.3 Step-by-step Example
In this section a Step-by-step example will be presented to illustrate the testing tool operation. The example presented is for an ESME device in which the Meter Point Administration Number is altered between different values and checked in the meter after every change.

The overall structure that is to be followed for creation of a test is as follows:
Open Wizard → Basic Properties → Define Scenario (e.g. Emulated/ESME/GSME) → Add Use Cases

4.3.1 Test Creation Wizard
Starting with an empty workspace, the first thing to do is to create a New Test. This may be done by invoking the New Test wizard from the context menu on the Explorer, from the File menu or from the respective button on the toolbar, then filling in the required options and configurations.
Properties

This dialog gathers the initial information required by the test Properties file (Section 2.3.2). Specifically, the user is prompted to enter the following information:

- ID: This is the identifier that will be used for the execution file name and shown on the Explorer pane.
- Name: A user defined name, typically a short but meaningful designation.
- Description: The detail used to include in the test purpose section of the test report.

See example in Figure 7.

Scenario

This dialog gathers the information required by the test scenario such as the network settings, outputs, scheduling, equipment, etc.. From these inputs, the file referred to as <scenario-file>.xml (Section 2.3.1) will be generated.

If the test uses meter emulation only the type of meter is available for selection.
If a physical (non-emulated) metering device is used, the following meter configurations need to be set properly:

---

**Figure 8 – Scenario selection (emulators)**

**Figure 9 – Scenario selection (physical meters)**
• ID: Entity Identifier of the Device in the Whitelist of the CHF.

• Installation Credentials: Installation credentials (installation code) of the Device in the Whitelist of the CHF.

• Digital Signature Certificate: Path to the .der file containing the device certificate.

• Key Agreement Certificate: Path to the .der file containing the device certificate.

• MAC: Entity ID (matching ZigBee EUI64 address) as in GBCS documentation.

• Pan ID: The ZigBee network Pan ID.

• Extended Pan ID: The ZigBee network extended Pan ID.

• Network Key: The ZigBee network key.

• Port: The serial port where the ZigBee Adapter is connected.

• Channel: The chosen channel (ranging from 11 to 26) and

• Join Timeout: The join timeout, specified in seconds.

Initially, the fields in this dialog are shown blank (as in Figure 9) but after a test is created successfully, the configuration is saved. These fields will then be always populated with data from the last known configuration. Using the button ‘Restore Defaults’ will revert to that same last known configuration.

**Search and Select Test Cases**

In this section test cases are selected to build the sequence for the new test. Clicking in the ‘Add Test Cases’ button (on the upper left corner of the dialog) will display a list of the test cases present in the test library – the SMETS Use Cases already coded.

![Search and Select Test Cases](image)
The required test cases should then be selected and added to the sequence. A filter may be applied to refine the test case list as shown in Figure 11. Upon selection, the information of a test-case is displayed in the pane on the right ‘Selected test case info’. Also, multiple selection is allowed for addition.

Figure 11 – Add Test Case

After all the required test cases are added, the sequence in the ‘Search and Select Test Cases’ dialog is populated.

At this point it is possible to tune the Test procedure. On the upper right corner, the arrow icon buttons (‘Move Up’ and ‘Move Down’) allow a selected test case to be moved up or down the sequence. On the upper left corner, next to the ‘Add Test Cases’ button, lay the ‘Delete’ and the ‘Duplicate’ buttons. As their names imply, they allow the possibility to remove or duplicate the selected (possibly more than one) test cases. Duplicated test cases will be added to the end of the list, if needed, they should then be moved to their correct order in the sequence.
Clicking ‘Finish’ in this dialog will create the new test which will then be listed in the GUI main window’s Explorer. Test Case data (such as number of iterations and the values for input/expected values) is created from default values present in a Test Case library. Repeated instances of a test case are automatically assigned a different identifier by appending a numerical suffix to the default test case identifier. This identifier can still be edited later. Properties and Scenario may be viewed and edited in the working area should any changes be required for a specific sequence (e.g. expected values on the output parameters). An example of this is given in Section 4.3.4. The tester will also be allowed to edit these parameters in Advanced Mode (directly in the XML files), changing the configuration through the application menus. Detailed information on the subject can be found in the next Section (4.3.2 - Edit Modes).

4.3.2 Edit Modes

There are two ways in which a Test Case may be edited. By default the Basic Mode provides graphical interfaces that allow the setting of Test / Test Case information as well as the addition of further Test
Cases and Iterations. Also, the parameters’ values, both Input and Expected may be edited according to the Test requirements. Alternatively, an Advanced edition mode is available for experienced users. The edition mode may be toggled using the application menus through **Edit ➜ Preferences**. A dialog will be presented where this configuration may be altered.

![GFI Preferences](image)

**Figure 14 – GFI Preferences**

**Basic Mode**

In order to edit the Test Scenario, all the configurations detailed in the previous section (Figure 9 – Scenario selection (physical meters)) may be tweaked. If an emulated environment is being used there will be no configurations to be change.

![Edit Scenario](image)

**Figure 15 – Edit Scenario**

To edit the Test Properties in Basic Mode (default), there are two tabs available: The Test Editor and the Test Case Editor.

- Test Editor: Contains the Test information for the report;
Test Case Editor: Allows all sorts of interaction with the test sequence. Adding, removing or duplicating either Test Cases or Iterations, as well as changing the parameter values (input or expected) in each Iteration. Test Cases may also be moved up and down in the test sequence.

The data in each test case is set by default and has been previously tested. The values are mainly gathered from the SMETS documentation where they are described as the default values for the SMETS object in the meter.

Advanced Mode
There is also the possibility to edit the Test Properties and Scenario directly on the respective XML files as described before.

Each input or expected value in a test case is defined as a resource (as it could be used in any possible way) and included in a `<resourcepool>` element (in the XML file) for that parameter. Each entry in a resource pool will be used in each of the test case’s iteration—‘n’ entries in the resource pool will become ‘n’ iterations in the test case. All resource pools should have the same number of data entries as, in the
scope of GFI, only linear combinations will be used – a resource pool with a lower number of entries will loop through the values in the extra iterations. On Section 4.3.4 a practical example is presented.

Switching to Advanced Mode as in Figure 14 – GFI Preferences, selecting the radio button ‘Advanced’ and clicking ‘OK’ will activate this edition mode. Testers are advised to use this mode with care as XML tags and properties misuse may corrupt the Test Case and render it unusable.

4.3.3 Test Execution

Using the example presented previously, this section will detail the test execution procedure.

A Test previously created may be executed through the ‘Run’ button in the toolbar or the option ‘Run’ in the Test’s context menu.
After clicking OK in the confirmation box the Test is executed.

![Confirmation Dialog]

**Figure 20 – Test execution confirmation**

A new feature in 1.0 RC3 is a database of counters per Remote Party/Device that is maintained by the application. The purpose of this database is to account for the situations where protection against replay is used (i.e. the meter will reject a message if its counter is lower than the meter’s). Every time a message is sent the counters used in message exchange are incremented and the total increment stored in the database after an iteration is concluded.

If a given Remote Party / Device association is being used for the first time, their counters are yet to be initialised in the database. So, if this is the case, a dialog is presented to obtain the Tester’s confirmation of the counters starting values.

![Remote Party Counters]

**Figure 21 – Remote Party Counters**

Although the counter values may be changed, they will be initially filled in with defaults read from the Test Scenario. Also, the Tester is allowed to change these values later on by means of the application menus: **Edit -> Counters**.
Click ‘Continue’ (Figure 21) for the test to be executed.

The console activity may then be observed in the console pane. It is the actual real-time output of the GFI execution as if it would have been run through the command line.

When the execution is finished, the execution log and the report are presented in the Execution Log pane. Only the report corresponding to the last execution of a test is displayed in this pane – all previously saved reports are displayed in the Edit/View pane. Upon running the test a new Campaign item resulting from this Test’s execution is added to the Explorer pane, along with all the respective configuration, execution and metric files.
During the execution, it is possible to skip the current iteration by clicking the skip button in the console pane. In much the same way a test execution may be stopped using the stop button.
This concludes the Test execution. As mentioned before, should any further executions of this Campaign be performed, new ‘Run’ items will be added to the Campaign. Old runs may be deleted but are kept by default for later analysis.

**Note:** Whenever a Test is executed, a new Campaign is generated. If, on the other hand a Campaign is executed, a new Run (belonging to this Campaign) is generated. Campaigns are ALWAYS executed based on the same test conditions (Properties and Scenario). Campaign Tests are not editable – only standalone Tests may be changed.

Figure 27 depicts multiple executions of a Campaign.

### 4.3.4 Results Analysis

In the previous section the Test was executed without any changes to the original test cases in the library, so the result was PASS for all of them. Test cases that read parameters from metering devices have been stripped of expected values (other than status or results that should be zero when messages are
exchanged successfully) as it is not possible to know what their values will be until actual meters are plugged in the system. Being so, no expected values in the read Test Case (like ATG-TC-ECS40 in our example) means that no check is performed but the parameter still gets printed in the report.

Using emulated devices we may change these Test Cases to perform the check adding a value to the respective Expected entry. The default SMETS object storage values were previously tested and known to yield pass results, these are:

- **MeterAdministrationPointNumber(combinedMPANS):**
  
  31 33 31 32 33 34 35 36 37 38 31 31 31 31 33 31 32 33 34 35 36 37 38 32 32 32

- **MeterAdministrationPointNumber(exportMPAN):**
  
  31 33 31 32 33 34 35 36 37 38 33 33 33

These values should be set in both Iterations of both read Test Cases (ATG-TC-ECS40-(1) and ATG-TC-ECS40-(2)).
Note that the report was moved to the view pane so it's side by side with the properties window.

However, the purpose (as described in the beginning of this Step-by-step example) was to assign different values to the MPAN parameter. This may be done by editing the Properties file in the Test, changing the input parameter to whichever value is required for the test purpose. In this example, the second update test case (ATG-TC-ECS39a-(2)) will be setting a different value.

This test case runs two iterations as seen in Figure 19 – two data values per resource pool in the XML if editing in Advanced mode. If the purpose of this test case in the Test is to set a specific value to the parameter rather than testing parameter boundaries, only one iteration is required to attain this objective. There are two ways to go about it:

- Remove the redundant iterations – On the Test Cases Editor select the iteration to be removed and click the ‘Remove’ button. Repeat the process until all redundant iterations are removed. Change the parameter values on the Iteration Detail form as required. If editing in Advanced mode, leave just one value per resource pool and set the value of the resource pool that relates to the parameter requiring change.

- Change the last iteration – The easier way as no significant editing is required. Just change the parameter value on the Test Case’s last iteration. Values set in previous iterations will be overwritten in the last one. If in Advanced edit mode set the last value (last iteration) in the resource pool that relates to the parameter requiring change.

Going with the second option, only the value highlighted in Figure 30 needs to be changed. Note that the proper Iteration should be selected.

![Figure 30 - Edit Test Case Properties - before](image)

After assigning a different value to the parameter, the Iteration Detail should look like Figure 31.
The file should then be saved and the Test executed.

After the new execution the Test will fail because the subsequent read test case (ATG-TC-ECS40-(2)) is still expecting to find the default values.
Changes need to be made so the check verifies the correct MPAN value set in the equipment - this should be the value set during the last iteration of the preceding update test case.

Again, if the purpose is to check the value only one iteration is needed. Either redundant iterations should be removed or all iterations should account for the parameter value and expected values in all iterations should be changed.

Figure 35 shows how the Expected value that needs to be changed. Both Iterations of ATG-TC-ECS40-(2) should undergo this correction, otherwise one of them will fail causing the failure of the Test Case and, consequently, the failure of the Test procedure’s overall result.
Again, the file should be saved and the Test executed.

This time the result is PASS as the expected value matches the response message field read. Meaning that the parameter was updated successfully.

As shown in Figure 36, a new Campaign is added for every Test run – The Test had to be re-run (not the Campaign) as the test cases in the sequence were subject to changes (not allowed in the Campaign).

Also, the reports may be viewed side by side for result analysis in a comparative fashion. This is allowed for every input or output artefact (console output or execution logs).
Logging

Further analysis may be performed in the test Log using filters in some columns or just by text. The icon on the upper right corner of the Log pane will display a Filter dialog.

![Open Filter Dialog](image)

Figure 37 – Open Filter Dialog

This will facilitate manual checks on exchanged messages. The filter can be applied to the combination of area and log levels but further filtering is possible by text keywords. If a log is very extensive it is possible to limit the number of log entries in the log filter by limiting the visible entries.

4.3.5 Other Features

Export / Import

This functionality allows saving Tests, Campaigns or Runs out of the application context for the purpose of archiving or sharing.

The Test export is composed of the Properties and the Scenario files. Unlike Campaigns or Runs, these may be used for posterior Import (creating a new Test), through the context menus or the 'Import' button.
Campaigns and Runs may also be Exported. The Campaign is composed of the Test files (containing the Properties and Scenario), all the Runs (each containing the Report files and the Execution Log) and the Metrics file. A single Run, in addition to the Report files and Execution Log, also contains the Test files from the respective Campaign.

All Export actions may be invoked either from the ‘Test’, ‘Campaign’ and ‘Run’ context menus or the ‘Export’ button in the tool bar. Dialogs will be displayed to select the output path for the produced files.
The selected output path should have write permissions for the user.

![Select output path](image)

**Figure 40 – Select output path**

If a whole campaign is exported, the output directory will contain the whole campaign structure as shown in the Explorer Pane. As referred before, this will include all available Run folders, as well as the Test directory that was used to generate them. On the other hand, if a single run is exported, then the output path will contain only the files used and produced by this Run (Test and current Run directories). Each run generates an execution log, a formatted test report (HTML), and an XML report file, which may be used by a report integration tool using the raw data as its input (unformatted data).

![Exported files](image)

**Figure 41 – Exported files**

**Search and Open**

A basic search/open functionality allows the user to search for test cases, tests and campaigns. This functionality is accessed via the navigator’s context menu, the main menu or selecting the icon in the toolbar.

The search and open dialog will load all entries found in the workspace. A text box will provide filtering, reducing the scope. Selecting one element will display its path (i.e. the project it belongs to) below the
grid. Test Projects (referred to as ‘Test Specs’) and Campaigns are also displayed in the grid as an entry of its own.

![Graphical User Interface Manual](image)

**Figure 42 – Search and Open**

Selecting ‘Open’ will open the project which the selected item belongs to with the appropriate editor (Test – Editable/Campaign – Read-Only) reflecting the currently selected mode in the preferences (Basic/Advanced).

## 5 Glossary

Table 2 presents the list of definitions used throughout this document.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicable Document</td>
<td>A document is considered applicable if it complements this document. All its content is directly applied as if it was stated as an annex of this document.</td>
</tr>
<tr>
<td>Reference Document</td>
<td>A document is considered a reference if it is referred but not applicable to this document. Reference documents are mainly used to provide further reading.</td>
</tr>
<tr>
<td>Test</td>
<td>A set of Test Cases (ranging from one to N) that are grouped together with the purpose of being executed in one run.</td>
</tr>
<tr>
<td>Test Case</td>
<td>A Use Case as defined by SMETS.</td>
</tr>
<tr>
<td>VSIS™</td>
<td>A CSW Critical Systems Validation Platform</td>
</tr>
</tbody>
</table>

**Table 2 – Definitions.**
Table 3 presents the list of acronyms used throughout this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACB</td>
<td>Access Control Broker</td>
</tr>
<tr>
<td>AD</td>
<td>Applicable Document</td>
</tr>
<tr>
<td>ATG</td>
<td>Automated Test of GBCS</td>
</tr>
<tr>
<td>CBKE</td>
<td>Certificate-Based Key Exchange</td>
</tr>
<tr>
<td>CHF</td>
<td>Communications Hub Function</td>
</tr>
<tr>
<td>CSW</td>
<td>Critical Software, S.A.</td>
</tr>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>ESME</td>
<td>Electricity Smart Metering Equipment</td>
</tr>
<tr>
<td>GBCS</td>
<td>Great Britain Companion Specification</td>
</tr>
<tr>
<td>GFI</td>
<td>GIT For Industry</td>
</tr>
<tr>
<td>GIT</td>
<td>GBCS Interface Testing</td>
</tr>
<tr>
<td>GSME</td>
<td>Gas Smart Metering Equipment</td>
</tr>
<tr>
<td>KRP</td>
<td>Known Remote Party</td>
</tr>
<tr>
<td>NA</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>PWO</td>
<td>Pass With Observations</td>
</tr>
<tr>
<td>RD</td>
<td>Reference Document</td>
</tr>
<tr>
<td>TBC</td>
<td>To be confirmed</td>
</tr>
<tr>
<td>TBD</td>
<td>To be defined</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>UTRN</td>
<td>Unique Transaction Reference Number</td>
</tr>
<tr>
<td>ZCL</td>
<td>ZigBee Cluster Library</td>
</tr>
</tbody>
</table>

Table 3 – Acronyms.