Graphical User Interface Manual

GIT for Industry

Version:	**4.3.0**
Date:	**28th January 2021**
Author:	**Smart DCC Ltd.**
Classification:	**DCC Public**

Document Control

Revision History

Revision Date	Summary of Changes	Changes Marked	Version Number
29/10/2015	First version.	N/A	1.0RC4
27/11/2015	Update	N/A	1.0RC5
22/07/2016	Update	N/A	1.3.2E
08/08/2017	Update	N/A	1.3.8E
25/08/2017	Update	N/A	2.0.0 RC1
20/11/2017	Update	N/A	2.0.0 RC2
28/11/2017	Review	N/A	2.0.0 RC2
09/02/2018	Update	N/A	2.0.0
27/03/2018	Update	N/A	2.0.1
30/04/2018	Update	N/A	2.0.2
28/05/2018	Update	N/A	2.0.3
05/07/2018	ZigBee Silabs stack updated to version 6.2.3	N/A	2.1.0
09/10/2018	Update	N/A	2.1.1
16/01/2019	Update	N/A	2.1.2
06/03/2019	Update	N/A	2.1.3
24/04/2019	Update	N/A	2.1.4
17/06/2019	Update	N/A	2.1.5
06/09/2019	Update	N/A	2.1.6
13/09/2019	Update	N/A	3.0.0
25/10/2019	Update	N/A	3.0.1
14/11/2019	Update	N/A	3.0.2
09/01/2020	Update	N/A	3.1.0
31/01/2020	Update	N/A	3.1.1
13/03/2020	Update	N/A	3.2.0
23/04/2020	Update	N/A	3.2.1
08/05/2020	Update	N/A	3.3.0
19/06/2020	Update	N/A	4.0.0
26/06/2020	Update	N/A	4.0.1
09/07/2020	Update	N/A	4.1.0
24/07/2020	Update	N/A	4.2.0

Revision Date	Summary of Changes	Changes Marked	Version Number
31/07/2020	Update	N/A	4.2.1
12/08/2020	Update	N/A	4.2.2
18/08/2020	Update	N/A	4.2.3
31/08/2020	Update	N/A	4.2.4
10/09/2020	Update	N/A	4.2.5
15/09/2020	Update	N/A	4.2.6
18/09/2020	Update	N/A	4.2.7
06/10/2020	Update	N/A	4.2.8
26/11/2020	Update	N/A	4.2.9
28/01/2021	Update	N/A	4.3.0

Reviewers

Name	Title / Responsibility	Release Date	Version Number
Gonçalo Gouveia	Critical Software Project Engineer		1.0RC5
Karim Kanso	Critical Software Senior Engineer		1.0RC5
Dinis Paes	Critical Software Technical Manager		1.3.2E
Dinis Paes	Critical Software Technical Manager		1.3.8E
Dinis Paes	Critical Software Technical Manager		2.0.0 RC1
José Costa	Critical Software Project Engineer		2.0.0 RC2
Pedro Almeida	Critical Software GFI Team Leader	09/02/2018	2.0.0
Helder Martins	Critical Software GFI Team Member	27/03/2018	2.0.1
Helder Martins	Critical Software GFI Team Member	30/04/2018	2.0.2
Helder Martins	Critical Software GFI Team Member	28/05/2018	2.0.3
Helder Martins	Critical Software GFI Team Member	05/07/2018	2.1.0
Helder Martins	Critical Software GFI Team Member	09/10/2018	2.1.1
Helder Martins	Critical Software GFI Team Member	16/01/2019	2.1.2
José Rodrigues	Critical Software GFI Team Member	06/03/2019	2.1.3
Nuno Besteiro	Critical Software GFI Team Member	24/04/2019	2.1.4
Francisco Pinto	Critical Software GFI Team Member	17/06/2019	2.1.5
Helder Martins	Critical Software GFI Team Member	06/09/2019	2.1.6
Helder Martins	Critical Software GFI Team Member	13/09/2019	3.0.0

Name	Title / Responsibility	Release Date	Version Number
Helder Martins	Critical Software GFI Team Member	25/10/2019	3.0.1
Fábio Silva	Critical Software GFI Team Member	14/11/2019	3.0.2
Helder Martins	Critical Software GFI Team Member	09/01/2020	3.1.0
Helder Martins	Critical Software GFI Team Member	31/01/2020	3.1.1
Nuno Besteiro	Critical Software GFI Team Member	13/03/2020	3.2.0
Sara Alves	Critical Software GFI Team Member	23/04/2020	3.2.1
Sara Alves	Critical Software GFI Team Member	08/05/2020	3.3.0
Nuno Besteiro	Critical Software GFI Team Member	19/06/2020	4.0.0
Nuno Besteiro	Critical Software GFI Team Member	26/06/2020	4.0.1
Nuno Besteiro	Critical Software GFI Team Member	09/07/2020	4.1.0
Helder Martins	Critical Software GFI Team Member	24/07/2020	4.2.0
Nuno Besteiro	Critical Software GFI Team Member	31/07/2020	4.2.1
Francisco Pinto	Critical Software GFI Team Leader	12/08/2020	4.2.2
Helder Martins	Critical Software GFI Team Member	18/08/2020	4.2.3
Helder Martins	Critical Software GFI Team Member	31/08/2020	4.2.4
Francisco Pinto	Critical Software GFI Team Leader	10/09/2020	4.2.5
Daniel Oliveira	Critical Software GFI Team Member	15/09/2020	4.2.6
Francisco Pinto	Critical Software GFI Team Leader	18/09/2020	4.2.7
Daniel Oliveira	Critical Software GFI Team Member	06/10/2020	4.2.8
Francisco Pinto	Critical Software GFI Team Leader	26/11/2020	4.2.9
Francisco Pinto	Critical Software GFI Team Leader	28/01/2021	4.3.0

Approvals

Name	Signature	Title / Responsibility	Release Date	Version Number
António Alves		Critical Software Project Manager	27/11/2015	1.0RC5
Dinis Paes		Critical Software Technical Manager	25/07/2016	1.3.2E
Dinis Paes		Critical Software Technical Manager	08/08/2017	1.3.8E
Dinis Paes		Critical Software Technical Manager	25/08/2017	2.0.0 RC1
Dinis Paes		Critical Software Technical Manager	04/12/2017	2.0.0 RC2
Pedro Almeida		Critical Software GFI Team Leader	09/02/2018	2.0.0
Pedro Almeida		Critical Software GFI Team Leader	27/03/2018	2.0.1

Name	Signature	Title / Responsibility	Release Date	Version Number
Pedro Almeida		Critical Software GFI Team Leader	30/04/2018	2.0.2
Pedro Almeida		Critical Software GFI Team Leader	28/05/2018	2.0.3
Pedro Almeida		Critical Software GFI Team Leader	05/07/2018	2.1.0
Pedro Almeida		Critical Software GFI Team Leader	09/10/2018	2.1.1
Pedro Almeida		Critical Software GFI Team Leader	16/01/2019	2.1.2
Pedro Almeida		Critical Software Technical Manager	06/03/2019	2.1.3
Pedro Almeida		Critical Software Technical Manager	24/04/2019	2.1.4
Pedro Almeida		Critical Software Technical Manager	17/06/2019	2.1.5
Pedro Almeida		Critical Software Technical Manager	06/09/2019	2.1.6
Pedro Almeida		Critical Software Technical Manager	13/09/2019	3.0.0
Pedro Almeida		Critical Software Technical Manager	25/10/2019	3.0.1
Pedro Almeida		Critical Software Technical Manager	14/11/2019	3.0.2
Pedro Almeida		Critical Software Technical Manager	09/01/2020	3.1.0
Pedro Almeida		Critical Software Technical Manager	31/01/2020	3.1.1
Pedro Almeida		Critical Software Technical Manager	13/03/2020	3.2.0
Pedro Almeida		Critical Software Technical Manager	23/04/2020	3.2.1
Pedro Almeida		Critical Software Technical Manager	08/05/2020	3.3.0
Pedro Almeida		Critical Software Technical Manager	19/06/2020	4.0.0
Pedro Almeida		Critical Software Technical Manager	26/06/2020	4.0.1
Pedro Almeida		Critical Software Technical Manager	09/07/2020	4.1.0
Pedro Almeida		Critical Software Technical Manager	24/07/2020	4.2.0
Francisco Pinto		Critical Software GFI Team Leader	31/07/2020	4.2.1
Pedro Almeida		Critical Software Technical Manager	12/08/2020	4.2.2
Pedro Almeida		Critical Software Technical Manager	18/08/2020	4.2.3
Pedro Almeida		Critical Software Technical Manager	31/08/2020	4.2.4
Pedro Almeida		Critical Software Technical Manager	10/09/2020	4.2.5
Pedro Almeida		Critical Software Technical Manager	15/09/2020	4.2.6
Pedro Almeida		Critical Software Technical Manager	18/09/2020	4.2.7
Pedro Almeida		Critical Software Technical Manager	06/10/2020	4.2.8
Pedro Almeida		Critical Software Technical Manager	26/11/2020	4.2.9
Pedro Almeida		Critical Software Technical Manager	28/01/2021	4.3.0

Table of Contents

1 Introduction 8

1.1 Objective 8

1.2 Document structure 8

2 Overview 8

2.1 High Level Architecture 8

2.1.1 VSIS-Core 9

2.1.2 Simulation 9

2.1.3 GFI-Testing-Tool 9

2.1.4 Test Library 10

2.1.5 Messages Database 10

2.1.6 GFI-Testing-Tool GUI 10

2.1.7 Communications Hub 10

2.2 Reference Testbed 11

2.3 Inputs and Outputs 13

2.3.1 Test Scenario File 13

2.3.2 Test Properties File 14

2.3.3 Test Report Files 14

2.3.4 Execution Log 14

2.3.5 Communications Hub Configuration File 14

3 Environment Start-up 14

3.1 ZigBee HAN network connection 14

3.2 System Start-up 15

3.3 Running the GFI Standalone Application 15

4 GFI Testing Tool GUI Operation 15

4.1 Running the GFI Testing Tool application 16

4.2 Work Environment description 16

4.2.1 Explorer 16

4.2.2 Console / Log Output / Report 17

4.2.3 Edit / View Pane 18

4.2.4 Toolbar 18

4.3 Step-by-step Example 19

4.3.1 GFI Communications Hub Configuration 19

4.3.2 Test Creation Wizard 29

4.3.3 Edit Modes 37

4.3.4 Test Execution 40
4.3.5 Analysis of Results 44
4.3.6 Other Features 50
5 Glossary. 65

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 anyone wishing 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 v4.0.0 supports all GBCS v4.0 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 its 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.

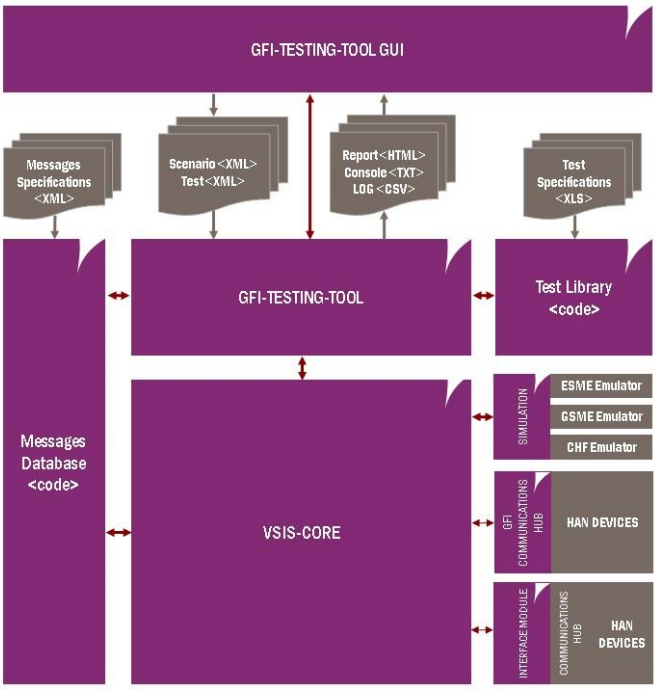


Figure 1 – GFI high level architecture.

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 in three different types of scenarios:

- Fully-emulated (all devices are emulated),
- Testing physical devices over a ZigBee network using the GFI communications hub module, and
- Testing physical devices using an external physical communications hub via an Interface Module.

The Simulation Manager is responsible for routing 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 the Core, the Simulation Manager, the 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.

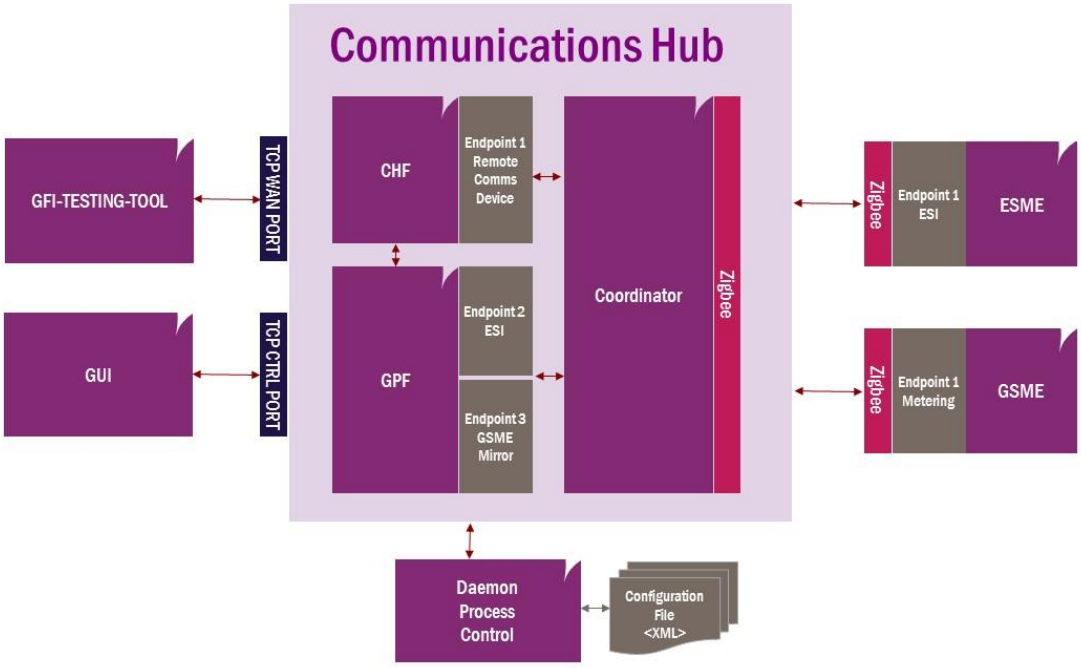


Figure 2 – Communications Hub's Architecture

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

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 for which it is the target device (e.g. CCS01), 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 to 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 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 three modes of execution: **fully-emulated**, using the **GFI Communications Hub** module and using an external Communications Hub via an **Interface Module**.

When working in its fully-emulated form, all devices (e.g. ESME, GSME, SAPC, CHF, GPF, PPMID and HCALCS) and remote parties (e.g. Supplier, ACB) are emulated. The focus is on validating the protocols rather than the use-case functionality (albeit in some cases some minimal functionality is offered). No

state is maintained between different tests. Control parameters are sometimes used to simulate state or meter behaviour[2] (e.g. meter behaving and responding like a single/twin element, triggering error responses, triggering alerts, etc.).

When the GFI Communications hub module is used, GFI acts as a client to a proper implementation of a communications hub, which is connected to a USB ZigBee radio dongle, serving (amongst other things) as the entry point to a HAN ZigBee network where physical devices can be connected.

When using the External communications Hub mode of operation, an interface module (usually built and used by a Communications Hub manufacturer) is used to send the GBCS payloads to a physical Communications Hub. This interface module acts as an adapter between GFI and the physical Communications Hub. The adapter provides a generic sockets-based interface to GFI, receiving the GBCS payloads and forwarding them to the physical Communications Hub.

Generally, messages from remote parties are routed appropriately[3] 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: As mentioned, when working in fully emulated mode, the GFI testing tool only performs conformance tests against the GBCS v4.0 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.

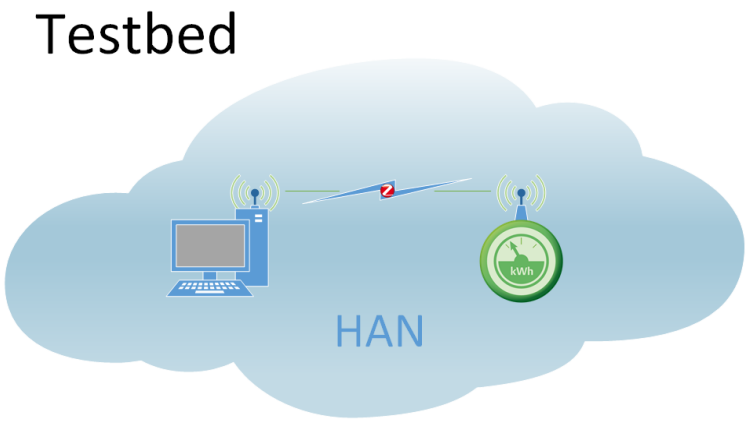


Figure 3 – GFI Testbed diagram[4]

[2] The control parameters will not be propagated to real meters over the ZigBee network.
[3] The appropriate infrastructure will route the messages, depending on the scenario considered – within GFI testing tool for emulated scenarios, or via GFI Communications Hub/IM for real devices.
[4] This is the testbed considered later when showing the GUI's features. The IM mode scenario configuration will be considered in a different section.

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.

File	I/O	Type	Description
<scenario-file>.xml	Input	Configuration file	The scenario configuration file required by the tool. Specifies in which mode GFI will run, which equipment to use, the device configurations, keys, certificates, among other configurations.
<properties-file>.xml	Input	Test execution file	The execution file required by the tool. Specifies which Test Cases should be executed and the inputs for each Test Case.
<test-report>.html	Output	Test report file	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.
<execution-log>.csv	Output	Execution log file	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.
<ch_conf_*timestamp*>.xml	Input	Communications Hub configuration file	An XML configuration file which the CH loads upon booting up[5]. Defines CHF and GPF's general settings, ZigBee radio configuration and the addition 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

[5] Only applicable when using the GFI Communications Hub.

be executed in a scenario with real ESME/GSME devices or in a scenario with emulated devices. This configuration is defined in this file by specifying the desired mode of operation (2.1.2 Simulation)

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 execution of a Test, a formatted HTML report file is generated featuring all the sets, checks, action prints and exchanged messages in 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 GFI 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, security options 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 GFI Communications Hub 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 GFI 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 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 18.04. 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.

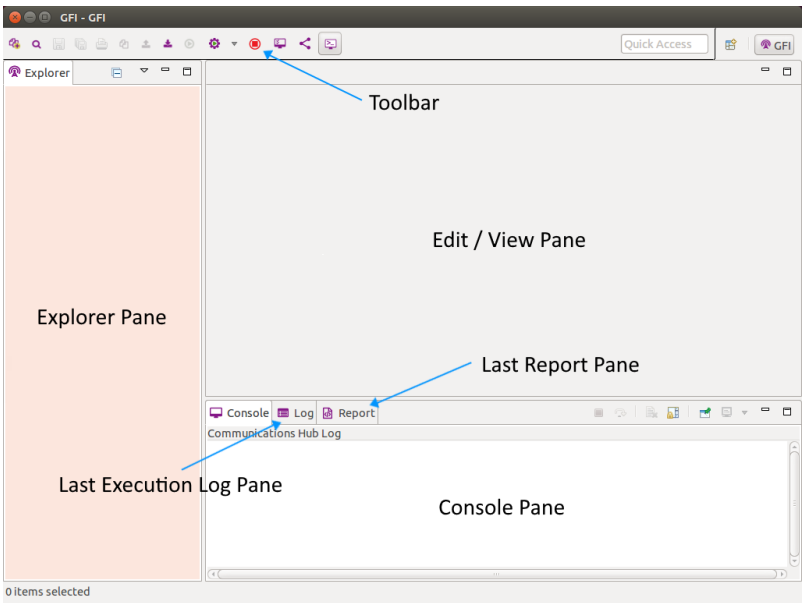


Figure 4 – GFI Testing Tool GUI

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 therefore 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 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.

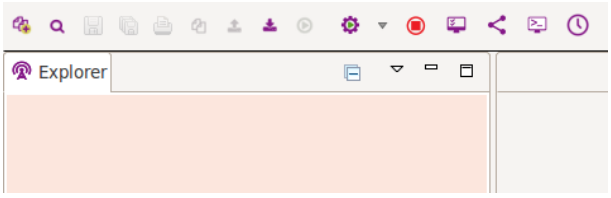


Figure 5 – Explorer

See also Figure 55 – Multiple Runs for a depiction of Tests and Campaigns in the application's Explorer pane.

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 therefore not editable.

Collapse Button

On the Explorer Pane there is a button to allow the user to collapse all Tests and Campaigns.

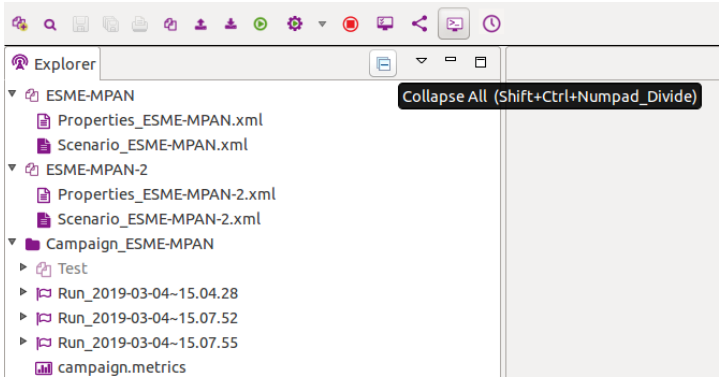


Figure 6 – Panel Before Clicking on Collapse Button

If there is any Test or Campaign open, the user can click the Collapse button to collapse it.

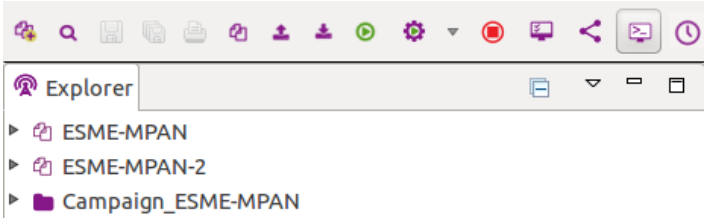


Figure 7 - Panel After Clicking on Collapse Button

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 GFI Communications Hub and the live contents of the GFI Communications Hub log file. 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 later sections.

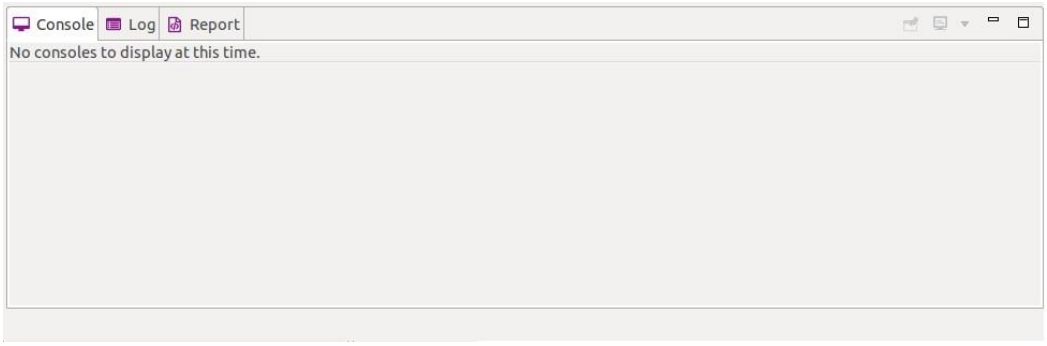


Figure 8 – Console

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. The execution output reports and the log files may also 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 57.

4.2.4 Toolbar

The toolbar provides an 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 has been selected).

Because the GUI can control the GFI Communications Hub module introduced in GFI1.3.2E, a specific subset of the toolbar is dedicated to this purpose[6]. Figure 9 highlights these actions, which will be explored in detail in the step-by-step example.

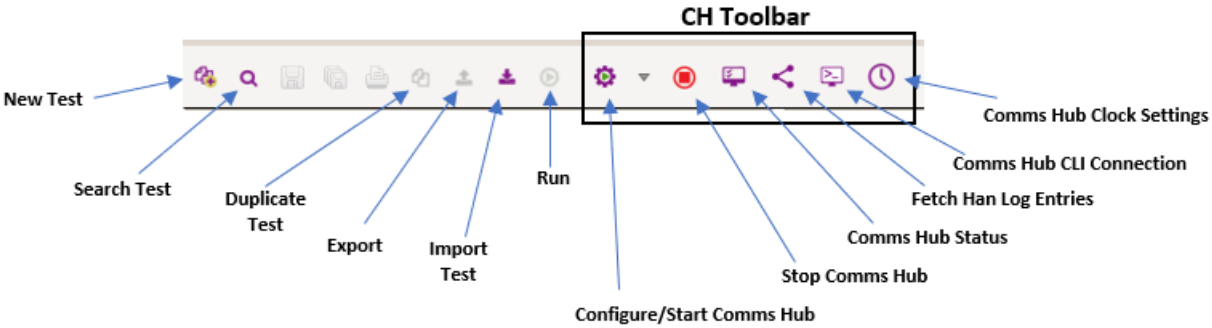


Figure 9 – The Toolbar

[6] Most of the actions are also available by right-clicking an item or via a menu option. It's not the case of the GFI 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 or an interface module) 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 GFI Communications Hub mode setup and as such the first step will be to configure and start the GFI 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 GFI Communications Hub Configuration

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

- The 'Configure/Start GFI Communications Hub' icon in the toolbar (refer to Figure 9) or
- Selecting the arrow next to it and then choosing 'Configure/Start GFI Comms Hub Configurations…' (Figure 10 and Figure 11).

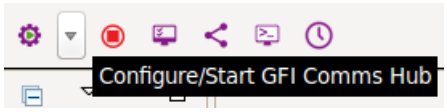


Figure 10 – Selecting the arrow next to the 'Configure/Start GFI Comms Hub' button

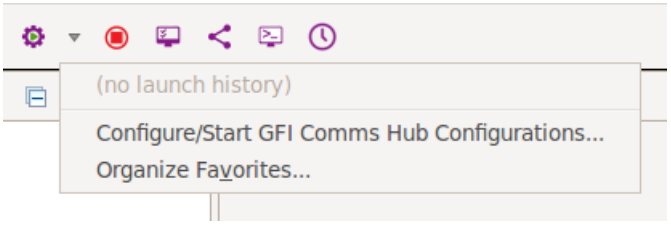


Figure 11 – Selecting the appropriate option from the menu

The GFI CH configuration management dialog should then be visible (Figure 12)[7].

[7] When using a workspace created with an earlier version of this GUI, existing configurations may be available, but they may lack values for some of the new parameters introduced in version 2.0.0. If this is the case, the configuration will need reviewing before it can be used.

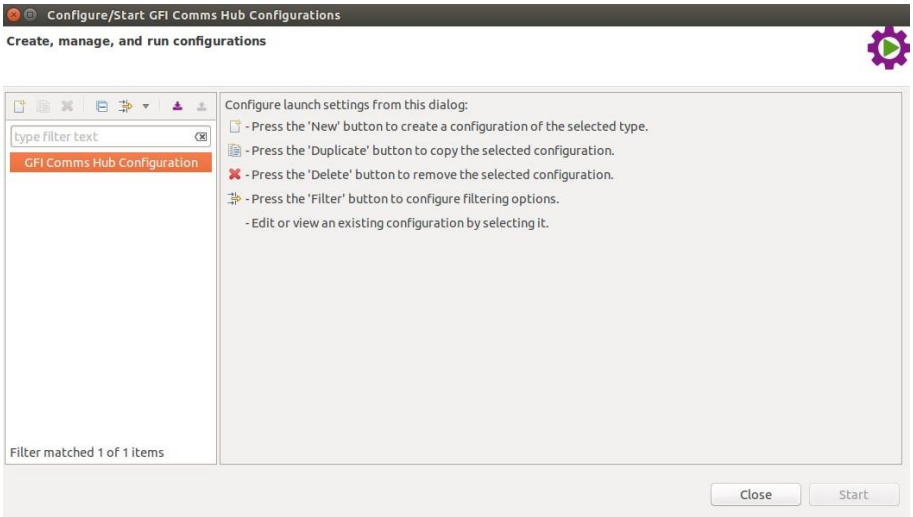


Figure 12 – CH Configuration Management Dialog

To create a configuration, either double click 'GFI Comms Hub Configuration' or, start by selecting 'GFI 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 13.

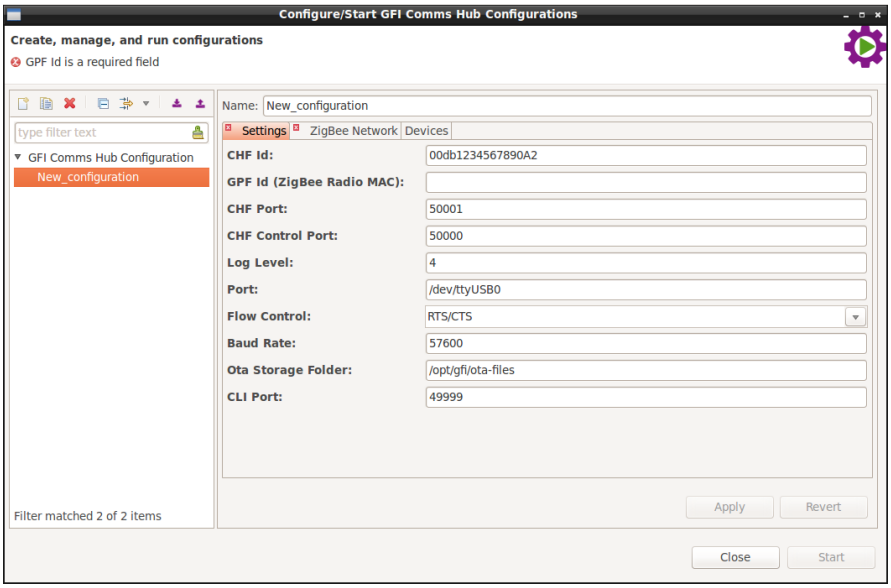


Figure 13 – New Configuration

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. A meaningful name should be assigned to each configuration created, so it is easily identified. In this case, the name chosen was 'CH'.

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 adaptor's 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.[8]
- Port: The USB port where the ZigBee Adapter is connected. Defaults to /dev/ttyUSB0.
- Flow Control: Choices are Xon/Xoff and RTS/CTS.
- Baud Rate: a value between 300 and 3000000. Defaults to 57600.
- OTA Storage Folder: Path to the folder that contains the OTA-files to the devices. Defaults to /opt/gfi/ota-files.
- CLI Port: The TCP port used to CLI connection.

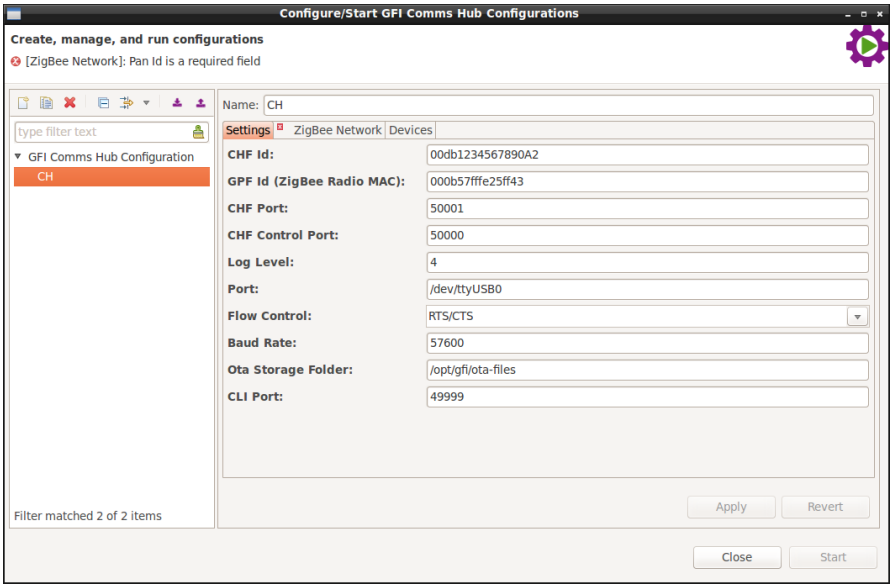


Figure 14 – Settings page complete

After entering the missing configuration (GPF Id), the dialog should look similar to Figure 14. Note that the 'Settings' tab is now cleared of errors, and the error message changed, mentioning a field present in

[8] 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.

the ‘ZigBee Network’ tab (in this case ‘Pan Id). Because of this, the ZigBee Network tab will be the next to 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.
- ZigBee Network Mode: The ZigBee network to be formed by the Comms Hub (e.g. only Sub-GHz). Default value is Dual (both SubGHz and GHz).
- Sleep End Device Timeout: The threshold which if exceeded will cause a sleeping device (e.g. GSME) to be removed from the parent table. Default value is 5 minutes
- 2.4 GHz Channel: The chosen GHz channel (ranging from 11 to 26)
- GHz Radio Tx Power: The GHz radio transmission power, in dBm.
- Sub-GHz Frequency Band: The chosen sub-GHz Frequency Band. Possible choices are 863-876 MHz and 915-921 MHz.
- Sub-Ghz Channel: The chosen sub-GHz channel. The values available will depend on the frequency band chosen in the previous step – for 863-876 MHz, the valid interval will be 0 to 48, whereas for 915-921 MHz, it will be from 0 to 12.
- Sub-GHz Radio Tx Power: The Sub-GHz radio transmission power, in dBm.
- Join Timeout: The join timeout, specified in seconds. Defaults to 255.

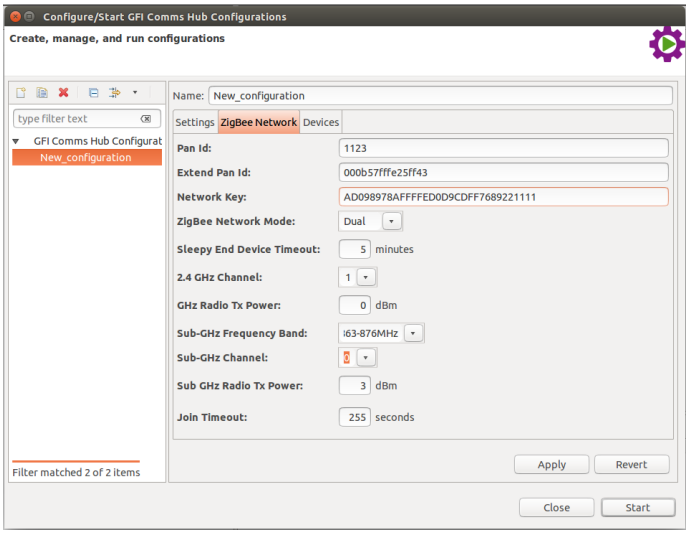


Figure 15 – ZigBee Network complete

After entering the missing elements, the dialog should look similar to the one in Figure 15. 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 now be introduced. Devices present in the table in the 'Devices' tab will populate the CH's whitelist 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, will be the only device in the HAN able to communicate9).

Devices

Using 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/SAPC, GSME, HCALCS, IHD, CAD and PPMID and
- Network Credentials: a 128-bit install code.

Figure 16 and Figure 17 illustrate this process.

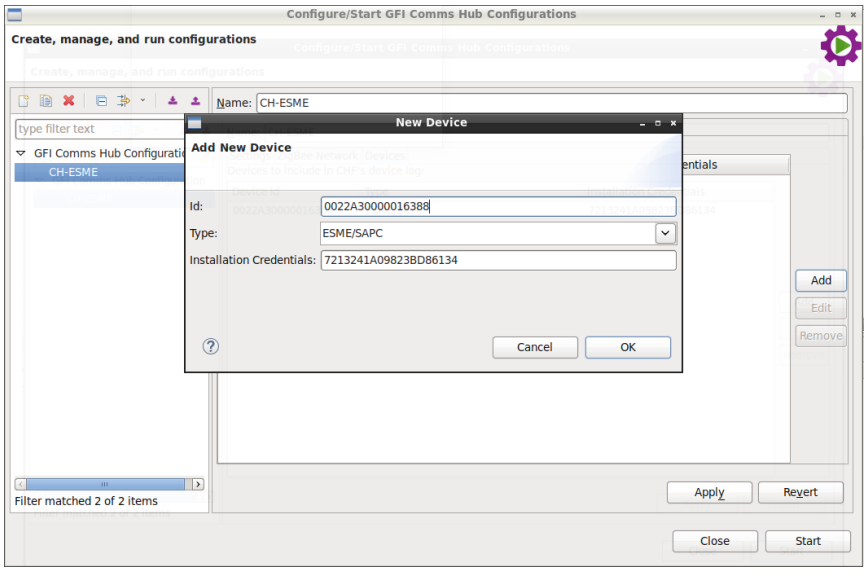


Figure 16 – New Device dialog

[9] An alternative to doing this would be to add this device to CH's whitelist via a CCS01 test-case.

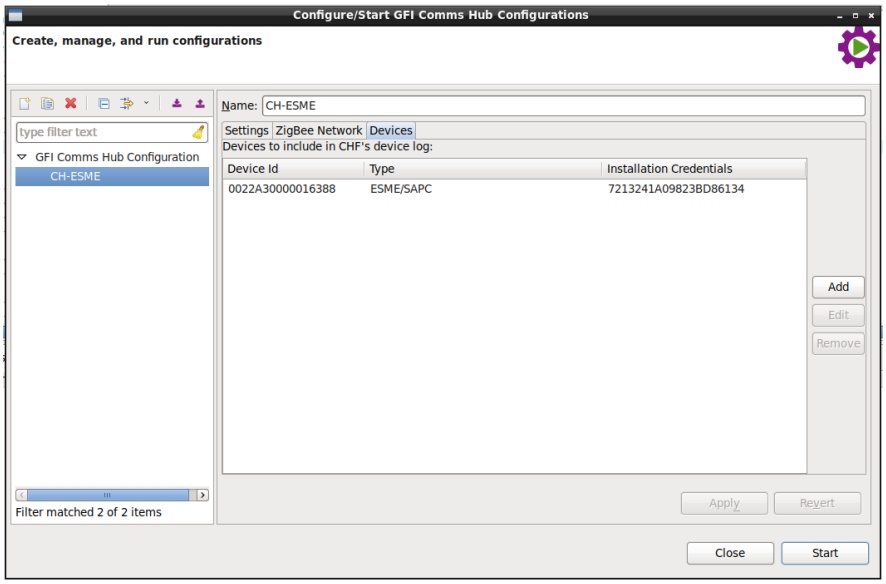


Figure 17 – Devices page featuring an ESME/SAPC

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 following dialog (see Figure 18) allows the user to select the CH start type.

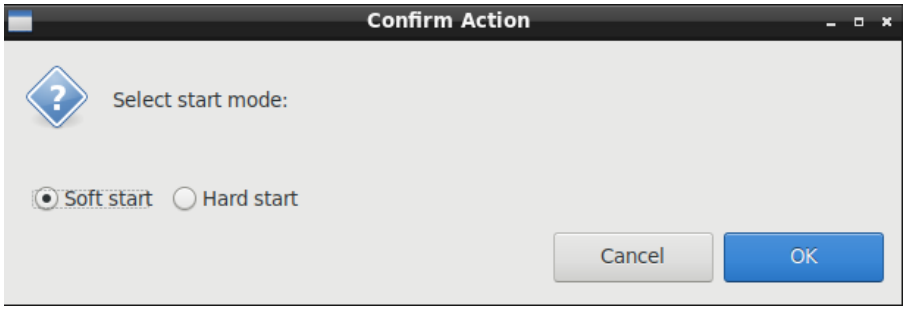


Figure 18 - CH Start Type Dialog

To run the CH with soft start mode, the user needs to have a previous run for that configuration. The previous run also needs to be stopped with soft stop. These conditions will allow the user to restore the HAN.

With hard start mode, the CH will not have any previous run information, so it will not restore the HAN.

The console pane should display the outcome of this operation (Figure 19).

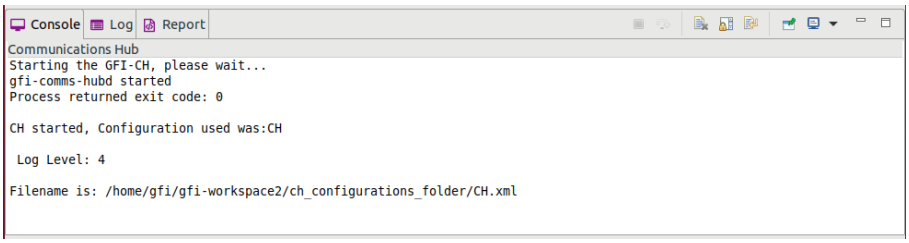


Figure 19 – Outcome of CH's start command shown in the console

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 GFI Comms Hub' button is pressed, it will attempt to start the last configuration used (soft start). To display the CH configurations management dialog, the user must now press the 'arrow' button (Figure 20).

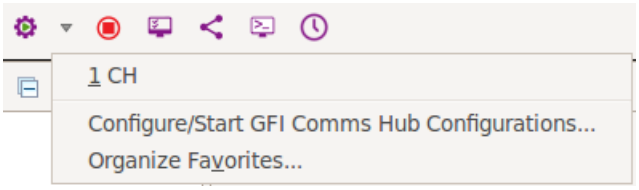


Figure 20 – The CH Configurations' menu

Stopping Communications Hub

Selecting the Stop Communications Hub button (Figure 21) a dialog box to choose the stop mode will appear (Figure 22).

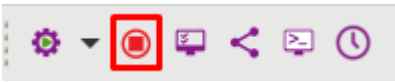


Figure 21 - Stop Communications Hub button

When the Soft stop mode is selected, the CH configuration and HAN information is saved, and the user can later start the CH with soft start to restore these configurations. If the Hard stop mode is selected, all HAN information will be deleted and hence, when starting the CH the Hard start mode shall be selected.

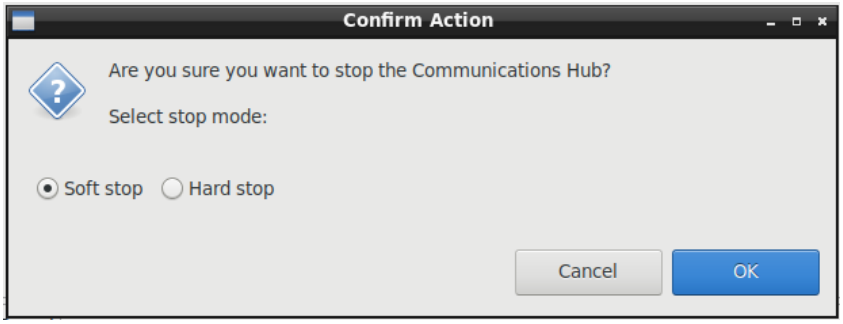


Figure 22 - Stop Communication Hub Dialog

Displaying the GFI 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 9). Again, the outcome is displayed in the console pane (Figure 23).

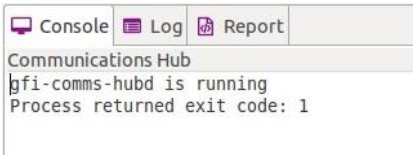


Figure 23 – Outcome of CH's 'status' command

GFI Communications Hub CLI Connection

By selecting the 'Communications Hub CLI Connection' button from the toolbar, the user can use the Comms Hub CLI.

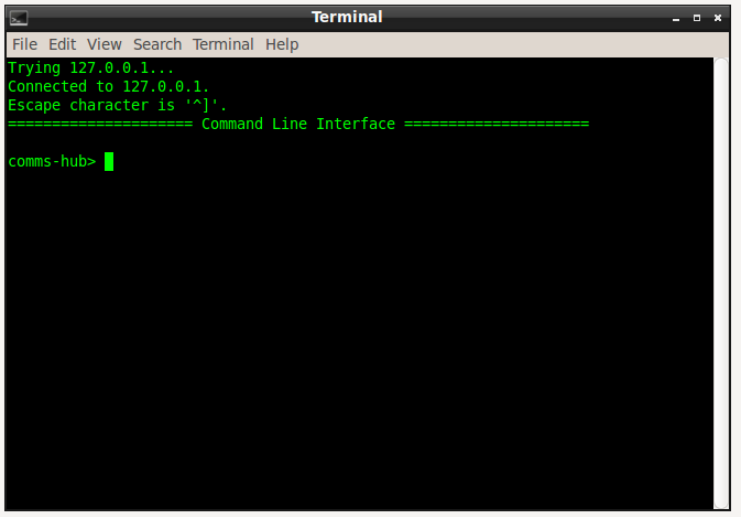


Figure 24 - Communications Hub CLI

To close the connection to the Communications Hub CLI the user is required to close the window, and the connection will terminate, but the Communications Hub will continue running.

The user can configurate the IP address and the Port to connect to the Communications Hub in the GFI Preferences.

GFI Communications Hub Clock Settings

By selecting the 'Communications Hub Clock Settings' button from the CH Toolbar, the user can change the date and time in the Communications Hub. The user has the option to sync the Communications Hub clock with the operating system or set a custom time.

To sync the Communications Hub Clock with the operating system the user needs to press the 'Sync With OS' button and the changes will be applied.

Using the 'Update From Comms Hub' button, the tool inserts the date/time from the Communications Hub in the date/time fields shown in Figure 22. The fields can also be changed by the user to set a custom time.

To apply the custom time, press the button 'Apply' and the changes will take effect in the Communications Hub.

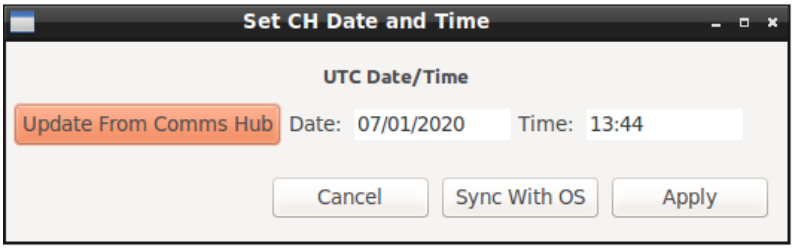


Figure 25 - Communications Hub Clock Settings

GFI 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. Depending on the log level chosen, the CH may log events to a log file. It is possible to display the contents of this file in real-time. This feature is active by default and the log activity is displayed as the CH is started. The outcome is shown in Figure 26.

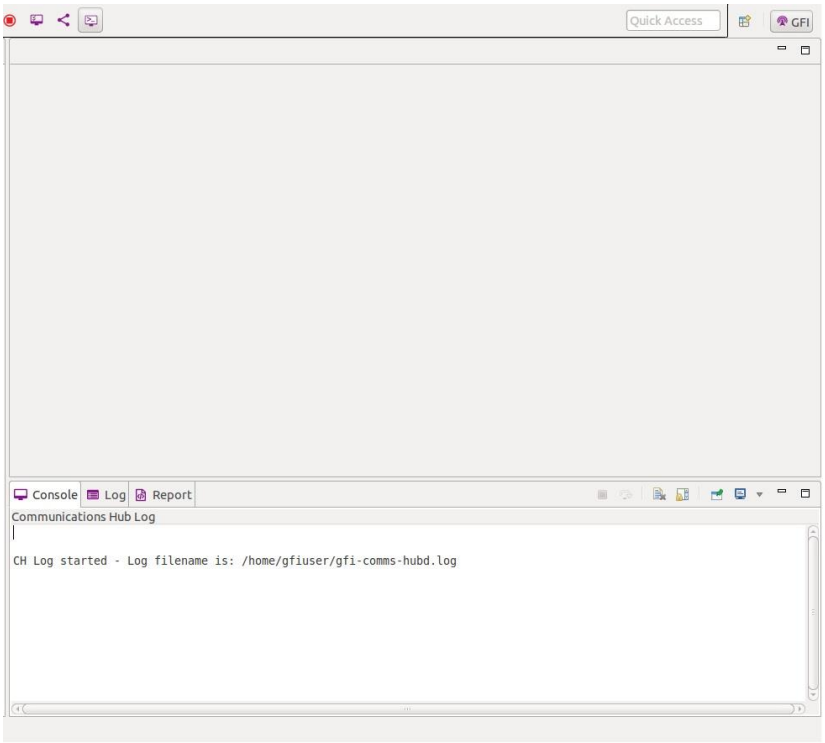


Figure 26 – CH Logging started

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 which page to display, the appropriate button in Figure 27 must be used.

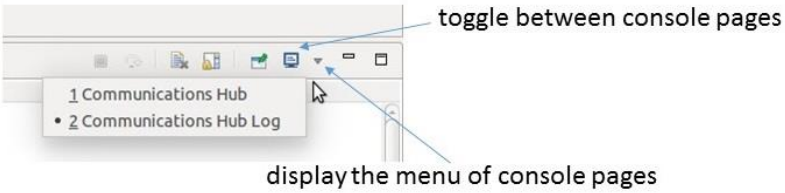


Figure 27 – Buttons to switch between console pages

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 28).

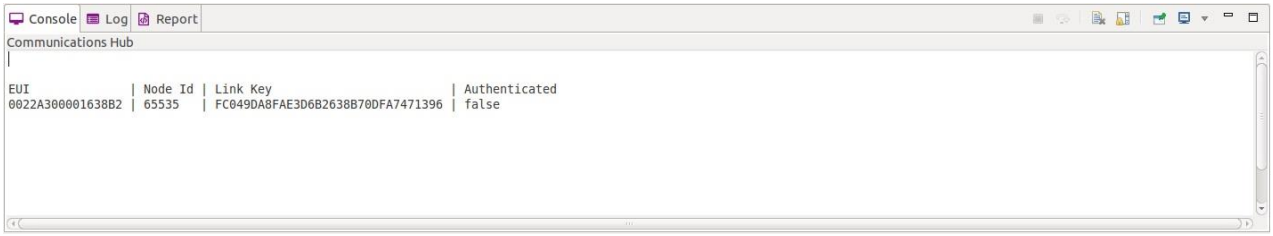


Figure 28 – Outcome of the ‘Fetch HAN Log Entries’ command

Notice the change in the console’s page (switching back to the Communications Hub page), as this is the outcome of a CH action. If the ‘Communications Hub Log’ contents is now shown, the log has now entries corresponding to this command (Figure 29).

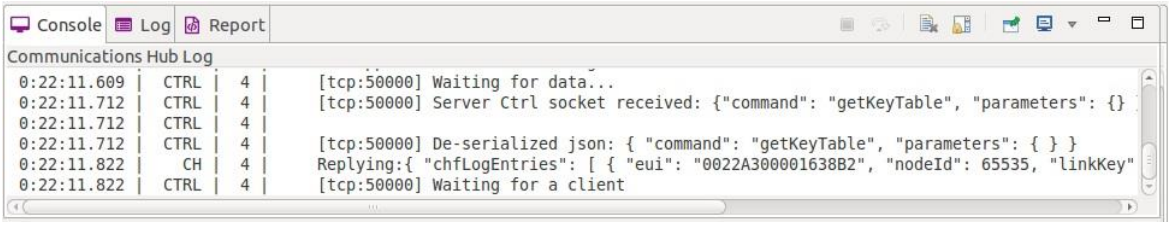


Figure 29 – CH’s log produced by ‘Fetch Han Log Entries’ command

Every time 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 requests. The next sections will deal with test creation and execution.

GFI Communications Hub LEDs behaviour

Table 2 specifies the expected behaviour of the GFI CH’s LEDs.

LED	Expected Behaviour
Yellow	Off: Not connected to USB or not recognized by the OS. On: Connected to USB and recognized by the OS.
Green	Blinking: Starting the network. Solid On: Network started. Off: Network is down (not started or stopped).
Red	Occasional blink: ZigBee traffic signal - sending/receiving ZigBee data. Blinking in a steady rate (~ 1blink/sec): There was a stack error, but it was probably recovered. Restart of the GFI CH is recommended and, in some cases, it may be required to function properly. Solid On: The ZigBee stack had an error and could not recover. Restart of the GFI CH is required. Off: No data is being exchanged.

Table 2 – GFI Communications Hub LEDs behaviour

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.

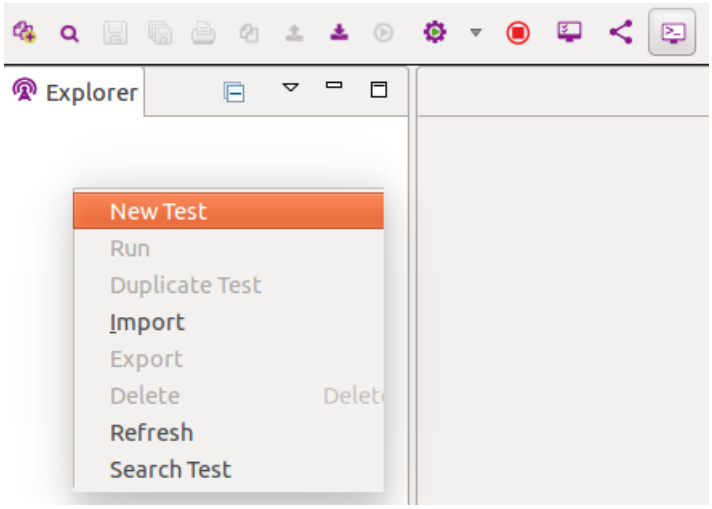


Figure 30 – New Test

New Test 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 (Figure 31):

- Identifier: This is the identifier that will be used for the test project name shown on the Explorer pane. The test folder will contain a file named after the concatenation of 'Properties' and the identifier entered.
- Test Name: A user-defined name, typically a short but meaningful description.
- Description: The detail used to include in the test purpose section of the test report (this field is not mandatory).

Figure 31 – New Test Properties

Comms Hub

The next two dialogues gather the information required by the test scenario such as the type of Comms Hub (and possibly also its configuration data) and the active device (and also possibly its id and certificates). 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 then no data is available to be configured, since the GFI's emulated scenario is assumed.

Figure 32 – Scenario selection (emulators)

If the GFI Comms Hub is selected, the following settings need to be entered:

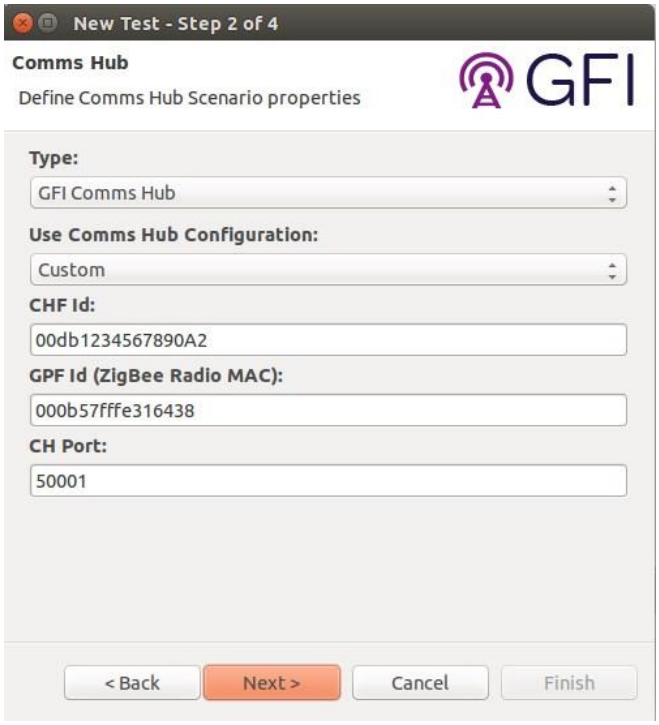


Figure 33 – Scenario selection (GFI Comms Hub)

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

These 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 manually, it will be easier and less error-prone if that configuration is recalled. This can be done via the “Use Comms Hub Configuration” selection box. (Figure 33).

If the Physical Comms Hub via Interface Module option is selected, the following fields need to be entered:

Figure 34 - Scenario selection (Physical Comms Hub via Interface Module)

- Module Address: The address of the interface module – either a hostname, an IPv4 address or an IPv6 address are valid,
- Module Port: the port in which the interface module listens for requests and
- Protocol: choices are TCP or UDP.

Regardless of the Comms Hub Type option selected, the fields will always be populated with data from the last known configuration, after a test is successfully created.

Clicking ‘Next’ will lead to the next dialog, where the user can choose the active device.

Devices

The Device window allows the user to choose the active device type.

The user can select one the following devices:

Figure 35 – Scenario selection (Emulated device)

- ESME, GSME, CHF, GPF, PPMID, HCALCS, SAPC

If the test is fully emulated, then no further data needs to be configured, as GFI's defaults are assumed. Otherwise, the following fields need to be filled[10]:

[10] If 'GFI Comms Hub' was chosen in step 2 and the device chosen is either the GPF or the CHF, then no further data needs to be entered – The values from step 2 will be used.

Figure 36 - Scenario selection (Device)

- ID: Entity Identifier of the active device in CHF's whitelist,
- Digital Signature Certificate: Path to the .der file containing the device certificate and
- Key Agreement Certificate: Path to the .der file containing the device certificate.

Data from the last configuration will also be recalled here, just like in the previous dialog. Clicking 'Next' will lead to the last 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 device selected in the previous step will filter the test-cases available (i.e. selecting an ESME will display ESME and CHF test-cases only[11]).

[11] The behaviour is consistent with earlier version of this GUI, namely the options 'electric' and 'gas'. It has been augmented with new devices but the legacy options are still available. For each new device is also possible to choose CHF test-cases.

Figure 37 – Search and Select Test Cases

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 38. Upon selection, the information of a test-case is displayed in the pane on the right 'Selected test case info'. Multiple selection is also possible.

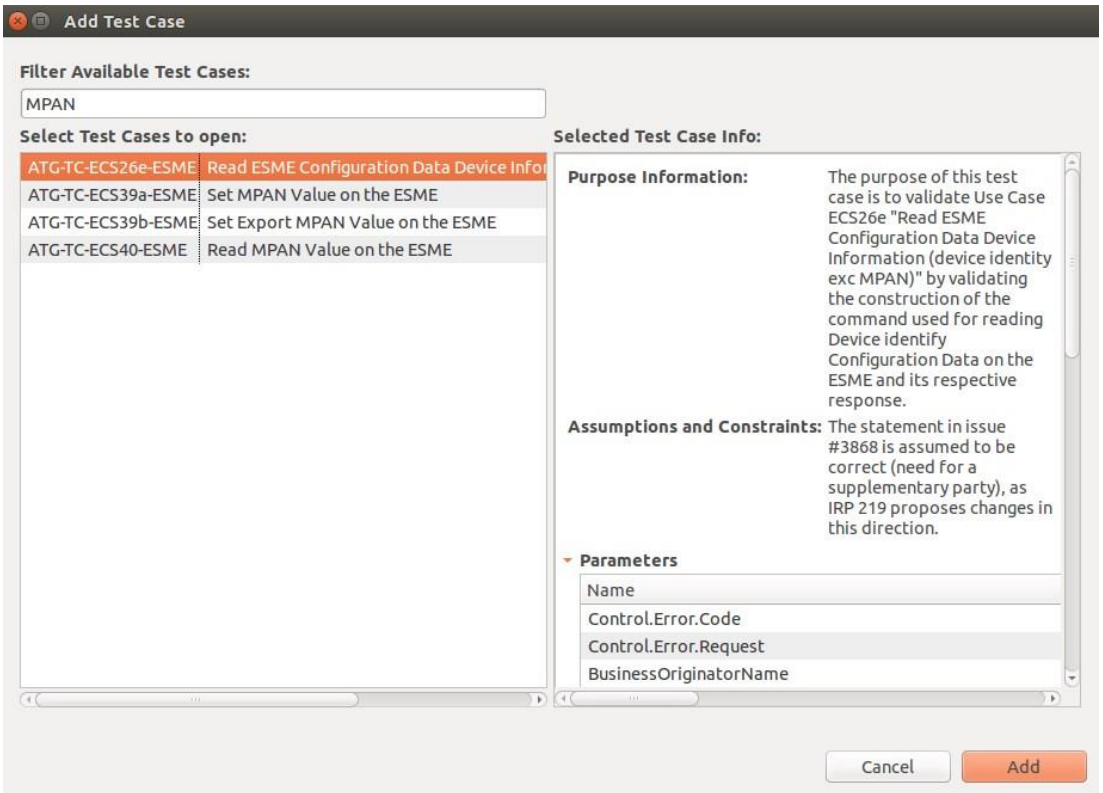


Figure 38 – Add Test Case

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 remove / 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.

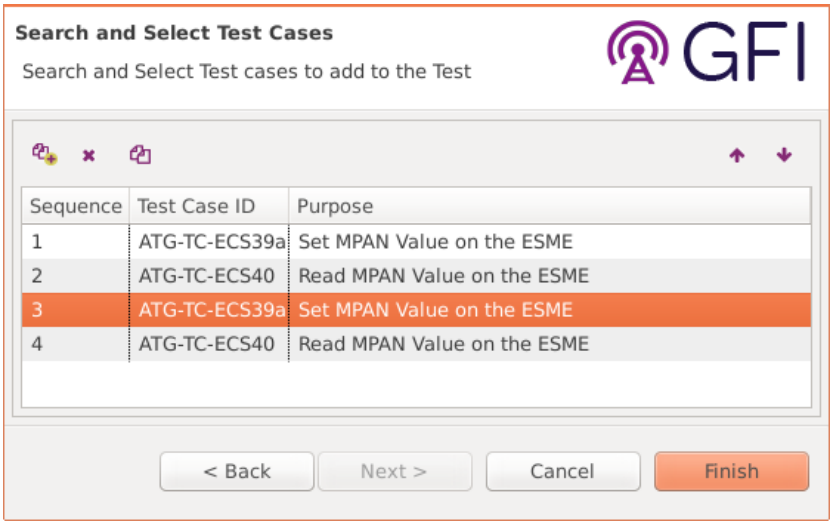


Figure 39 – New Test populated sequence

Clicking ‘Finish’ in this dialog will create the new test which will then be listed in the GUI’s Explorer window. 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).

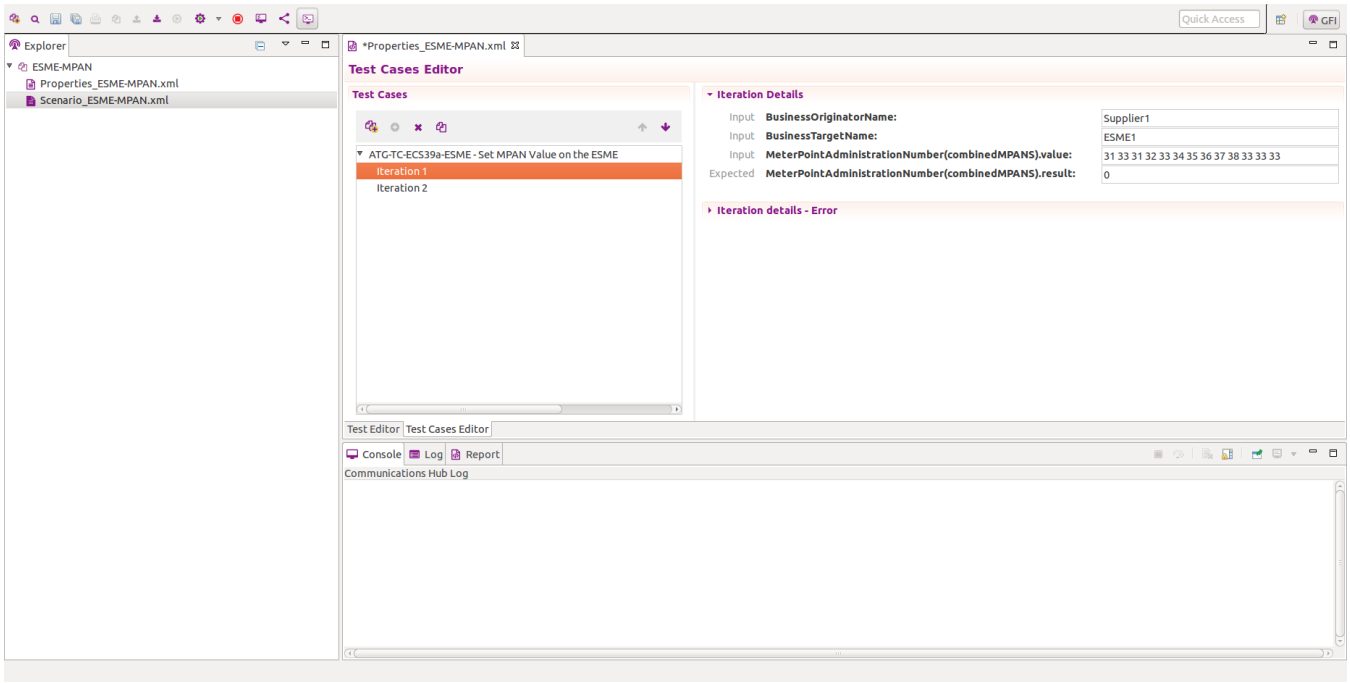


Figure 40 – New Test added

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 to edit Test / Test Case information as well as the addition (or removal) of further Test Cases and Iterations. The parameters' values, both Input and Expected may also 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 following the **Edit → Preferences** menu options. A dialog will be presented where this configuration may be altered.

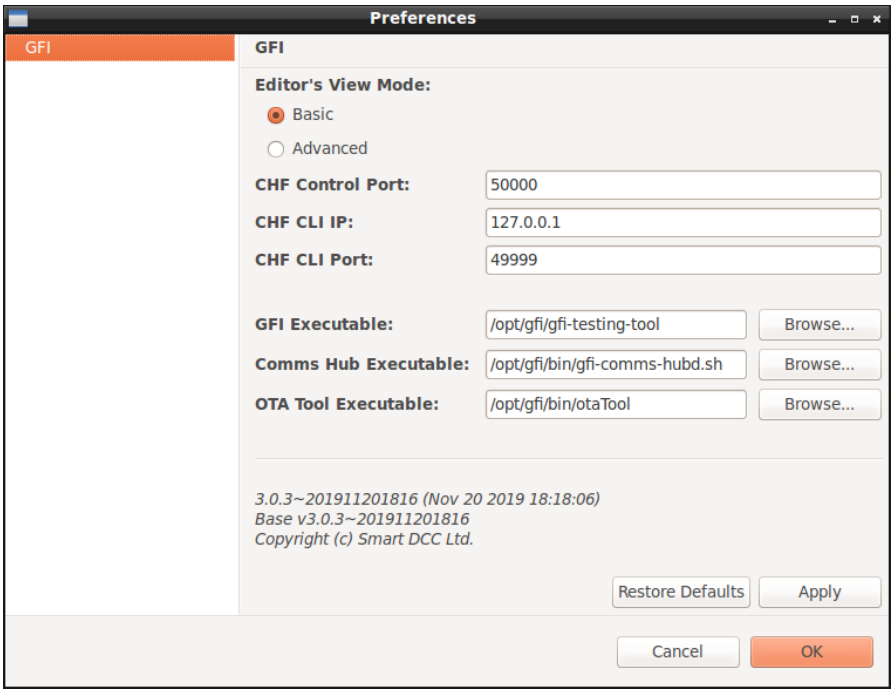


Figure 41 – GFI Preferences

Basic Mode

In order to edit the Test Scenario, all the configurations detailed in the previous section (Figure 33 – Scenario selection (GFI Comms Hub) and Figure 34 - Scenario selection (Physical Comms Hub via Interface Module) may be tweaked.

The Test Scenario editor also allows to configure the 'Force Message Corruption' option.

Figure 42 – 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;

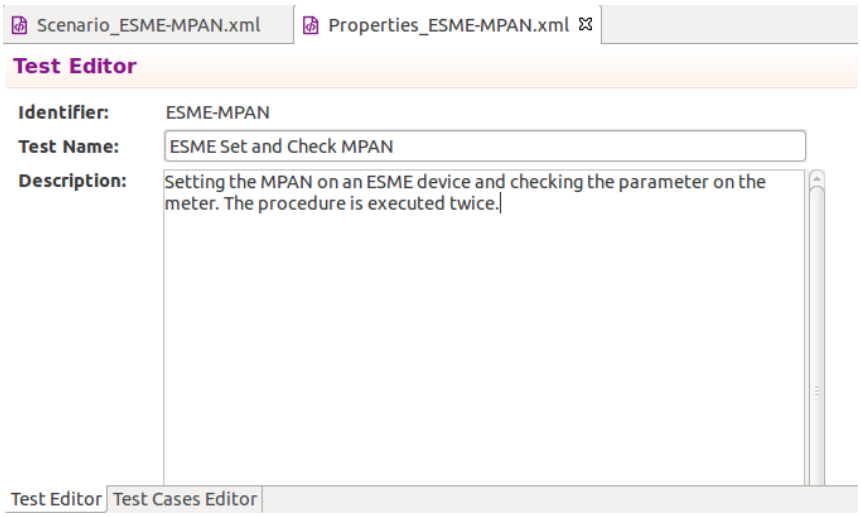


Figure 43 – Edit Test Properties

- 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.

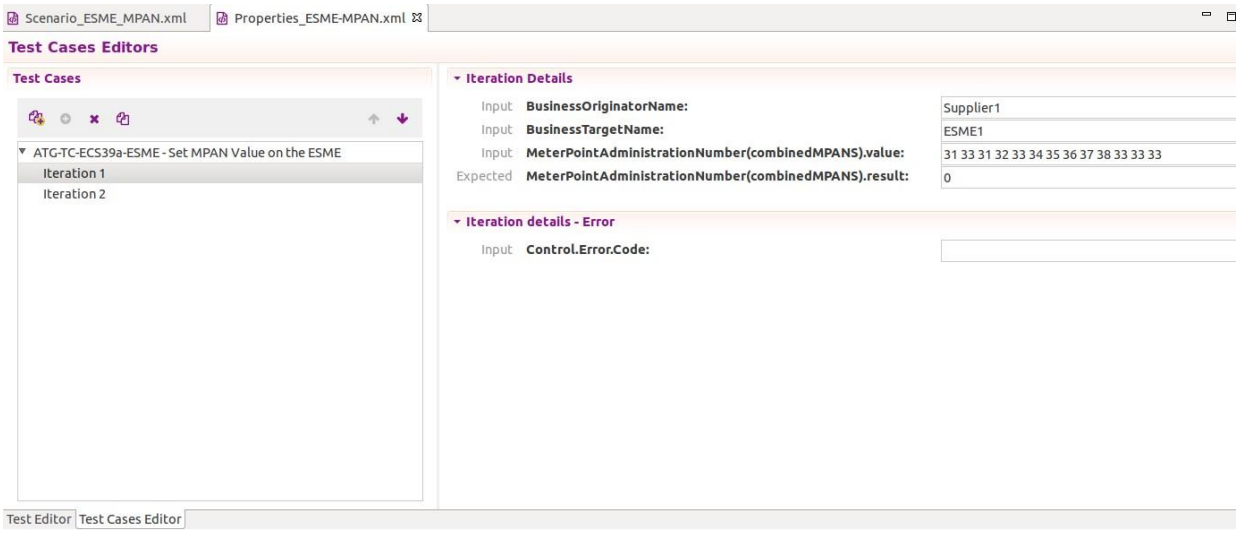


Figure 44 – Edit Test Case Properties

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 41 – 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.

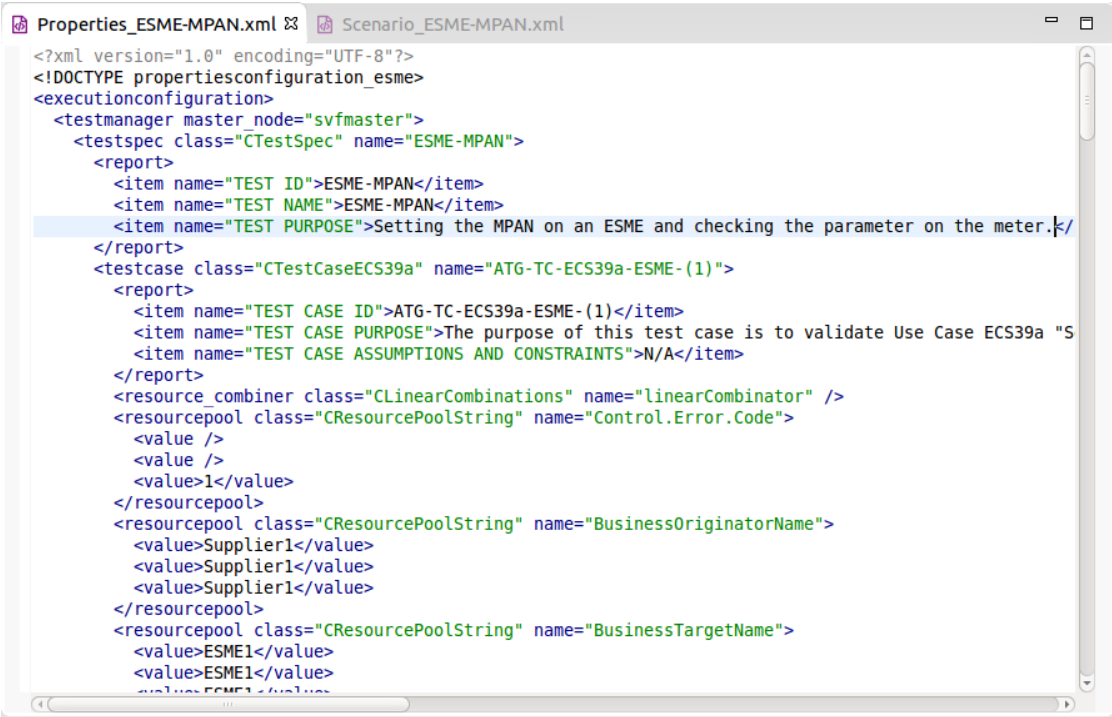


Figure 45 – Edit Properties in Advanced Mode

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 46).

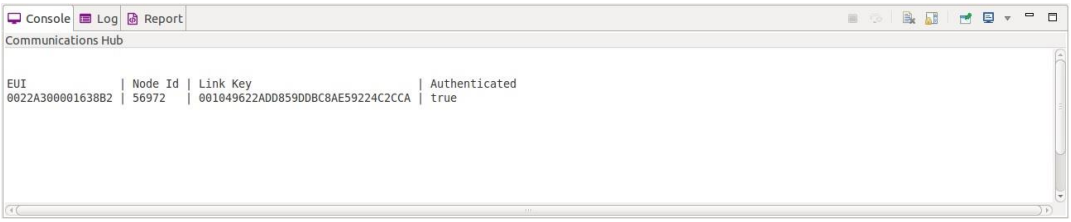


Figure 46 – 'Fetch Han Log Entries' showing the device authenticated

A Test previously created may be executed using the 'Run' button in the toolbar or the option 'Run' in the Test's context menu. For versions 2.0.0 and up it is now possible to run multiple tests in sequence. To do this, select the tests to be run (holding the control key while making the selection) before pressing the run button.

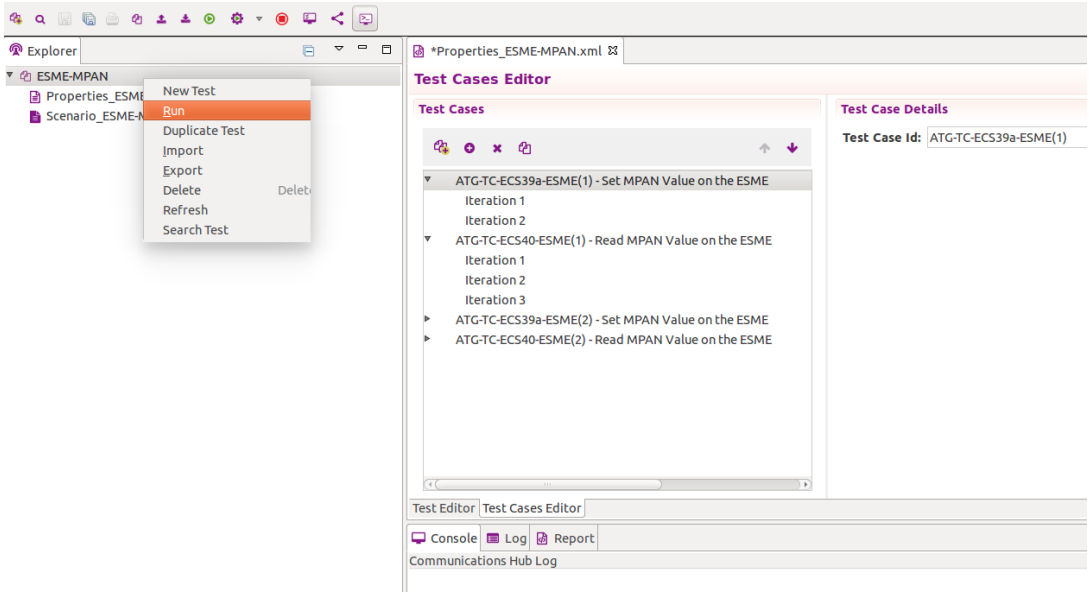


Figure 47 – Run Test from context menu

A database of counters per Remote Party 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 higher than the message's). Every time a message is sent, the counters used in message exchange are incremented and the total increment is stored in a counter database after an iteration is concluded.

If a given Remote Party is being used for the first time, their counters will not have yet been initialised in the database. If this is the case, a dialog is presented to obtain the Tester's confirmation of the counters starting values.

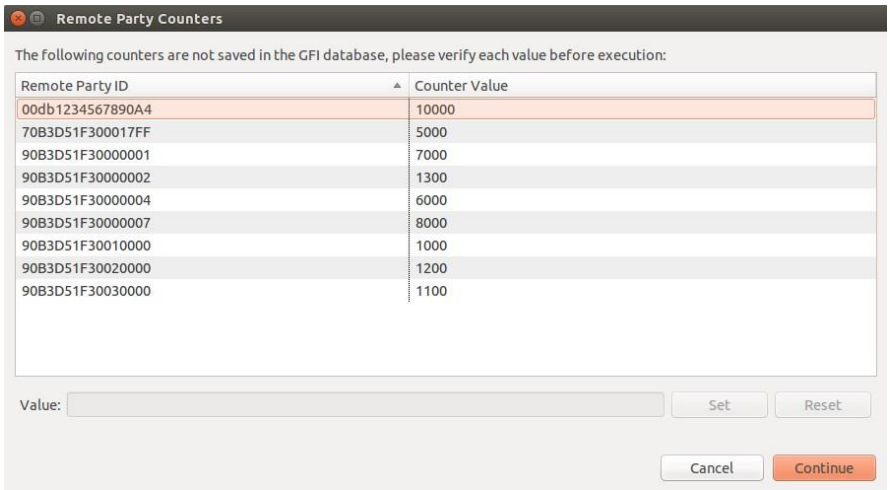


Figure 48 – Remote Party Counters

Although the counter values may be changed, they will be initially filled in with defaults read from the Test Scenario. The Tester may always change these values later on by means of the application menu sequence: **Edit -> Counters**.

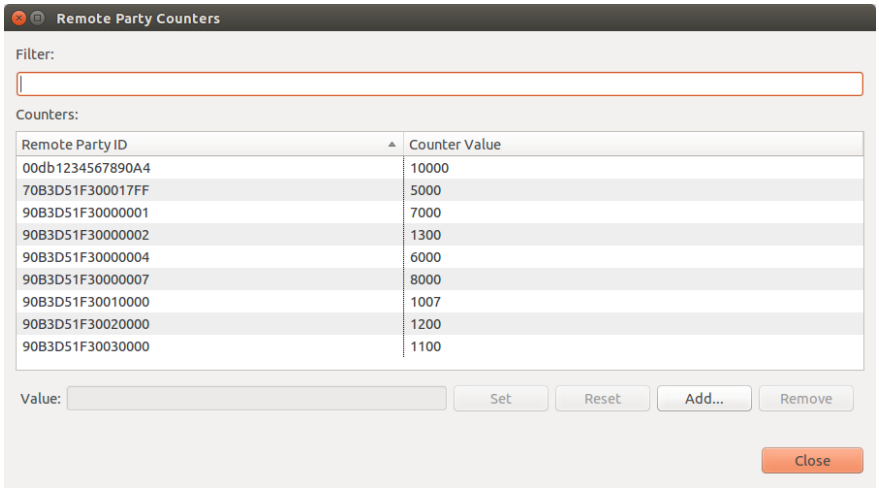


Figure 49 – Remote Party Counters Editor

Click 'Continue' (Figure 48) for the test to be executed.

The console activity will then be displayed 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 50). 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 27.

Figure 50 – Console activity

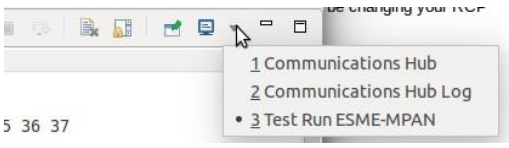


Figure 51 – Console page for Test Run selected

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[12] – all previously saved reports are displayed in the Edit/View pane. Upon running the test, a new Campaign item resulting from the Test's execution is added to the Explorer pane, along with all the respective configuration, execution and metric files.

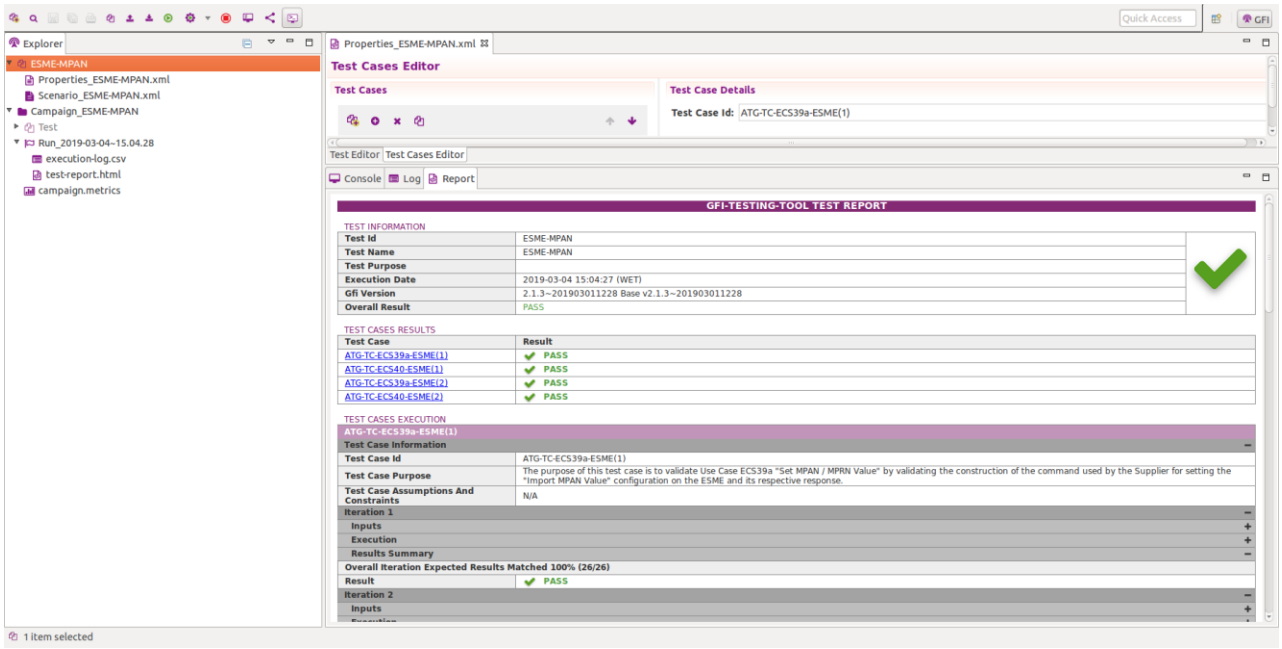


Figure 52 – Execution finished

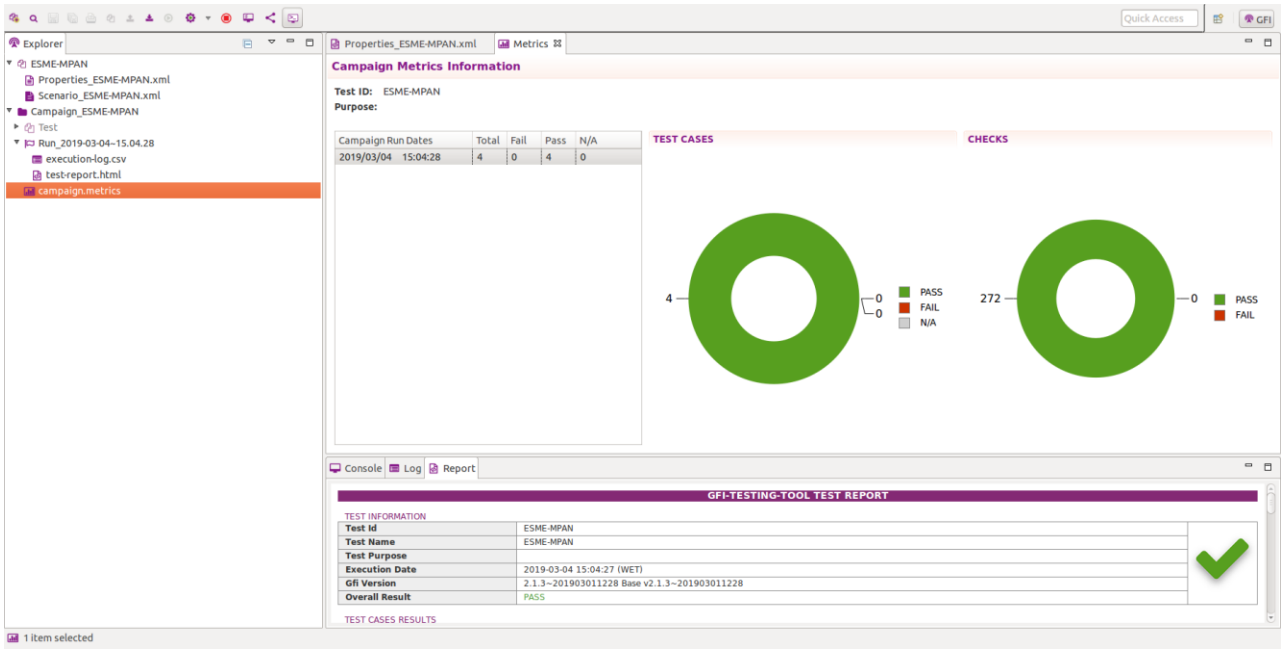


Figure 53 – Campaign Metrics

[12] This is also true when multiple tests are run: only the last report will be shown.

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.

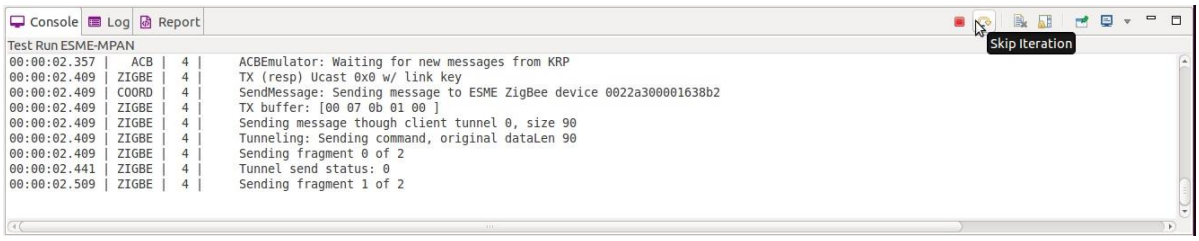


Figure 54 – Skip and Stop

This concludes the Test execution. As mentioned before, should 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 55 depicts multiple executions of a Campaign.

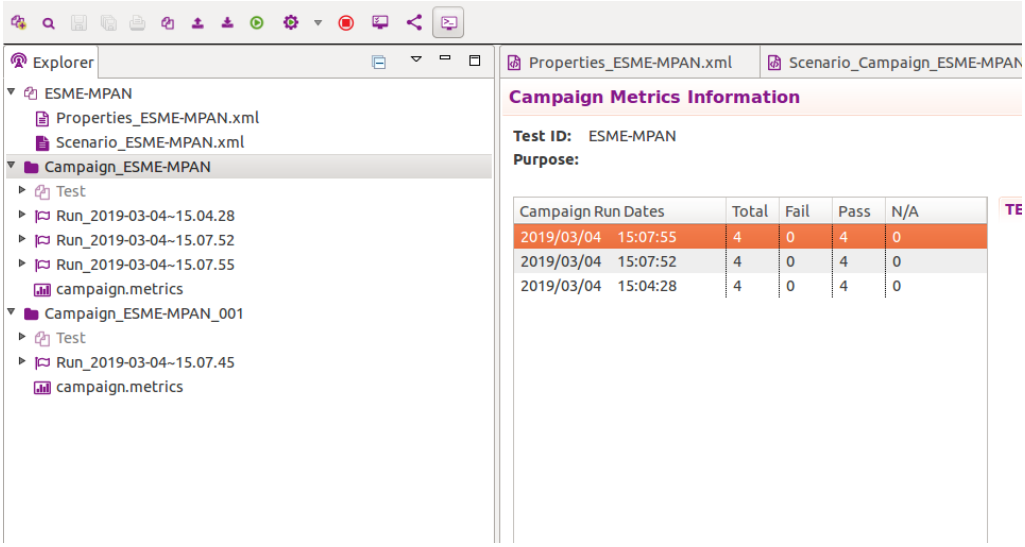


Figure 55 – Multiple Runs

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

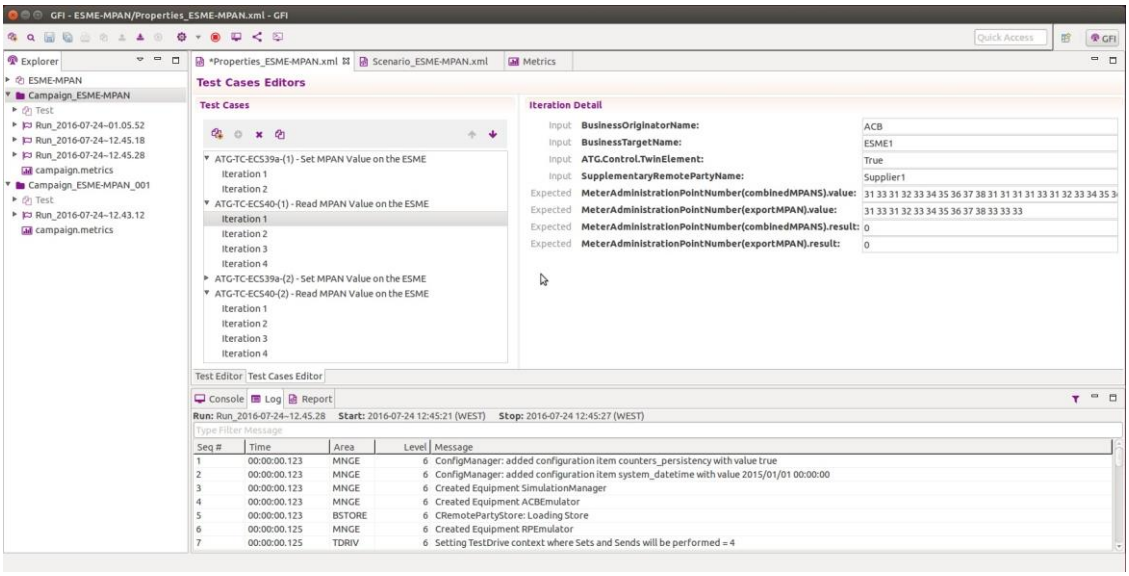


Figure 56 – Set Expected values

These values should be set in **both** Iterations of **both** read Test Cases (ATG-TC-ECS40-(1) and ATG-TC-ECS40-(2)).

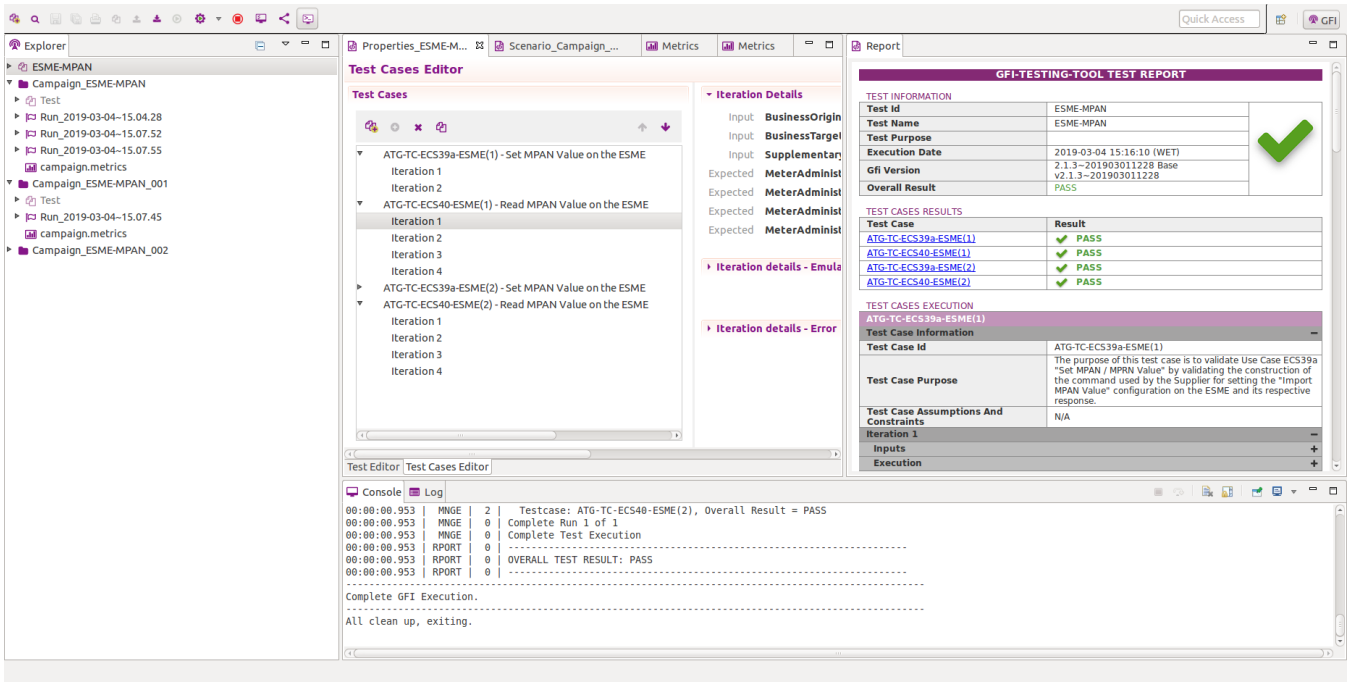


Figure 57 – IDE layout (Properties and Report).

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 the desired value. 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 47 – 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 58 needs to be changed. Note that the proper Iteration should be selected.

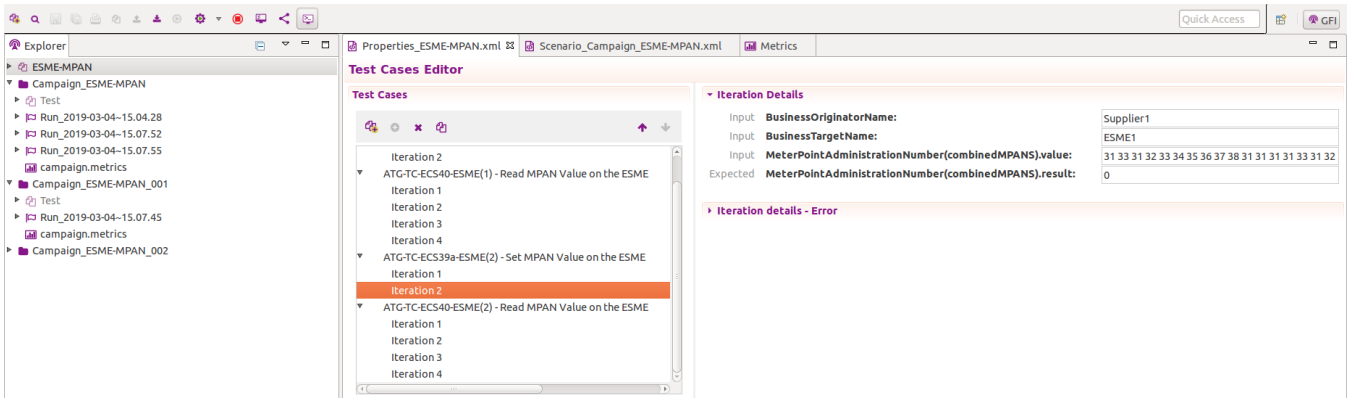


Figure 58 – Edit Test Case Properties – Before changing the MPAN Value

After assigning a different value to the parameter, the Iteration Detail should look like Figure 59.

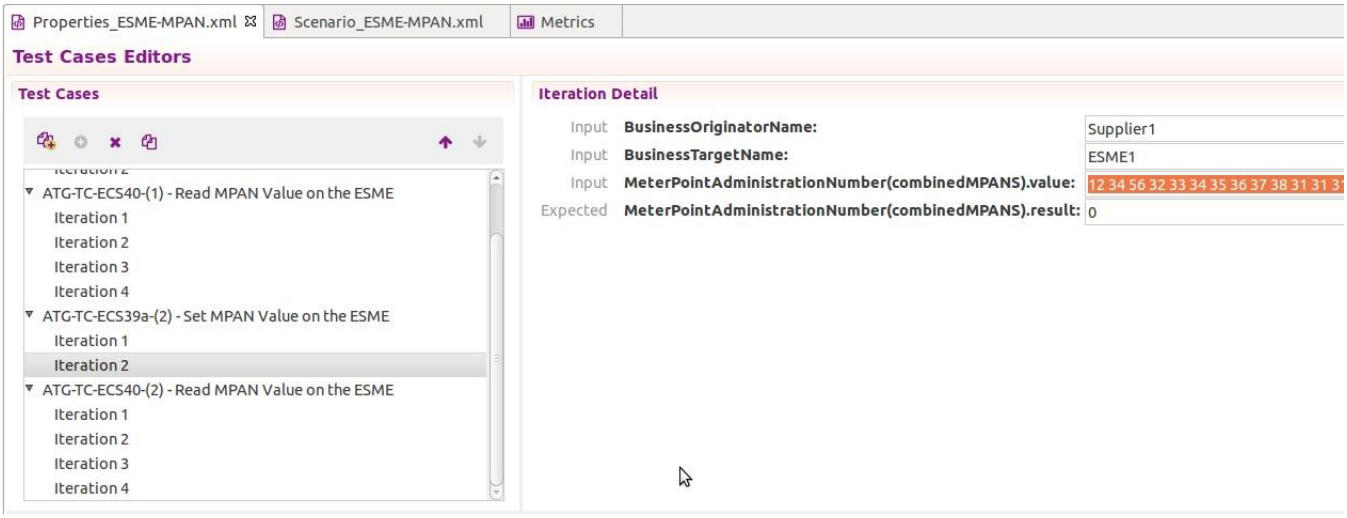


Figure 59 – Edit Test Properties - After changing the MPAN value

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.

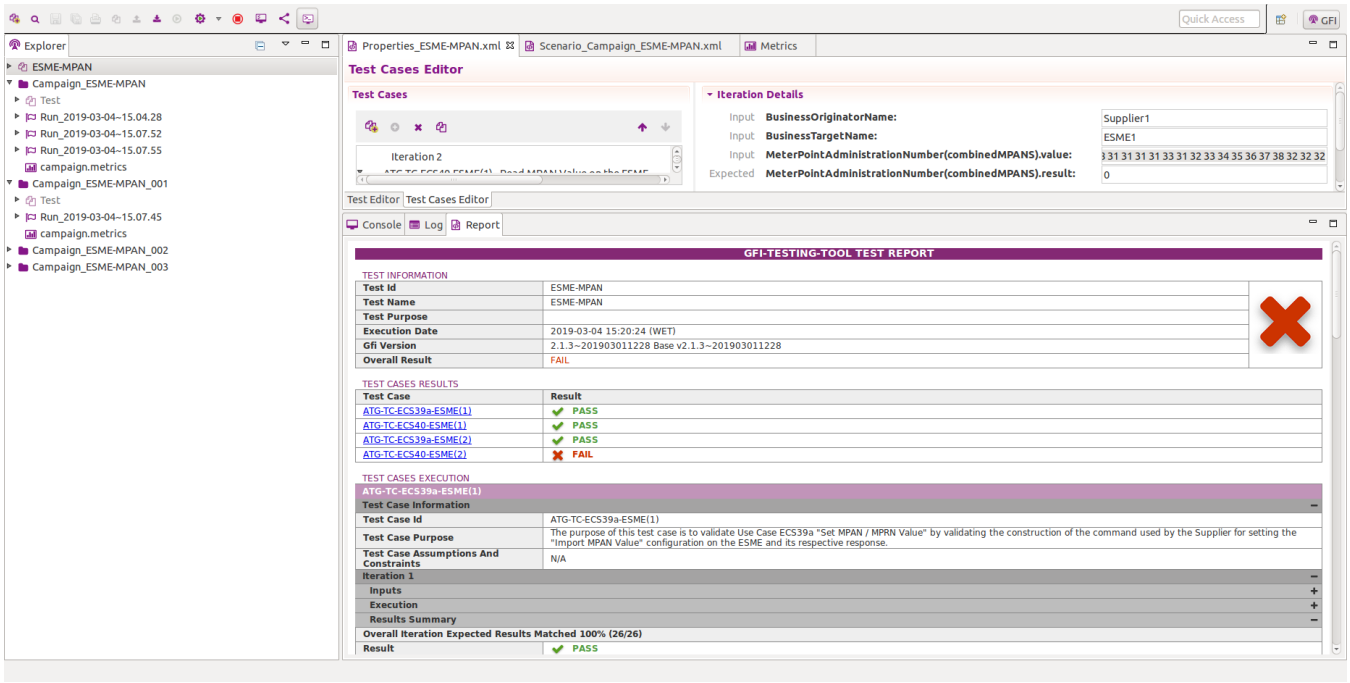


Figure 60 – Test FAIL

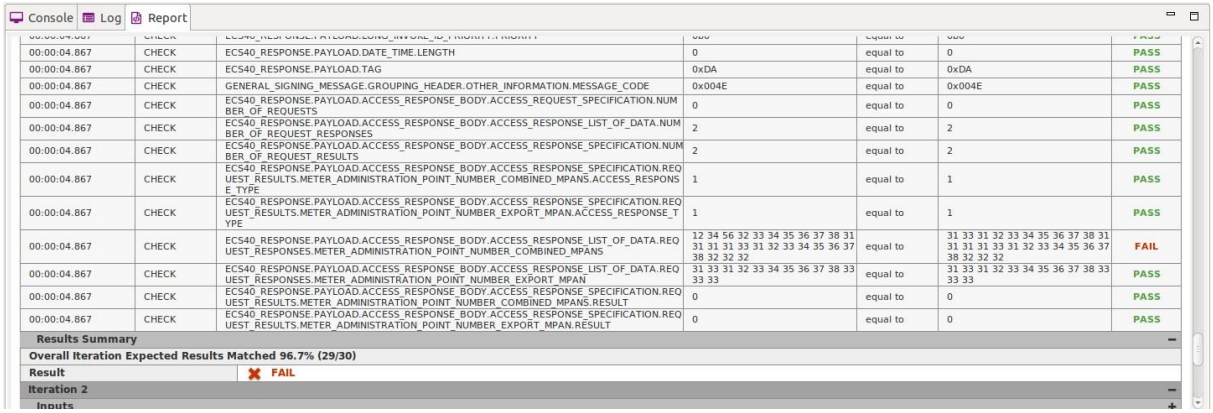


Figure 61 – Test FAIL detail

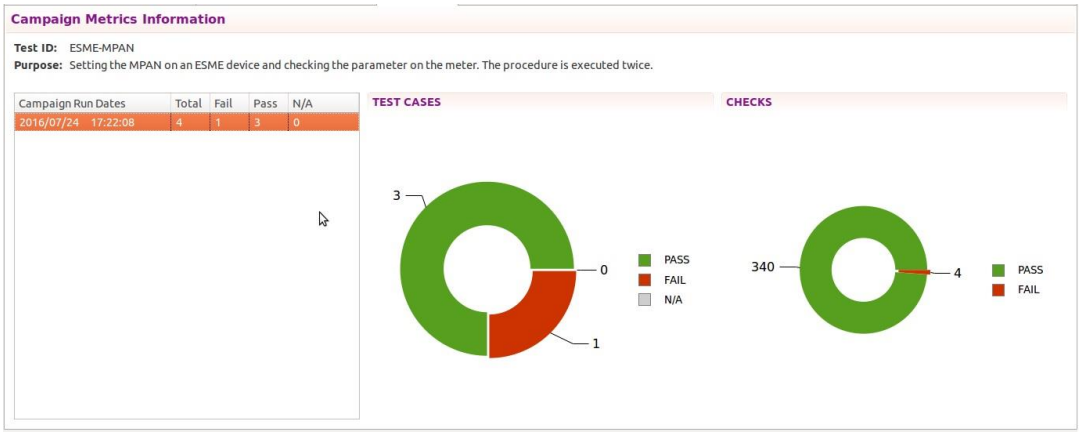


Figure 62 – Metrics Information

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 altered parameter value and so, expected values in all iterations should be changed.

Figure 63 shows how to change the Expected MPAN value. 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.

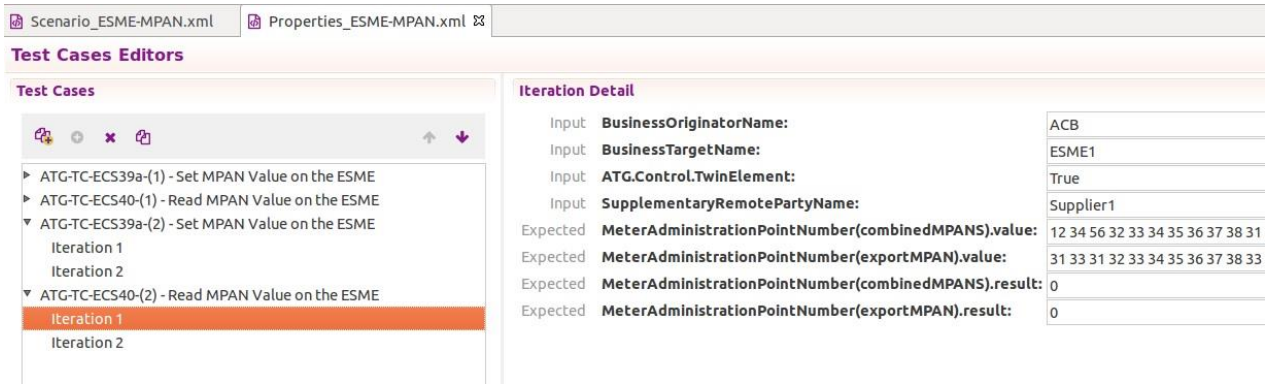


Figure 63 – Change expected value

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. The parameter was updated successfully.

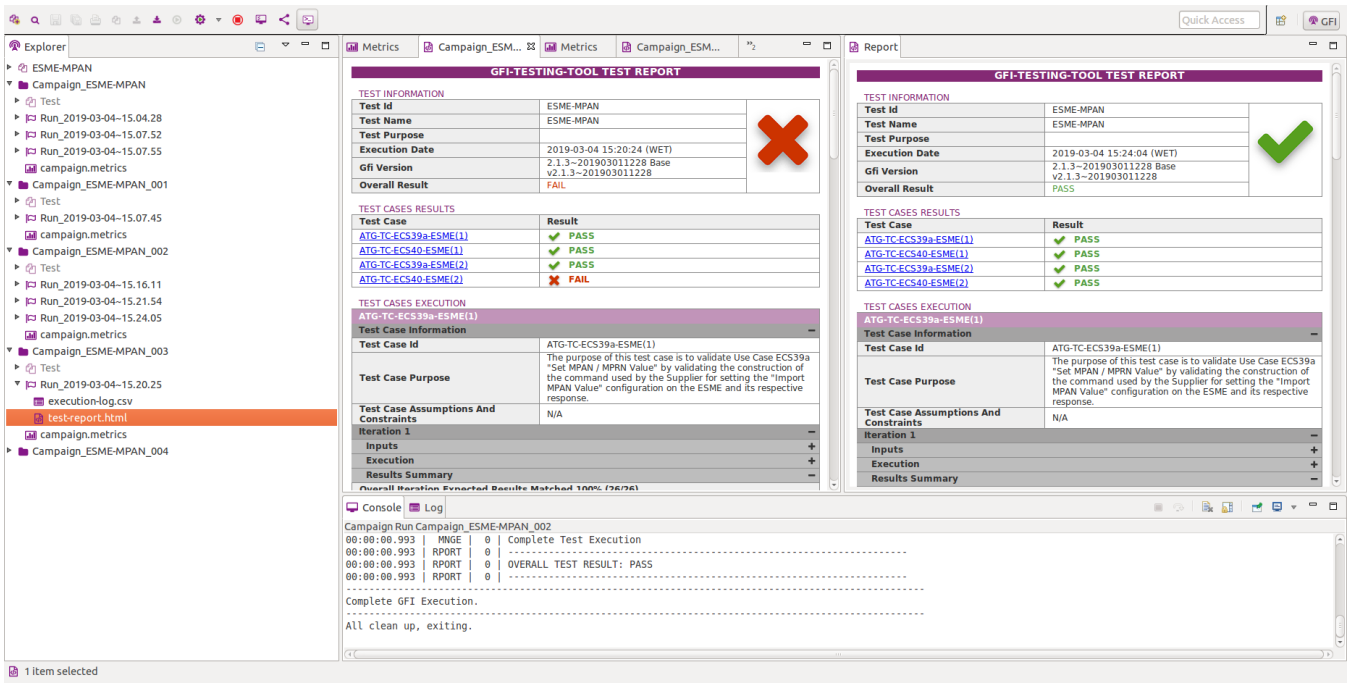


Figure 64 – Test reports side by side

As shown in Figure 64, 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).

Additionally, the reports may be viewed side by side for result analysis in a comparative fashion. This can be done for every input or output artefact (console output or execution logs).

Logging

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

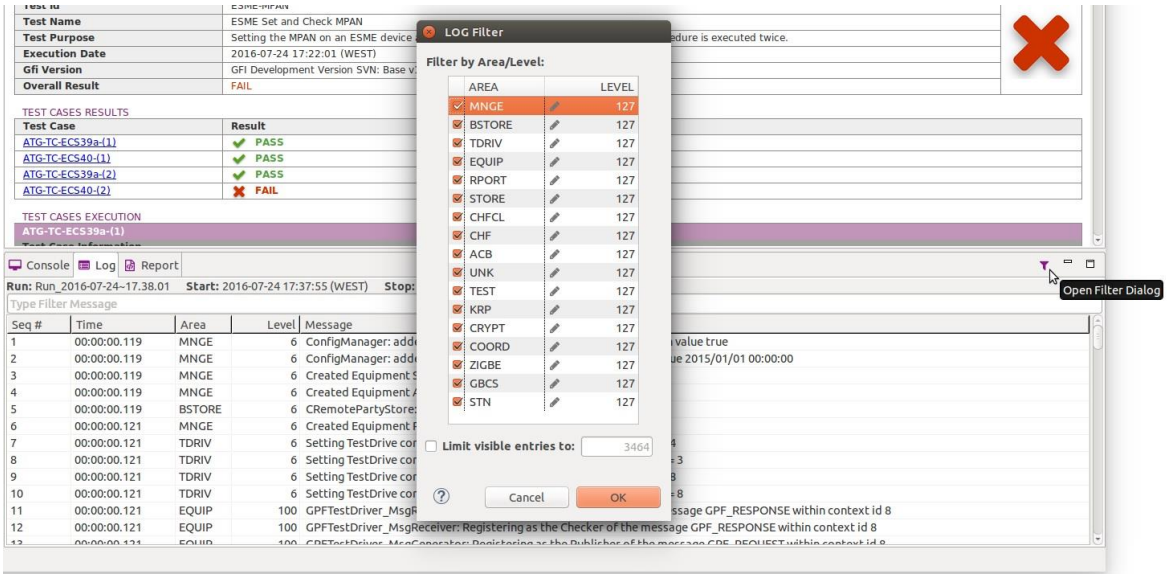


Figure 65 – Open Filter Dialog

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.

4.3.6 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 reused for later Import (creating a new Test), through the context menus or the 'Import' button.

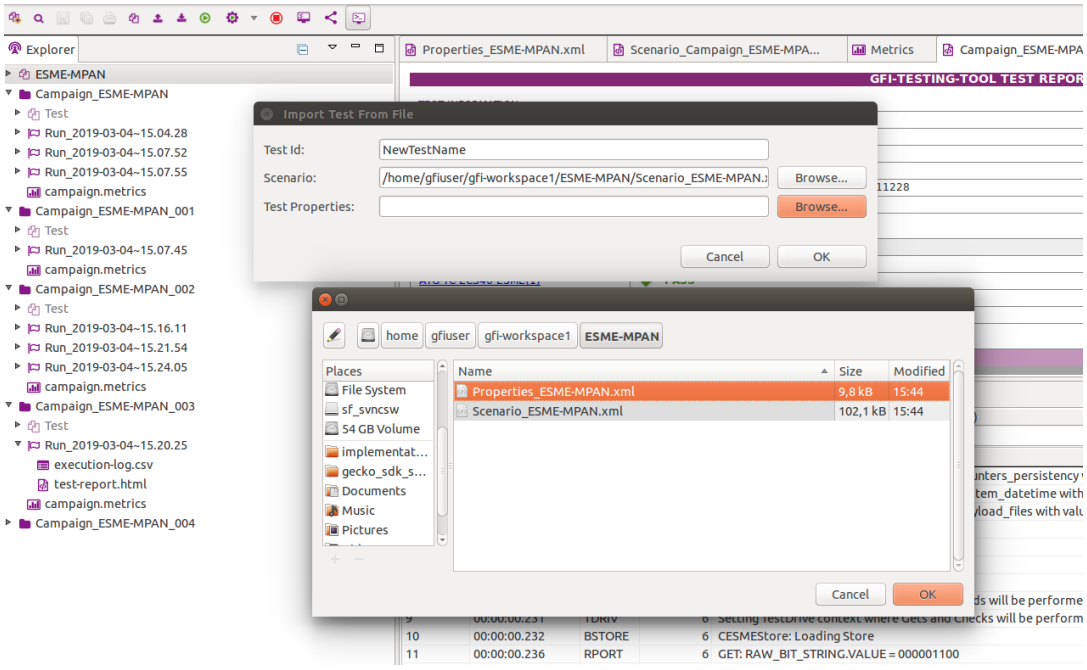
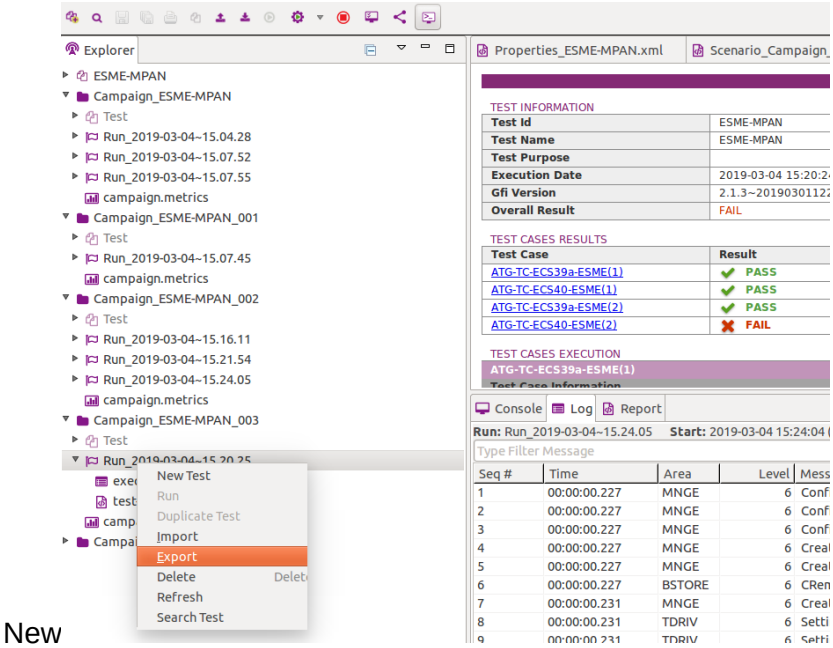


Figure 66 – Import Test

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.



New

Figure 67 – Export Test Run

The selected output path should have write permissions for the user.

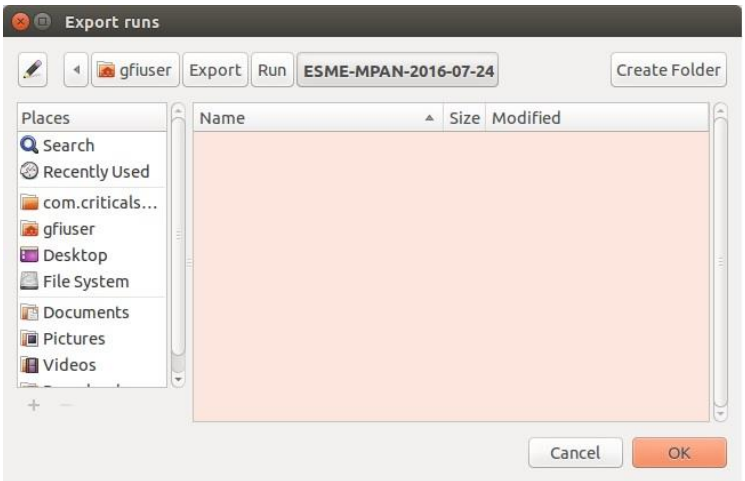


Figure 68 – Select output path

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).

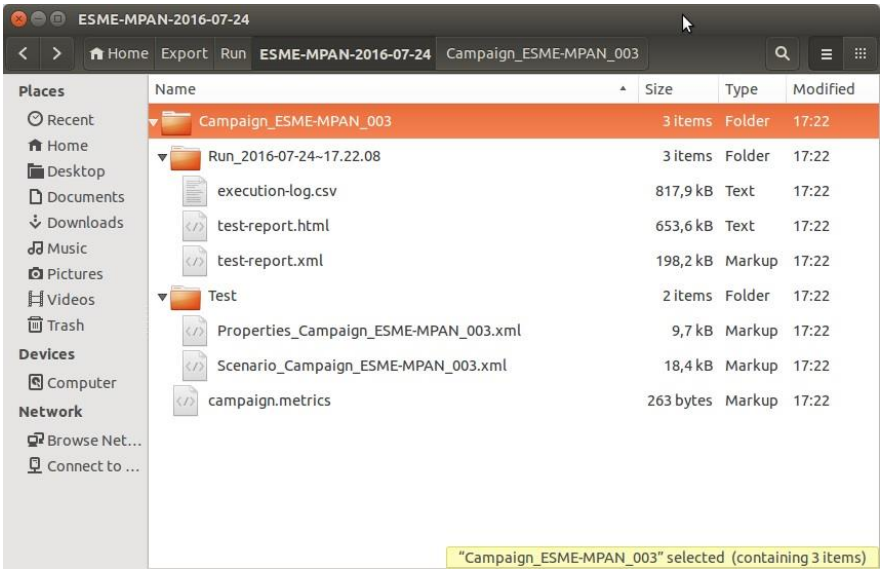


Figure 69 – Exported files

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.

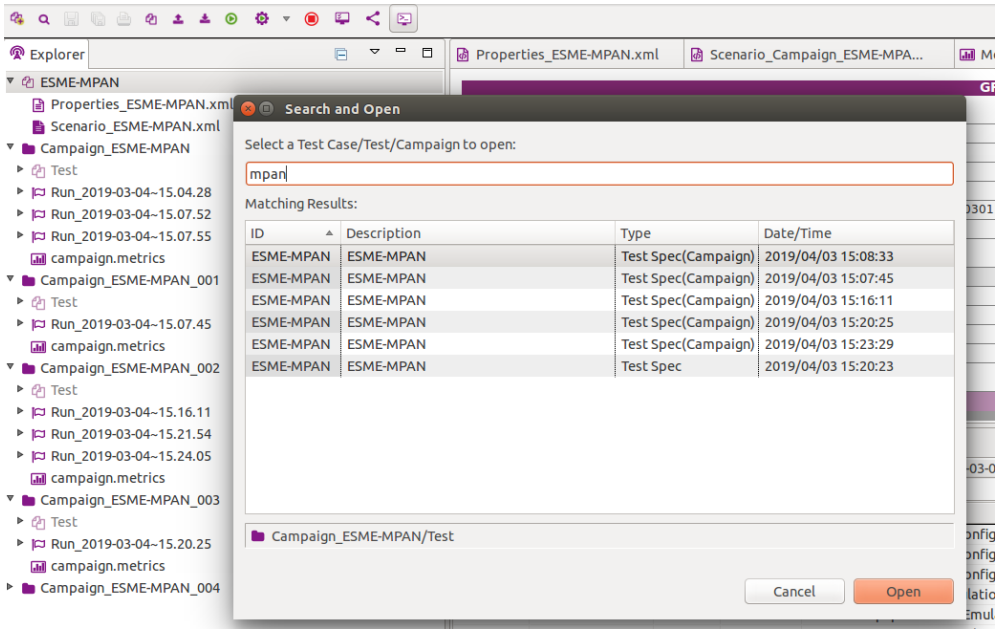


Figure 70 – Search and Open

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).

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 (right-clicking over a test), the same dialog will be displayed, but the scenario and properties' file paths are already filled with the file paths from the test selected. The process is illustrated in Figure 71, Figure 72 and Figure 73. Notice that the user can still alter the file paths if they so wish.

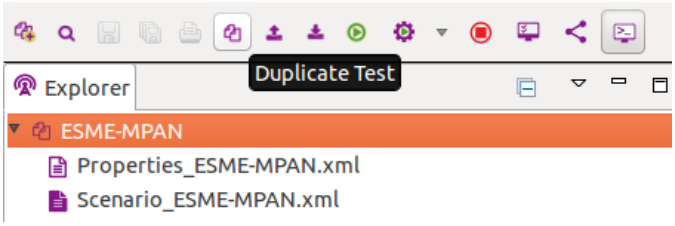


Figure 71 – Invoking the 'Duplicate Test' functionality

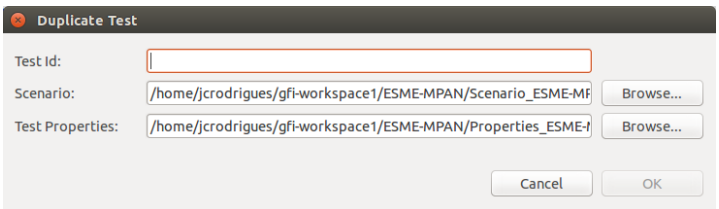


Figure 72 – 'Duplicate' Test dialog

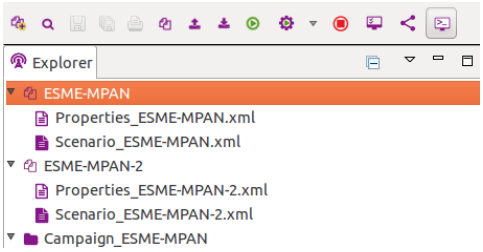


Figure 73 – Explorer showing duplicated test

Import/Export CH Configurations[13]

As has 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. If is also possible to export the configurations produced to a XML file and loading a CH configuration file into the list.

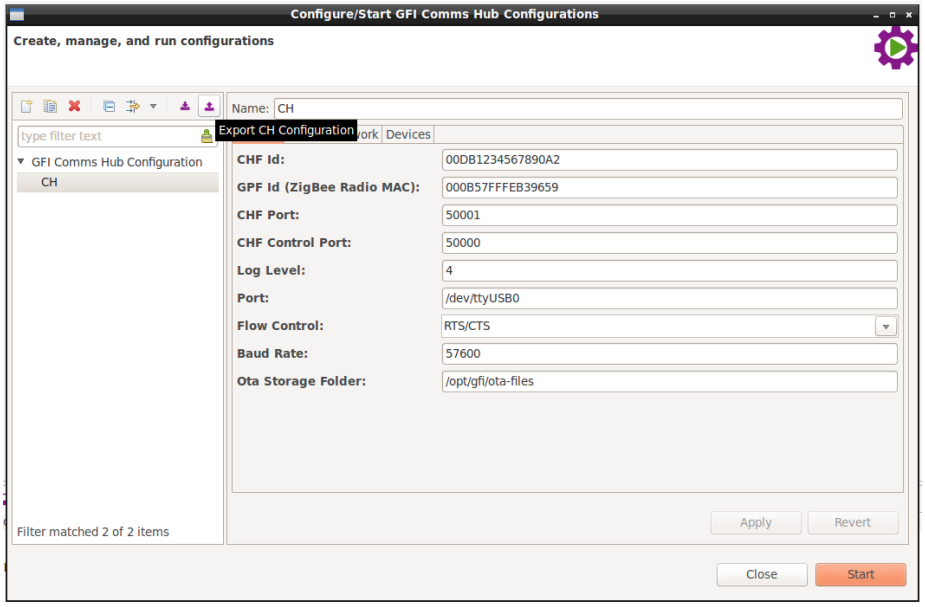


Figure 74 – Export CH configuration icon

Revisiting the CH configuration management dialog, and selecting the export configuration file icon (Figure 74), the save file dialog will be displayed (Figure 75).

[13] It is not possible to import CH configurations created with a version of this GUI earlier than 2.0.0. The file formats are not compatible.

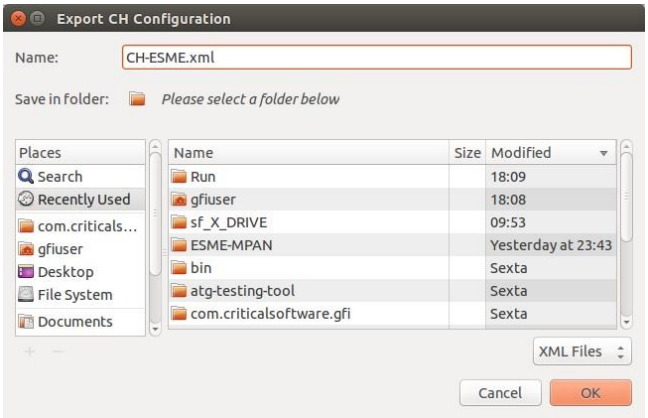


Figure 75 – CH configuration export file save dialog

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 76).

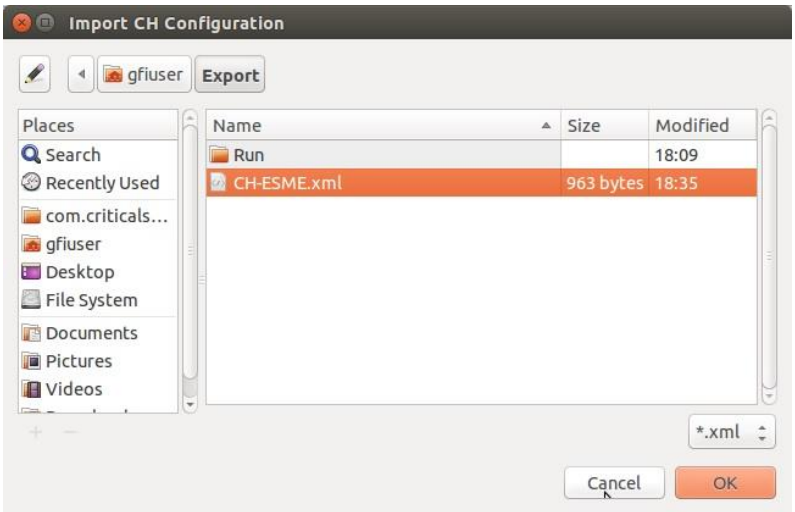


Figure 76 – CH Import file open dialog

A configuration will be created in accordance with the contents of this file (Figure 77). 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.

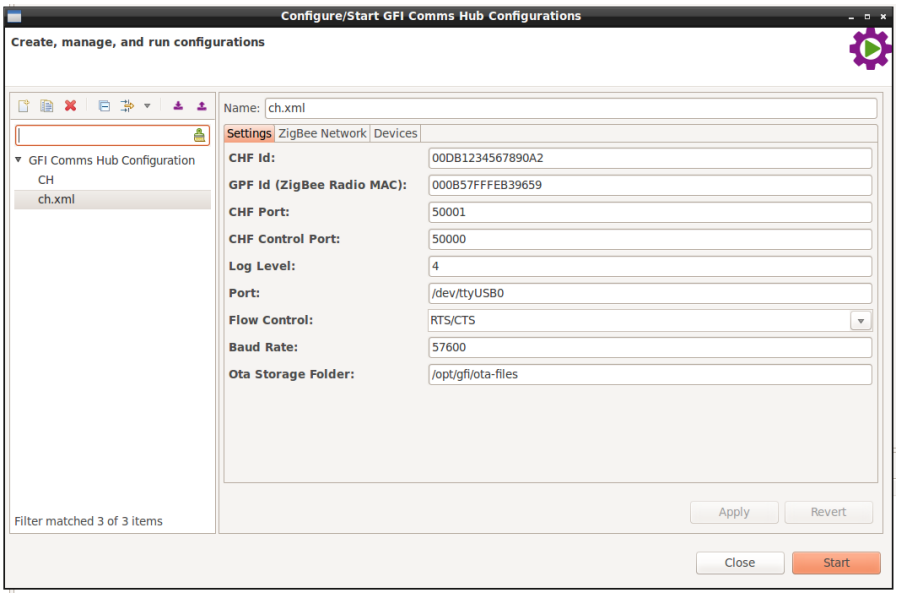


Figure 77 – Configuration imported from the XML file exported previously

Import Scenario functionality

This functionality allows the user to select a workspace folder and import all the tests in that workspace at once.

All the files and folders inside the workspace folder (e.g. Campaign folders) that are not GFI tests will be ignored.

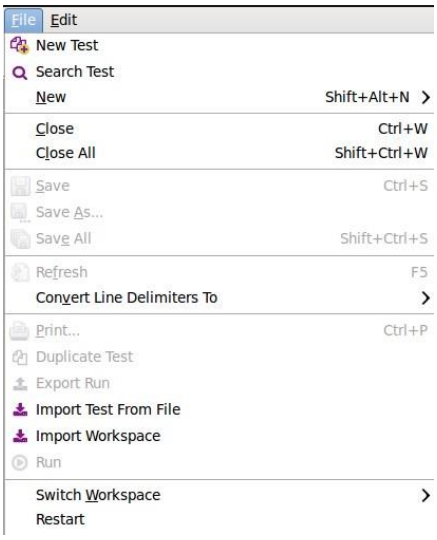


Figure 78 - Import Workspace option in the menu File

By clicking on the Import Workspace option on the menu File, a new dialog window should open to allow the user to select the folder of the workspace to be opened.

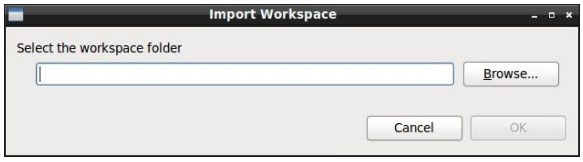


Figure 79 -Import Workspace window

Selecting a workspace folder will make the OK button available. By clicking on the OK button all the tests in the workspace should be imported.

After the tests have been imported, a summary of the operation is shown in the console area.

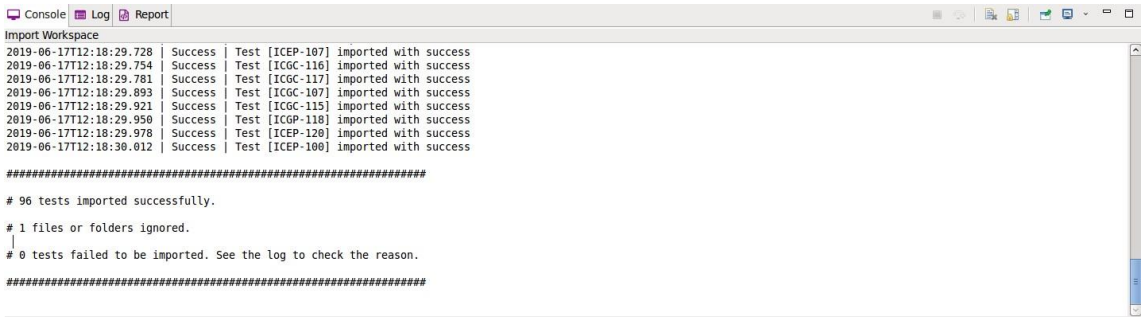


Figure 80 - Import resume

Rename Test

This functionality allows to change the name of a test. When invoked, either from the File menu through Rename Test or from the context menu (right-clicking over a test), the dialog box appears where the Test Id can be edited. Click OK to save the changes. The process is illustrated in Figure 81 and Figure 82.

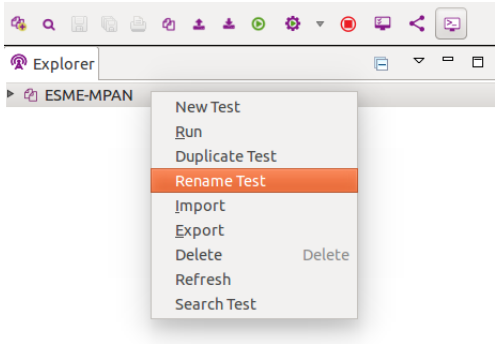


Figure 81 – Invoking the 'Rename Test' functionality through Right-Click over the test

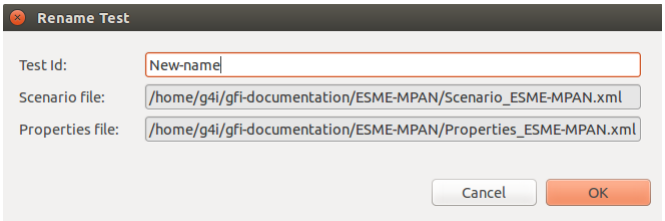


Figure 82 – Dialog box for 'Rename Test' functionality

Change Scenario

This functionality allows to switch scenarios between Emulated, GFI CH and Interface Module modes. It is invoked through the File menu by selecting Change Scenario (Figure 83). On the dialog box, select the desired CH Configuration (Figure 84). In the case the GFI CH or the Interface Module are selected, the certificates for each device must be specified by clicking on the device followed by the Edit Device button. It is possible to choose the devices to change by selecting the corresponding checkbox or to select/clear all devices using the buttons "Check All" and "Clear All", respectively. When editing the device configuration, a dialog box similar to Figure 36 is presented a in the case of the Emulated mode the default configurations are used. Click Start to apply the new configuration. Note that if a device configuration changes, this functionality will change all the test cases in the Workspace. If the device configuration remains the same, no changes will be applied.

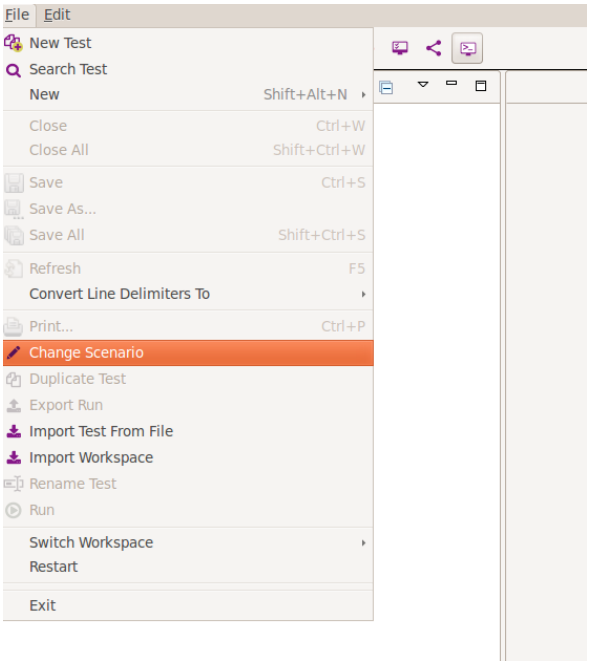


Figure 83 – Invoking the 'Change Scenario' functionality through the File menu

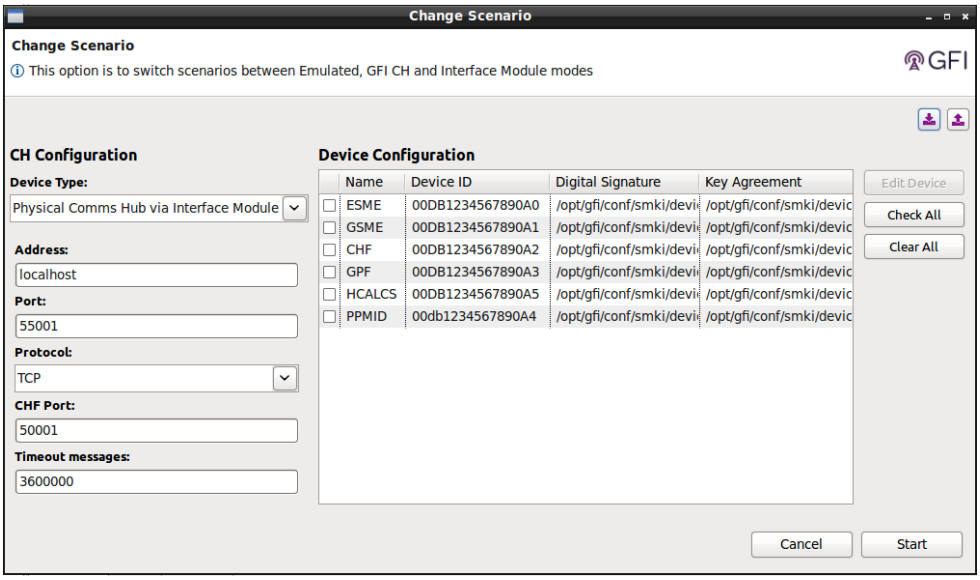


Figure 84 – Dialog box for 'Change Scenario' functionality

OTA Tool

This functionality allows to build and/or verify OTA Upgrade Images. It is invoked through the Utilities menu by selecting OTA Tool (Figure 80).

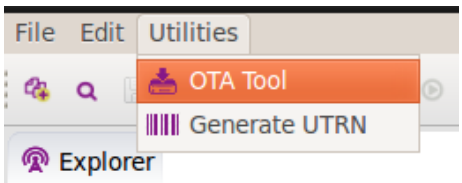


Figure 85 – Invoking the 'OTA Tool' functionality through the Utilities menu

On the dialog box there are two tabs, the first one is to build an Upgrade Image and the second one is to verify the Upgrade Image, or the files required to build it.

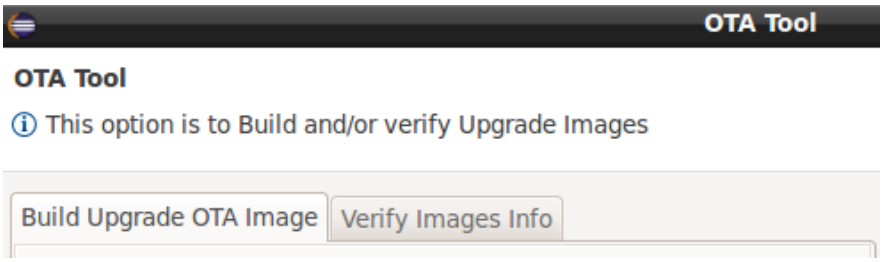


Figure 86 – Build and Verify tabs

On the **Build Upgrade OTA Image** tab, it is required to input some mandatory files: Header bin file, Manufacturer Image bin file and Private Key file. Instead of Header and Manufacturer Image bin files (Figure 82) it is possible to use a Single file containing both contents (Figure 83), using the option "Build

with Header + Manufacturer Image files" under the **Select Build Type** option. The folder where the Upgrade Image will be saved, can also be changed by setting the path using "Choose the output folder" option. It is also possible to set the **Force Replacement Flag** using the checkbox.

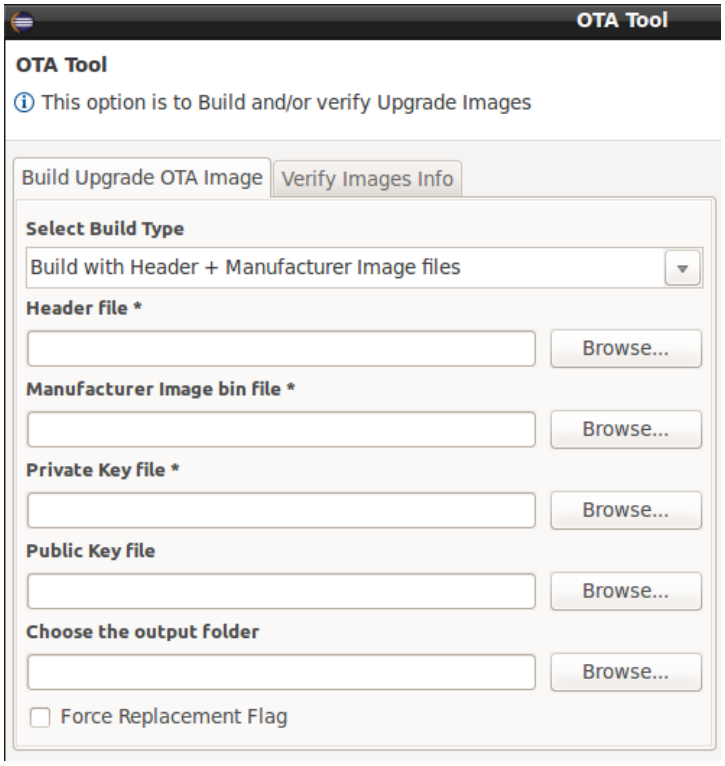


Figure 87 – Build Upgrade Image with Header and Manufacturer Image

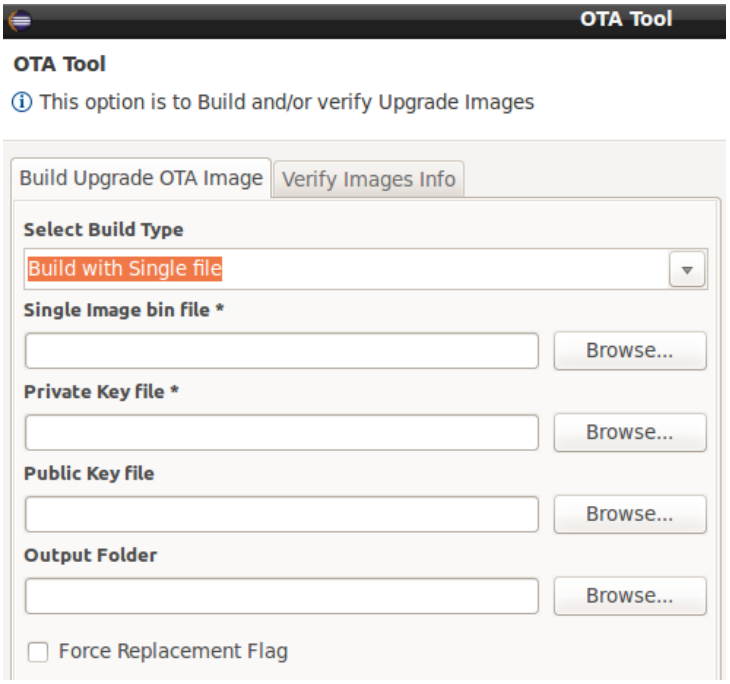


Figure 88 – Build Upgrade Image with Single File

On the **Verify Images Info** tab the mandatory files are the Upgrade Image (Figure 84) or the Manufacturer Image (Figure 85) depending on the option selected on **"Select Verify Option"**. On Manufacturer Image verification the Header bin file or/and the Signature and Public Key files are required.

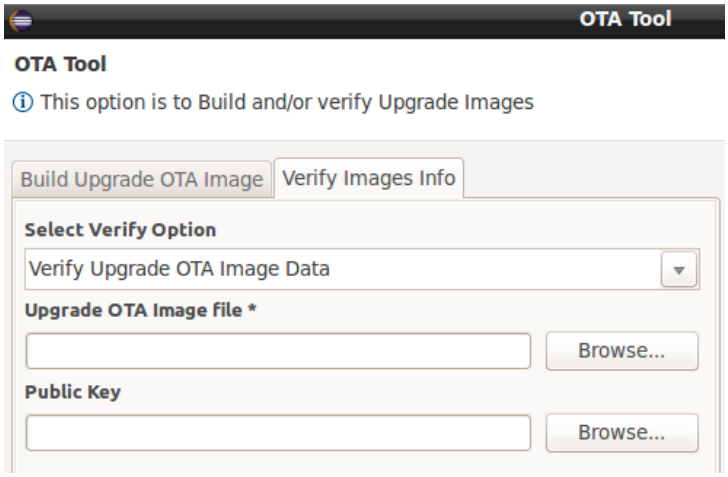


Figure 89 – Verify Upgrade Image

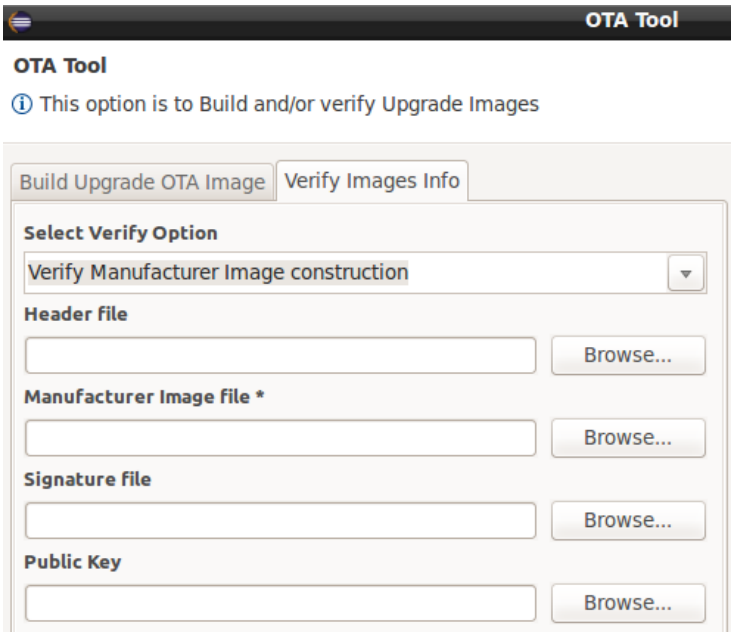


Figure 90 – Verify Manufacturer Image

The form errors will be shown at the Dialog Header in an info section (Figure 86).

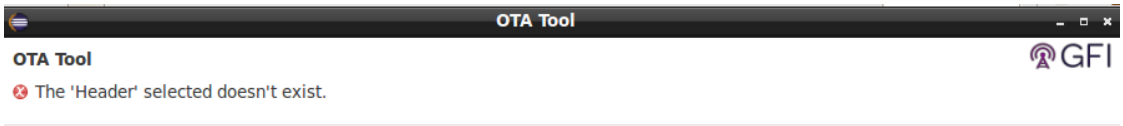


Figure 91 – Errors

The operation result and other information will be shown in a log section in the right part of the Dialog (Figure 87).

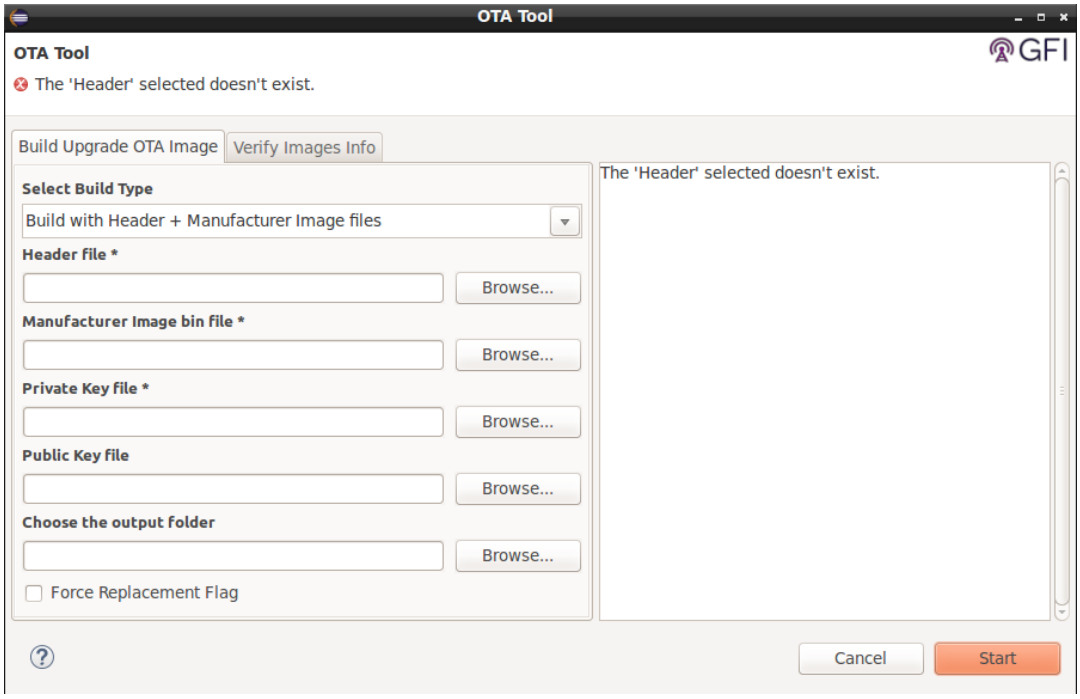


Figure 92 – Log area

Generate UTRN Tool

This functionality allows to generate UTRN codes for Prepayment Top-Ups. It can be invoked through the Utilities menu by selecting Generate UTRN (Figure 93), or by right-clicking on an existing test (Figure 94).

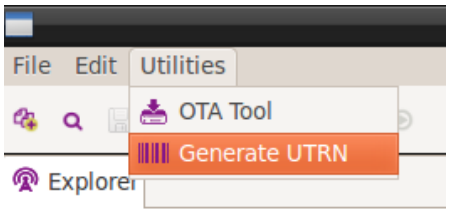


Figure 93 – Invoking the 'Generate UTRN' functionality through the Utilities menu

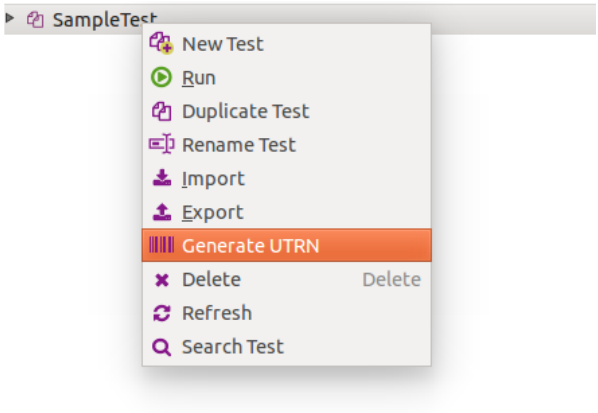


Figure 94 – Invoking the 'Generate UTRN' functionality through right-clicking on a test

If accessed through the Utilities menu, a window is opened as in Figure 95, where the user must fill all the fields manually. If accessed through right-clicking on a test, a window is opened as the one from Figure 96, where the fields are populated with the existing values on the Scenario file of the selected test. The Originator Counter is automatically updated if the selected Supplier is on the Counters list, otherwise it is started at 0. The user can select the button Edit Counters to manage the counters. After all fields are correctly filled, click Generate and a new window will open with the generated UTRN (Figure 97). After clicking OK, the user can continue to generate UTRN codes on the same Generate UTRN window.

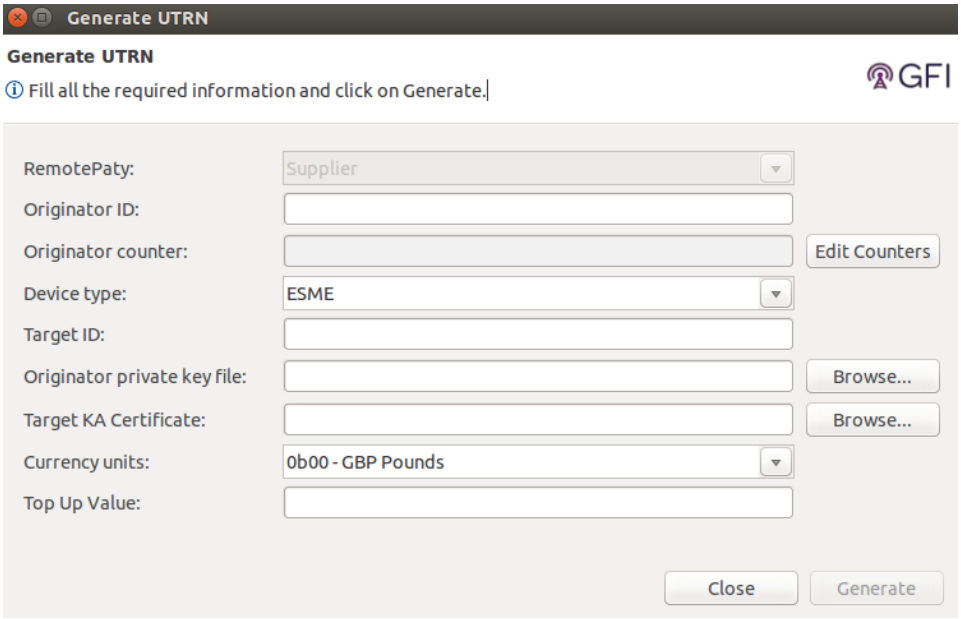


Figure 95 – Generate UTRN window from Utilities menu

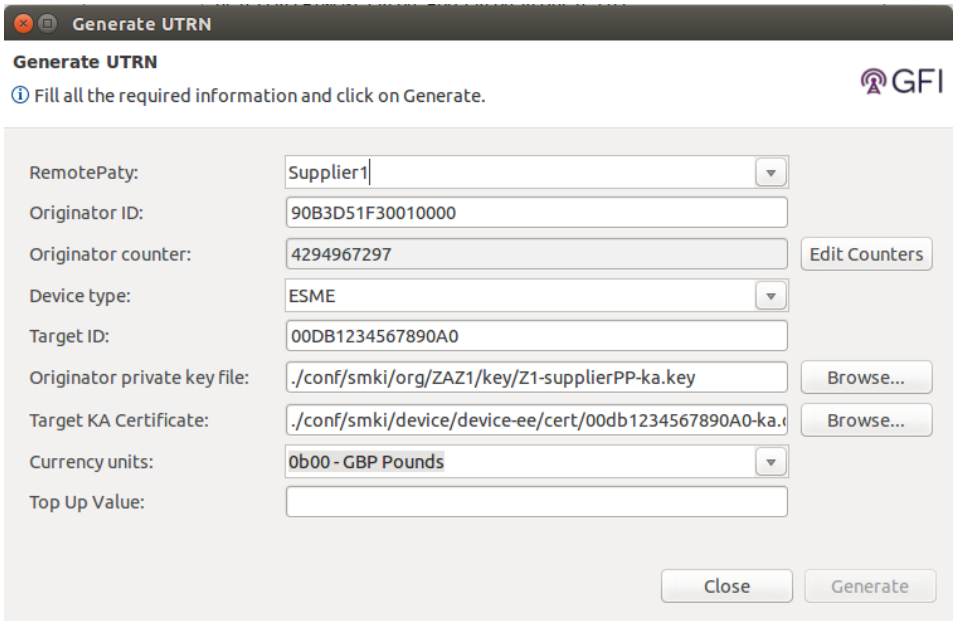


Figure 96 – Generate UTRN window by right-clicking a test

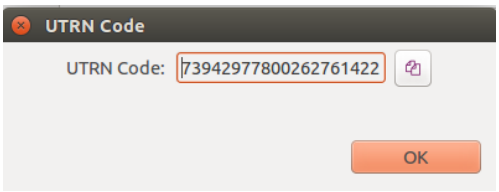


Figure 97 – UTRN window after generating

A Campaign with a unique timestamp is left on the workspace with the information related to the UTRN generation.

Custom use cases

It is possible to send a custom (non-GBCS) use case to a device. To do so, when selecting the test cases, choose the ATG-TC-Custom to the respective target device (Figure 98).

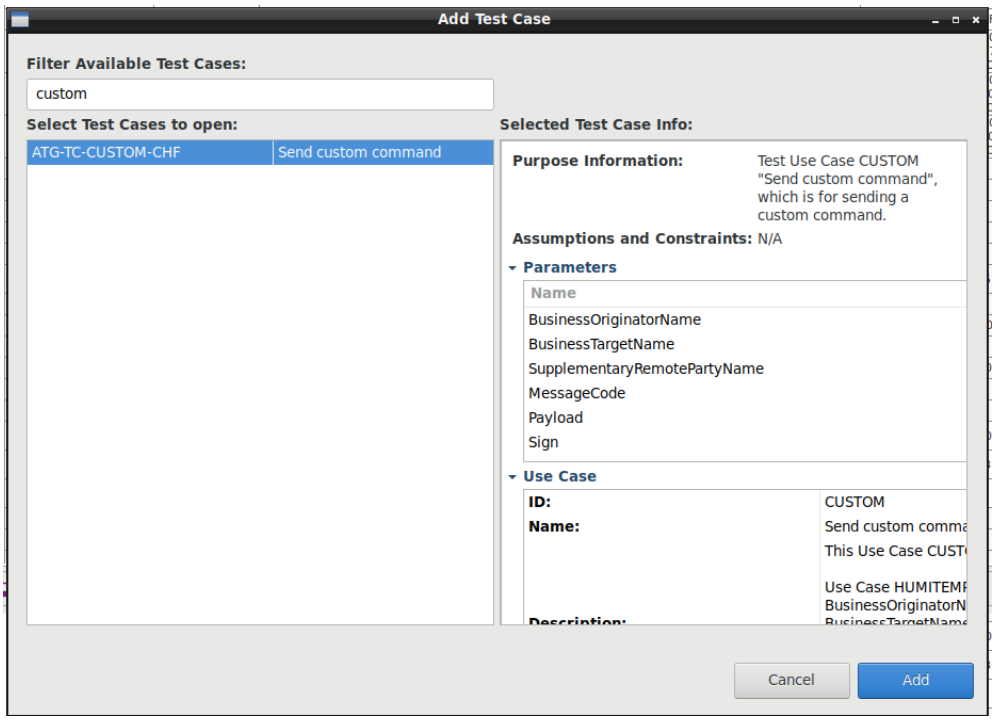


Figure 98 – Selection of the custom use case

It is required to specify the message code, payload and whether the command must be signed before sending to the target device (Sign option) – please refer to Figure 99. In this example, a command to read the device's firmware version (ECS52) is to be sent.

Figure 99 - Properties of the custom use case

5 Glossary

Table 3 presents the list of definitions used throughout this document.

Name	Description
Applicable Document	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.
Reference Document	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.
Test	A set of Test Cases (ranging from one to N) that are grouped together with the purpose of being executed in one run.
Test Case	A Use Case as defined by SMETS.
VSIS™	A CSW Critical Systems Validation Platform

Table 3 – Definitions.

Table 4 presents the list of acronyms used throughout this document.

Acronym	Description
ACB	Access Control Broker

Acronym	Description
AD	Applicable Document
ATG	Automated Test of GBCS
CBKE	Certificate-Based Key Exchange
CHF	Communications Hub Function
CSW	Critical Software, S.A.
DUT	Device Under Test
ESME	Electricity Smart Metering Equipment
GBCS	Great Britain Companion Specification
GFI	GIT For Industry
GIT	GBCS Interface Testing
GPF	Gas Proxy Function
GSME	Gas Smart Metering Equipment
KRP	Known Remote Party
NA	Not Applicable
OTA	Over-The-Air
PWO	Pass With Observations
RD	Reference Document
TBC	To be confirmed
TBD	To be defined
UTRN	Unique Transaction Reference Number

Acronym	Description
ZCL	ZigBee Cluster Library

Table 4 – Acronyms.