## 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>
<tr>
<td>22/07/2016</td>
<td>Update</td>
<td>N/A</td>
<td>1.3.2E</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>
<tr>
<td>Dinis Paes</td>
<td>Critical Software Technical Manager</td>
<td></td>
<td>1.3.2E</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>
<tr>
<td>Dinis Paes</td>
<td></td>
<td>Critical Software Technical Manager</td>
<td>25/07/2016</td>
<td>1.3.2E</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 Simulation ......................................................................................................... 6
      2.1.3 GFI-Testing-Tool ............................................................................................. 6
      2.1.4 Test Library ..................................................................................................... 6
      2.1.5 Messages Database ........................................................................................ 7
      2.1.6 GFI-Testing-Tool GUI ................................................................................... 7
      2.1.7 Communications Hub ...................................................................................... 7
  2.2 Reference Testbed ..................................................................................................... 8
  2.3 Inputs and Outputs.................................................................................................... 9
      2.3.1 Test Scenario File ............................................................................................ 10
      2.3.2 Test Properties File ....................................................................................... 10
      2.3.3 Test Report Files ............................................................................................ 10
      2.3.4 Execution Log ................................................................................................. 10
      2.3.5 Communications Hub Configuration File .................................................... 10

3 Environment Start-up ................................................................................................. 11
  3.1 Zigbee HAN network connection .......................................................................... 11
  3.2 System Start-up ...................................................................................................... 11
  3.3 Running the GFI Standalone Application ............................................................... 11

4 GFI Testing Tool GUI Operation ............................................................................... 11
  4.1 Running the GFI Testing Tool application .............................................................. 12
  4.2 Work Environment description .............................................................................. 12
      4.2.1 Explorer ........................................................................................................... 12
      4.2.2 Console / Log Output / Report ....................................................................... 13
      4.2.3 Edit / View Pane ............................................................................................. 14
      4.2.4 Toolbar ............................................................................................................ 14
  4.3 Step-by-step Example ............................................................................................. 15
      4.3.1 Communications Hub Configuration ............................................................ 15
      4.3.2 Test Creation Wizard ..................................................................................... 22
      4.3.3 Edit Modes ...................................................................................................... 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.4 Test Execution</td>
<td>31</td>
</tr>
<tr>
<td>4.3.5 Analysis of Results</td>
<td>35</td>
</tr>
<tr>
<td>4.3.6 Other Features</td>
<td>41</td>
</tr>
</tbody>
</table>

5 Glossary

47
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 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 who wishes to check the consistency and correctness of their interpretation and implementation of the Great Britain Companion Specification for smart meters (GBCS) against Smart DCC’s. GFI v1.3.2E supports all GBCS v0.8.2 use cases over a ZigBee HAN. In addition to the library of test-cases provided, GFI allows end users to create new test-cases or extend existing ones. This manual provides detailed technical information for advanced operation of the GUI provided to interact with 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 test-cases, a number of steps need to be taken. The following sections provide detailed information for standard and advanced user operation. Additionally, the next section provides an overview of the top level architecture, describing each of the testing tool's main modules and the part they play in the system.

Figure 1 depicts the high level overview of the system architecture. Besides the GFI Testing Tool GUI there are six main modules (shown in purple): the GFI-Testing-Tool application, VSIS-Core, the Communications Hub, the Simulation manager, the Test Library and the Messages Database. Following is a small description of each of these modules.
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 Simulation
GFI supports testing over a fully-emulated scenario, and also using physical devices over a ZigBee network via the communications hub module.

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 a fully-emulated scenario (with emulated Devices only).

2.1.3 GFI-Testing-Tool
This is the highest layer of the test framework, responsible for establishing the connections between Core, Simulation Manager, Communications Hub and the Test Library. This tool may be invoked either by the GUI (which generates the commands to be executed) or directly from the command line.

2.1.4 Test Library
This is a library containing test-cases based on the use cases specified by GBCS and available for the tester to use in tests.
2.1.5 **Messages Database**

This is a database containing all the messages that provide support for the execution of the test-cases. For instance, this database contains the General Ciphering Message and General Signing Message as well as all the other messages specified by GBCS.

2.1.6 **GFI-Testing-Tool GUI**

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

2.1.7 **Communications Hub**

Beginning with GFI version 1.3.2E, an implementation of a communications hub\(^1\) is offered which acts as a gateway to a persistent ZigBee HAN where physical devices may be connected, and supports several CHF and GPF features including maintaining a whitelist of devices allowed in the HAN and mirroring of GSME data. Figure 2 shows the Communications Hub’s architecture. Next is a brief description of each block.

\(^1\) This should not be confused with the CHF/GPF emulators offered in the fully-emulated option. This module is the gateway to a real HAN (using physical devices) via a ZigBee adaptor.

---

![Communications Hub](chart.png)

Figure 2 – Communications Hub’s Architecture

2.1.7.1 **Daemon Process Control**
The Communications Hub runs as a service. As such, and upon starting, it will run in the background waiting for requests. The Daemon Process Control is an ordinary script which provides an interface to the Communications Hub process, supporting the usual service commands such as start, stop, status and restart. To start the Communications Hub, a configuration file needs to be provided. The GUI provides a friendly user-interface to build the start-up configuration file and issuing the commands to control the communications Hub.

One consequence of being a service is the capability of persisting information between tests, as will be seen in later sections, something which cannot be achieved using full emulation, as state is not kept between tests in this latter case.

2.1.7.2 CHF

The Communications Hub Function will service all test-cases in which it is the target device (e.g. CCS01) and validate and forward messages destined to other devices (e.g. GPF, ESME). As show in Figure 2, the GFI-testing-tool uses a TCP port (referred to as WAN TCP port) to send/receive the GBCS messages. There is an additional TCP port used by the Communications Hub referred as the Control port, which can be used by any application to issue commands to the Communications Hub (it is used by the GUI to query the list of devices).

2.1.7.3 GPF

The Gas Proxy Function’s main role is to support a copy of GSME data. This copy is maintained using the ZSE mirroring and GBCS Tapping Off (TOM) mechanisms. The GPF is then able to provide this data both to the WAN using the relevant GBCS use-cases and to the HAN using ZSE commands.

2.1.7.4 Coordinator

The coordinator manages the interactions with the ZigBee stack and makes the functionality supported by the stack available to the CHF and GPF modules.

2.1.7.5 Endpoints

The Communications Hub provides the following ZigBee endpoints:

- Remote Communications Device Endpoint: used for communication in the HAN.
- Gas ESI Endpoint: mirroring of GSME data and processing of Tapping Off commands.
- Gas Mirror Endpoint: mirroring of GSME data.

2.2 Reference Testbed

As mentioned before, GFI has two modes of execution: fully-emulated and physical. In the former, all devices (e.g. ESME, GSME, CHF, GPF) and remote parties (e.g. Supplier, ACB) are emulated. In the latter case an implementation of a communications hub is used, which will serve (amongst other things) as the entry point to a HAN ZigBee network where physical devices can be connected. Messages from remote parties are routed via the network to the target devices in order to execute tests. Messages are exchanged using the GBCS protocol: commands are sent and the relevant devices’ responses and alerts are gathered to produce the test reports. Conformance with the protocol can be deduced by analysing the test reports produced. The reference testbed is presented in Figure 3.
Note: The GFI testing tool only performs conformance tests against the GBCS v0.8.2 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.

The test-case library provides a set of test-cases with sample iterations. Whilst these were designed to pass using emulation, the values of their parameters will need to be adapted to a real case.

![Figure 3 – GFI Testbed diagram](image)

# 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><code>&lt;scenario-file&gt;.xml</code></td>
<td>Input</td>
<td>Configuration file</td>
<td>The scenario configuration file required by the tool. Specifies which equipment to use, the device configurations, keys and certificates, among other configurations.</td>
</tr>
<tr>
<td><code>&lt;properties-file&gt;.xml</code></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><code>&lt;test-report&gt;.html</code></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><code>&lt;execution-log&gt;.csv</code></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.</td>
</tr>
</tbody>
</table>
### File | I/O | Type | Description
---|---|---|---
<ch_conf_timestamp>.xml | Input | Communications Hub configuration file | An XML configuration file which the CH loads upon booting up. Defines general settings, ZigBee radio configurations and may also add a set of devices to the CH’s whitelist.

**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 Execution Log

The tool also produces an execution log, with a configurable level of detail defined in the Scenario Configuration file `<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.

#### 2.3.5 Communications Hub Configuration File

To start the Communications Hub a configuration file needs to be provided. The GUI can be used to create and store several configurations, corresponding to different environments (e.g. different ZigBee radio settings and/or pre-configured whitelists). The GUI provides a user-friendly way of generating this file and issuing the commands to the CH, shielding the user from having to deal with the command line. The files generated feature a timestamp and will always be deleted upon shutting down the daemon. The process of generating a CH configuration file should ideally be the first step when the tool is operating in physical mode, and will be described in detail in section 4.
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 Communications hub configuration file, 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 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 (one for the consoles, one for the generated log with filtering capabilities and one for the HTML report); a working area for test-case/scenario edition and report analysis and a toolbar where actions are available (by pressing buttons) depending on the items selected.
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.

![GFI Testing Tool GUI](image)

**Figure 4 – GFI Testing Tool GUI**

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, message codecs and certificates).

**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 47 – Multiple Runs for a depiction of Tests and Campaigns in the application’s Explorer pane.

**4.2.2 Console / Log Output / Report**

The Console pane will show information from three different sources: Test/Campaign execution, outcome of commands sent to the Communications Hub and the live contents of the Communications Hub log file after a given moment in time. Although the information from these three sources will be shown in the same pane, it will be displayed in different pages (the user will be able to switch between these pages).

Information regarding the last test/campaign run will also be available in the log output pane (in a tabular, searchable form) and in the report pane. These panes display in the information in a format more suitable for analysis.

These concepts will be explored in depth in latter sections.
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 49.

4.2.4 Toolbar

The toolbar provides easy and quick access to a set of actions, some of which will only be available if specific items have been selected beforehand (e.g. 'Run' will only be enabled if a Test or a Campaign have been selected).

Because the GUI can control the Communications Hub module introduced in GFI1.3.2E, a specific subset of the toolbar is dedicated to this purpose. Figure 7 highlights these actions, which will be explored in detail in the step-by-step example.

---

2 Most of the actions are also available via a right click over an item or via a menu. It’s not the case of the Communications Hub actions which are only available via the designated buttons on the toolbar.
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.

As mentioned before, a test can be run in emulated mode or, using the appropriate hardware (i.e. a ZigBee radio dongle) in a physical environment with real devices. This version of GFI features a Communications Hub implementation which can be controlled via GUI. This step-by-step example will focus on the physical setup and as such the first step will be to configure and start the Communications Hub.

The basic steps carried out in the following sections to create/execute a test will be as follows:

Configure and Start the Communications Hub → Define Test using the Wizard → Execute Test.

4.3.1  Communications Hub Configuration

Assuming no configuration has yet been created or that the Communications Hub hasn’t been started from within the GUI, there are two ways of displaying the CH Configuration Management dialog:

- The ‘Configure/Start Comms Hub’ icon in the toolbar (refer to Figure 7) or
- Selecting the arrow next to it and then choosing ‘Configure/Start Comms Hub Configurations…’ (Figure 8 and Figure 9).

![Figure 8 – Selecting the arrow next to the ‘Configure/Start Comms Hub’ button](image)

![Figure 9 – Selecting the appropriate option from the menu](image)

The CH configuration management dialog should then be visible (Figure 10).
To create a configuration, either double click ‘Comms Hub Configuration’ or, start by selecting ‘Comms Hub Configuration’, and then either right-click and select ‘New’ or, select the first icon on the dialog’s toolbar (New). The outcome should be similar to Figure 11.

Three tabs are available for configuration: Settings, ZigBee Network and Devices. As will be seen some elements are mandatory, and some will be filled with default values which can be overridden. Also, a meaningful name should be assigned to each configuration created, so it is easily identified. In this case, the name chosen was ‘CH-ESME’.
This dialog can be used to manage configurations but also to start the Communications Hub (using the ‘Start’ button). The ‘Start’ button will only be enabled if the selected configuration is valid (all mandatory fields are filled with valid values).

An error message will be displayed when data is missing or not valid and the tab(s) containing the offending field(s) will be highlighted with an error icon.

Next, all tabs and their fields will be reviewed as the configuration is created.

**Settings**

This tab features the following elements:

- **CHF Id**: CHF’s Entity Identifier. Defaults to 00db1234567890A2;
- **GPF Id**: GPF’s Entity Identifier. Should match the ZigBee’s adaptor MAC Address;
- **CHF Port**: The TCP port used by the GFI testing tool to exchange GBCS messages with the CH. Defaults to 50001;
- **CHF Control Port**: TCP port used to send commands to the CH. It is used by the GUI to fetch the list of devices in the HAN. Defaults to 50000;
- **Log Level**: The level of verbose the CH will output to its log file. The default value is 4.4

![Settings page complete](image)

---

3 Note that although the values entered may be within the range of acceptable values, the Communications Hub might still fail to start if the data provided is not consistent with the environment (e.g. the GPF does not match the ZigBee dongle MAC address).

4 This value is not written to the configuration file – it is a parameter and as such will not be saved if the configuration is exported.
After entering the missing configuration (GPF Id), the dialog should look similar to Figure 12. Note that the 'Settings' tab is now cleared of errors, and the error message changed, mentioning a field present in the 'ZigBee Network' tab (in this case 'Pan Id'). Next, the ZigBee Network tab will be reviewed.

**ZigBee Network**

This tab features the following fields:

- **Pan ID**: The ZigBee network Pan ID.
- **Extended Pan ID**: The ZigBee network extended Pan ID.
- **Network Key**: The ZigBee network key.
- **Port**: The USB port where the ZigBee Adapter is connected. Defaults to /dev/ttyUSB0.
- **Channel**: The chosen channel (ranging from 11 to 26) and
- **Join Timeout**: The join timeout, specified in seconds. Defaults to 255.

![Figure 13 – Zigbee Network complete](image)

After entering the missing elements, the dialog should look similar to Figure 13. The 'Start' button should now be enabled as this configuration is a valid one. As such, the CH could be started by selecting 'Start'. Instead, the 'Devices' tab will be introduced. This tab allows the CH's whitelist to be populated upon start-up. As the test sequence that will be created later features an ESME, this device will be added to CH's whitelist (and as a consequence, being the only device in the HAN able to communicate).

---

5 An alternative to doing this would be to add this device to CH's whitelist via a CCS01 test-case.
Devices
Via this dialog, it is possible to Add, Change and Remove device information from the CH’s whitelist. The information required to configure a device is:

- **Device Id**: Entity Id of the device;
- **Device Type**: Choice between ESME, GSME, HC ALCS, IHD, CAD and PPMID and
- **Network Credentials**: a 128-bit install code.

Figure 14 and Figure 15 illustrate this process.

Now that the configuration is ready, it should be saved selecting ‘Apply’.
Every time a configuration is changed it is possible to revert the changes as long as ‘Apply’ was not selected, by selecting ‘Revert’. ‘Close’ would close this dialog but, as there are changes pending, a confirmation dialog would be displayed, allowing the user to save the configuration.

**Starting the Communications Hub**

To start the Communications Hub, select ‘Start’. The console pane should display the outcome of this operation (Figure 16).

![Figure 16 – Outcome of CH’s start command shown in the console](image)

The Communications Hub was started successfully.

Every time an attempt is made to start the Communications Hub, the configuration used is added to the menu. The next time the ‘Configure/Start Comms Hub’ button is pressed, it will attempt to start the last configuration. To display the CH configurations management dialog, the user must now press the ‘arrow’ button (Figure 17).

![Figure 17 – The CH Configurations’ menu](image)

**Displaying the Communications Hub Status**

The GUI does not maintain the CH’s state – instead, it can query it at any time to obtain its status. To do so, select the ‘Comms Hub Status’ button on the toolbar (Figure 7). Again the outcome is displayed in the console pane (Figure 18).

![Figure 18 – Outcome of CH’s ‘status’ command](image)

**Communications Hub Log File**

So far we have explored some of the CH’s actions and we have seen the outcome of these actions being displayed in the console pane. The CH writes entries to a log file whenever certain events happen (depending on the log level chosen). It is possible to display these events as they happen after a moment in time, which is triggered by the user by pressing a button. In other words, every new event will be
displayed until the user deactivates this feature again\(^6\). To activate this feature, the user must select the ‘Start/Stop CH Log’ in the toolbar (refer to Figure 7. The outcome is shown in Figure 19).

![Figure 19 – CH Logging started](image)

Figure 19 – CH Logging started

Notice that the title has changed from ‘Communications Hub’ to ‘Communications Hub Log’. Although the same pane (Console) is used to display this information, it is displayed in a different page. To select with page to display, the appropriate button in Figure 20 must be used.

![Figure 20 – Buttons to switch between console pages](image)

Figure 20 – Buttons to switch between console pages

Displaying the devices in the HAN

By selecting the ‘Fetch Han Log Entries’ button from the toolbar, the user can display the devices present in the CH’s whitelist. Since an ESME was added during the creation of this configuration, we can now confirm its presence in the CH’s whitelist (Figure 21).

\(^6\) Users of the UNIX OS will recognise this feature as the command tail –f.
Notice the change in the console’s page (back to the Communications Hub), as this is the outcome of a CH action. If the ‘Communications Hub Log’ is now displayed, the log has now entries corresponding to this command (Figure 22).

Everytime the user wishes to activate a different configuration, they must first bring the CH down (via the ‘Stop Comms Hub’ button from the toolbar).

This concludes the first task – CH configuration. The CH is now configured and ready to answer to requests. The next sections will deal with test creation and execution.

4.3.2 Test Creation Wizard

Assuming we start with an empty workspace, we will now create a 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, and 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 24.

![Figure 24 – New Test Properties](image)

**Scenario**

This dialog gathers the information required by the test scenario such as the type of the active meter, its id and certificates and CHF/GPF data. From these inputs, the file referred to as `<scenario-file>.xml` (Section 2.3.1) will be generated.

If the test uses full meter emulation only the type of meter is available for selection.

![Figure 25 – Scenario selection (emulators)](image)

If a physical (non-emulated) metering device is used, the following settings need to be entered:
ID: Entity Identifier of the active meter in CHF’s whitelist.

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

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

CHF Id, CHF Port and GPF Id: refer to section “ZigBee Network”.

The three latter fields are also used in the CHF’s boot up configuration – that’s the reason it makes sense to create one beforehand. Although the user can still enter these fields, it will be easier and less error-prone if that configuration is recalled. This can be done via the ‘Load From CH Conf’ button (Figure 27 and Figure 28).

---

7 Although the meter is identified in this step, it will not be added to CHF’s whitelist. To achieve this, either a CHF configuration containing this device must be used or otherwise it must be added later by sending a CCS01 command.
Initially, the fields in this dialog are shown blank (as in Figure 26) 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. Clicking ‘Next’ will lead to the next dialog, where the user will choose the test-cases to be added to the test.

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.
The type of active meter selected in the previous step will serve as filter for the test-cases available (i.e. selecting an Electric meter will display electric test-cases only). GFI 1.3.2E introduces CHF and GPF test-cases. CHF test-cases will always be available, whereas GPF’s will only be available if the active meter is a gas meter.

![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 30. 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.

---

8 Refer to the release notes for the particular cases when no meter is present and the user is only interested in testing the CHF/GPF.
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 4.3.5. 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 (4.3.3 - Edit Modes).

4.3.3 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 33 – GFI Preferences**

**Basic Mode**

In order to edit the Test Scenario, all the configurations detailed in the previous section (Figure 26 – 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 34 – 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 4.3.5 a practical example is presented.

Switching to Advanced Mode as in Figure 33 – 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.

4.3.4 Test Execution

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

Firstly, we will confirm the meter has successfully joined the HAN by querying the CH (‘Fetch Han Log Entries’). The outcome of this command shows the device authenticated successfully (Figure 38).

A Test previously created may be executed through the ‘Run’ button in the toolbar or the option ‘Run’ in the Test’s context menu.
A database of counters per Remote Party/Device 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 is 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.

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 40) 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 testing tool execution as if it would have been run through the command line. Notice the title has changed to ‘Test Run ESME-MPAN’ (Figure 42). This is yet another page in the same pane, corresponding to a third source of input: the output of running a test (or a campaign). As seen before, changing between pages in the console page can be achieved as shown in Figure 20.

When 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 47 depicts multiple executions of a Campaign.

4.3.5 Analysis of Results

In the previous section the test was executed without any changes to the original library test-cases, so as expected the result was PASS for all of them. Test-cases that read parameters from metering devices have been stripped of expected values (other than statuses or results that should be zero when messages are exchanged successfully) as it is not possible to predict what their values will be when 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.
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 39 – 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 50 needs to be changed. Note that the proper Iteration should be selected.

![Figure 50 – Edit Test Case Properties - before](image)

After assigning a different value to the parameter, the Iteration Detail should look like Figure 51.
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 55 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.

![Figure 53 – Test FAIL detail](image)

![Figure 44 – Metrics Information](image)

![Figure 55 – Change expected value](image)
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 56, 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.
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.

4.3.6 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 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.

![Figure 60 – Select output path](image)

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).

![Figure 61 – Exported files](image)

**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.

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).

**Duplicate Test**

This functionality works in a similar way to the ‘Import Test’ functionality. When invoked, either from the toolbar icon or from the context menu (mouse right-click over a test), the same dialog will be displayed, but the scenario and properties’ file path are already filled with the file paths from the test selected. The process is illustrated in Figure 63, Figure 64 and Figure 65. Notice that the user can still alter the file paths if they so wish.
Import/Export CH Configurations

As been shown in section 4.3.1, the GUI provides a convenient way of producing and managing CH configurations, as well as issuing commands to the CH. It is also possible to export the configurations produced to a XML file and loading a CH configuration file into the list.

Revisiting the CH configuration management dialog, and selecting the export configuration file icon (Figure 66), the save file dialog will be displayed (Figure 67).
The file name is chosen to be the name of the configuration plus the XML extension. Similarly, to re-import this file, and upon clicking the import icon in the toolbar, the file open dialog will be visible (Figure 68).

![Figure 68 – CH Import file open dialog](image)

A configuration will be created in accordance with the contents of this file (Figure 69). In this case it will be an exact copy of the one used to create it. The only value that may differ is the log level, which is not part of the CH configuration file. In this case, a value of 4 is assigned by default. The name generated is guaranteed to be unique.

![Figure 69 – Configuration imported from the XML file exported previously](image)
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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</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>GPF</td>
<td>Gas Proxy Function</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>UTRN</td>
<td>Unique Transaction Reference Number</td>
</tr>
<tr>
<td>ZCL</td>
<td>ZigBee Cluster Library</td>
</tr>
</tbody>
</table>

Table 3 – Acronyms.