

Graphical User Interface Manual

GIT for Industry

Smart DCC Ltd.
v1.0 (Draft)
2015/06/29

The contents of this document are under copyright of Smart DCC Ltd., released on condition that it shall not be copied in whole, in part or otherwise reproduced (whether by photographic or any other method) and therefore shall not be divulged to any person other than the addressee (save to other authorized offices of his organization having need to know such contents, for the purpose for which disclosure is made) without prior written consent of submitting company.

Table of Contents

1	Introduction	2
1.1	Objective	2
1.2	Document structure	2
2	Overview	3
2.1	High Level Architecture	3
2.2	Reference Testbed	4
2.3	Inputs and Outputs	5
3	Environment Start-up	7
3.1	Zigbee HAN network connection	7
3.2	System Start-up	7
3.3	Run Eclipse with GFI plugin	7
4	GFI Testing Tool GUI Operation	8
4.1	Running Eclipse	8
4.2	Work Environment description	8
4.3	Step-by-step Example	10
5	Glossary	27

1 Introduction

1.1 Objective

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

1.2 Document structure

Section 1 (Introduction) presents a general description of this document's contents.

Section 2 (Overview) presents a general description of the framework.

Section 3 (Environment Start-up) presents a description of the environment start-up.

Section 4 (GFI Testing Tool GUI Operation) presents a detailed description of the Test creation, execution and analysis process using the graphical user interface.

Section 5 (Glossary) presents the definitions and acronyms used throughout the document.

2 Overview

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

2.1 High Level Architecture

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

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

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

Figure 1 provides a high level overview of the system architecture. Besides the GFI Testing Tool Graphical User Interface there are six main modules (shown in purple): GFI-Testing-Tool application interface, VSIS-Core, VSIS-Comms, Simulation manager, Test Library and Messages Database.

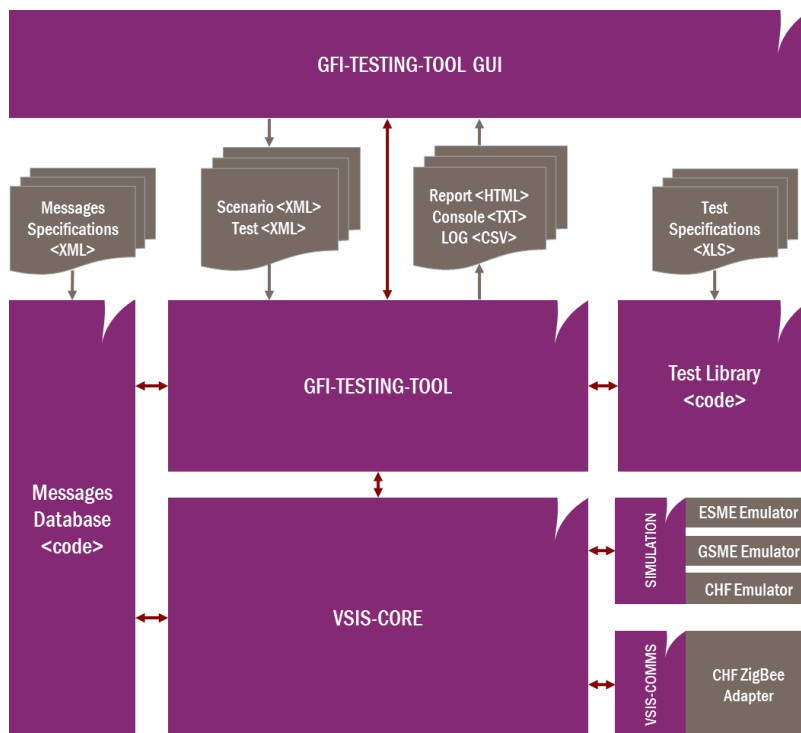


Figure 1 – GFI high level architecture.

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 VSIS-Comms

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

2.1.3 Simulation

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

2.1.4 GFI-Testing-Tool

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

2.1.5 Test Library

This is the library containing all the Use Cases specified by GBCS and available for the Tester to use in the construction of Tests.

2.1.6 Messages Database

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

2.1.7 GFI-Testing-Tool GUI

Provides a Graphical User Interface for the Testing Tool that allows for all the editing, execution and analysis functionality in the user implemented Test procedures.

2.2 Reference Testbed

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

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

Testbed

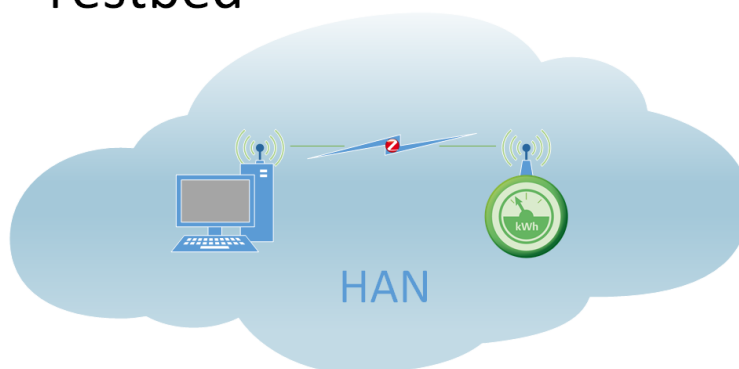


Figure 2 - GFI Testbed diagram

2.3 Inputs and Outputs

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

File	I/O	Type	Description
<scenario-file>.xml	Input	Configuration file	The scenario configuration file required by the tool. Specifies which equipment to use, the device configurations, keys and certificates, among other configurations.
<execution-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.

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 Configuration 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 instance, a Test can be executed in a scenario with real ESME/GSME devices or in a scenario with emulated devices. This configuration is defined in this file.

2.3.2 Test Execution File

The Test Cases that constitute the Test should be defined in the Test Execution file <test-execution>.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 the Test execution, a report file is generated: a formatted HTML with all the sets, checks, action prints and exchanged messages with a human readable appearance. This file (along with the Execution Log) may be considered the final output and contains the overall PASS/FAIL information as well as the detailed PASS/FAIL for each expected result.

2.3.4 Console Output

All the relevant actions executed in the Test are outputted to the console. The detail level of this execution output is maximum by default.

2.3.5 Execution Log

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

3 Environment Start-up

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

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

3.1 Zigbee HAN network connection

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

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

3.2 System Start-up

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

- Username: gfi
- Password: gfi

3.3 Run Eclipse with GFI plugin

The Testing Tool GUI is built as an Eclipse CDT (C/C++ Development Tooling) plugin. Therefore, it is advised for Users without any background in Eclipse to read the online documentation, namely the "Workbench User Guide", which is available at <http://help.eclipse.org/luna/index.jsp>.

The Test environment displays the newly created Tests and shows only the indispensable menus and options which allows for a simple use when the purpose is to create new test procedures.

4 GFI Testing Tool GUI Operation

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

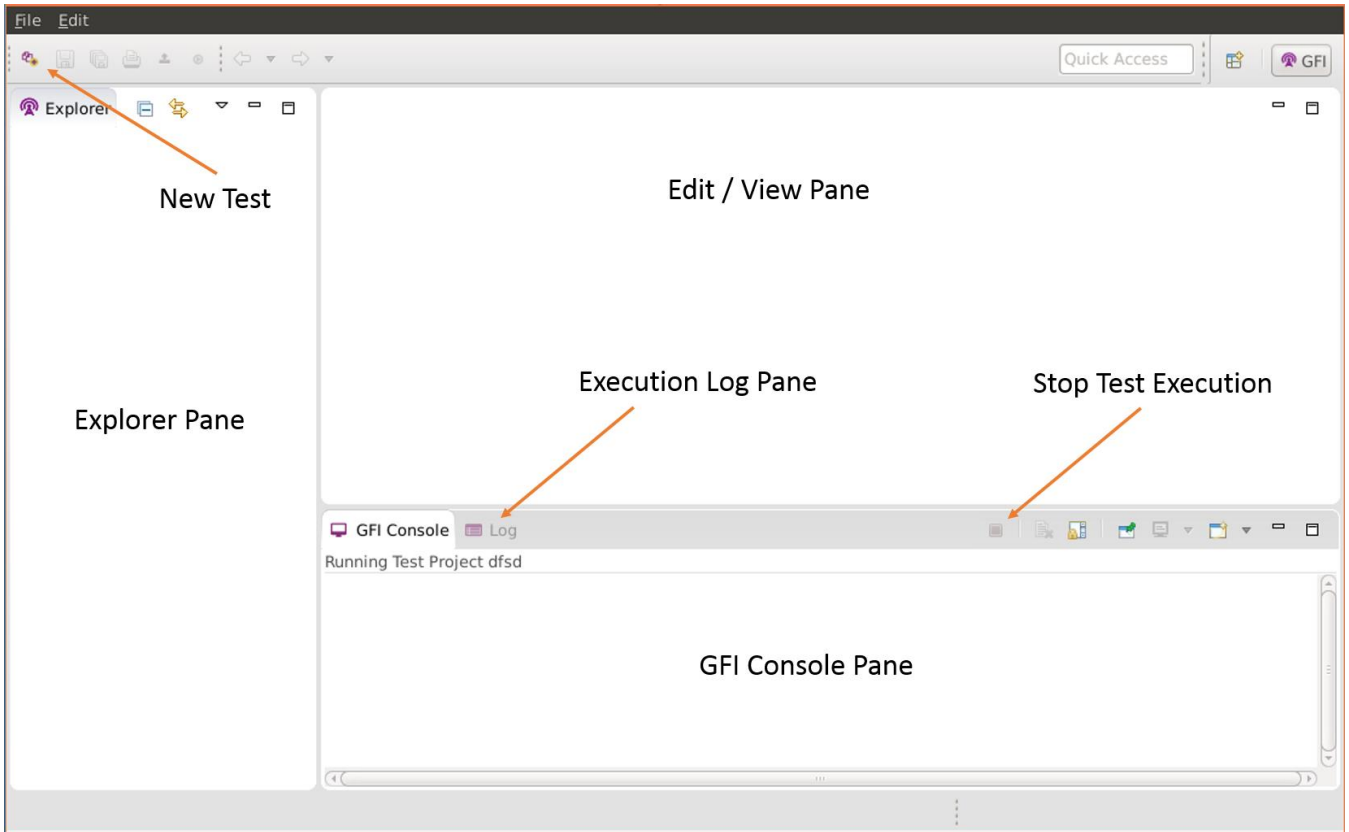


Figure 3 - GFI Testing Tool GUI

The following sections will detail each of these components.

4.1 Running Eclipse

To start the framework GUI use the shortcut for the application on the desktop. On start-up, the tests and campaigns created previously will be shown.

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

various configurations needed to run the tool are set up (like the output console and log levels, schedulers, remote parties and metering devices emulators, SMETS object storage information and message codecs).

Campaign

A Campaign contains a test section and a set of Runs.

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

Each Run contains the execution Report and Log files for analysis.

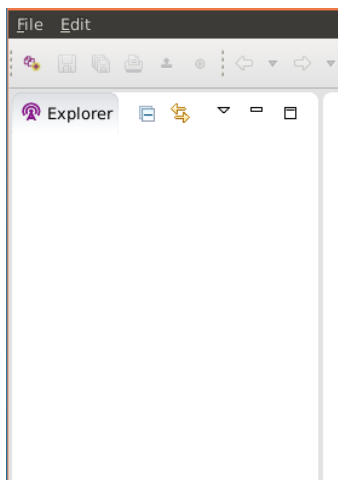


Figure 4 - Explorer

4.2.2 Console / Log Output

Upon a given test execution, the console feed will be shown in the console pane. In the same way, the execution log will be available for analysis in the Log pane. See Section 4.3.3.

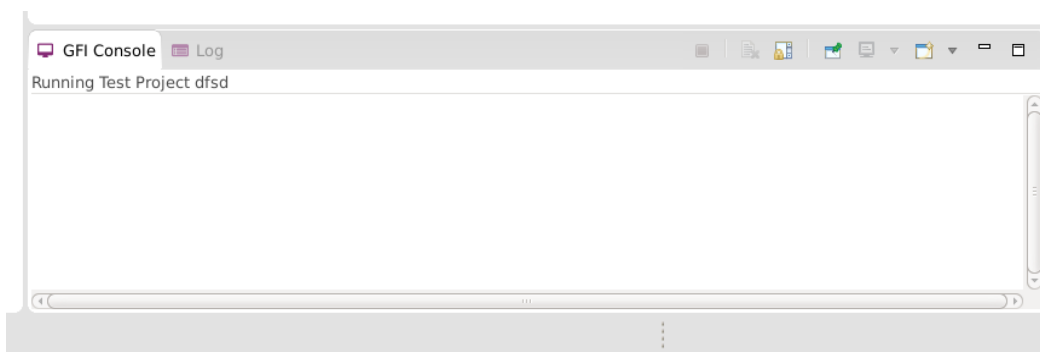


Figure 5 - Console

4.2.3 Edit / View Pane

On the working area it is possible to edit the properties of a test and of a scenario through the XML files generated by the test wizard. Also, the execution output reports and the log files may be viewed in this area for detailed analysis.

4.3 Step-by-step Example

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

4.3.1 Test Creation Wizard

Starting from a clean state, the first thing to do is to create a New Test. This may be done by invoking the New Test wizard from the context menu on the Explorer, from the File menu or from the respective button on the toolbar, then filling in the required options and configurations.

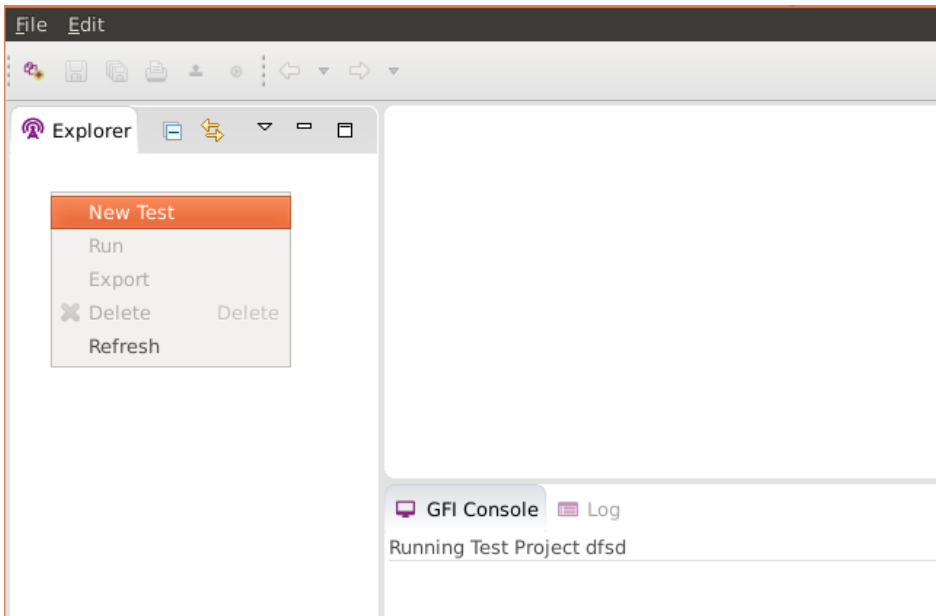


Figure 6 - New Test

Properties

This dialog gathers the information required for the test Properties, the sequence of use cases needed to verify a given functionality. From this inputs, the file referred to as <execution-file>.xml will be generated.

4.3.1.1.1 ID

This is the identifier that will be used for the execution file name and shown on the Explorer pane.


4.3.1.1.2 Name

A user defined name, typically a short but meaningful designation.

4.3.1.1.3 Description

The detail used to include in the test purpose section of the test report.

See example in Figure 7.

New Test properties 

Create a new Test and define Test properties

* All fields required

Identifier:

Test Name:


Description:

Figure 7 - New Test Properties

Scenario

This dialog gathers the information required for the test scenario, all the configurations for the networking, outputs, scheduling, equipment, etc.. From these inputs, the file referred to as <scenario-file>.xml will be generated.

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

Scenario selection 

Define Test Scenario properties

Scenario:

Active Meter:

ESME (Electric Smart Meter)


GSME (Gas Smart Meter)

Figure 8 - Scenario selection (emulators)

If an actual metering device is used, the meter configurations need to be properly set.

Scenario selection

Define Test Scenario properties



Scenario:

** All fields required*

Active Meter:

ESME (Electric Smart Meter)

GSME (Gas Smart Meter)

Id:

Installation Credentials:

Digital Signature Certificate:

Key Agreement Certificate:

CHFE

MAC:

ZigBee Network

Pan Id:

Extend Pan Id:

Network Key:

Port:

Join TimeOut: seconds

Figure 9 - Scenario selection (physical meters)

- ID: Entity Identifier of the Device in the Whitelist of the CHF.
- Installation Credentials: Installation credentials (installation code) of the Device in the Whitelist of the CHF.
- Digital Signature Certificate: Path to the .der file containing the device certificate.
- Key Agreement Certificate: Path to the .der file containing the device certificate.

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

- Pan ID: The ZigBee network Pan ID
- Extended Pan ID: The ZigBee network extended Pan ID

- Network Key: The ZigBee network key
- Port: The serial port where the ZigBee Adapter is connected.
- Join Timeout: The join timeout, specified in seconds.

On the first use this dialog's fields are shown blank (as in Figure 9) but after the first Test is created, on yet another New Test, the fields are populated with data from the last configuration. Using the button 'Restore Defaults' will revert to that same last configuration (from a successfully added 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 in the test library – the SMETS Use Cases already coded.

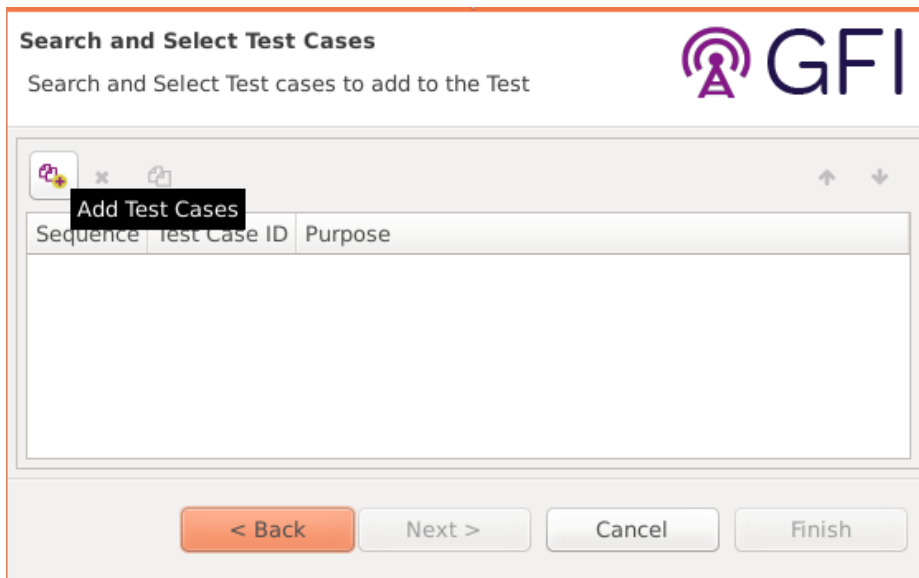


Figure 10 - Search and Select Test Cases

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 11. Upon selection, the information of a test-case is displayed in the pane on the right 'Selected test case info'. Also, multiple selection is allowed for addition.

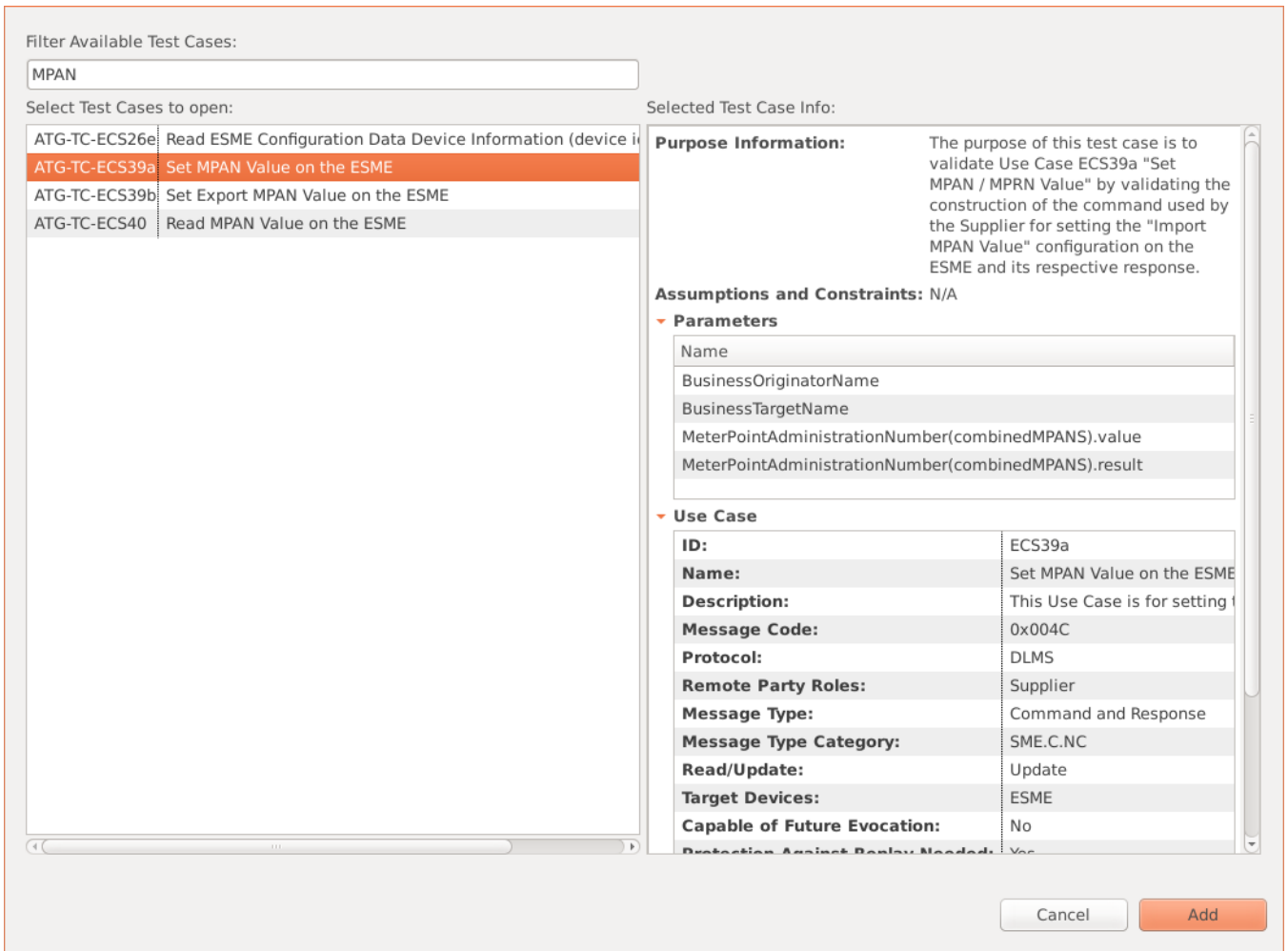


Figure 11 - Add Test Case

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

At this point tools are available to tune the Test procedure. On the upper right corner, the arrow icon buttons ('Move Up' and 'Move Down') allow a selected test case to be moved up or down the sequence. On the upper left corner, next to the 'Add Test Cases' button, lay the 'Delete' and the 'Duplicate' buttons. As their names imply, they allow the possibility to remove or duplicate one or more test cases as multiple selection is permitted for these functions. 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.

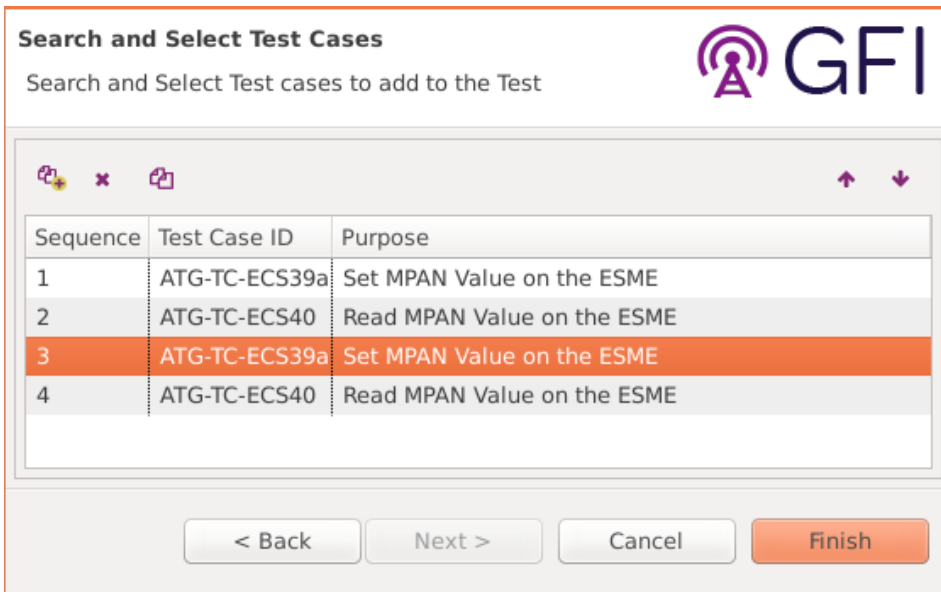


Figure 12 - New Test populated sequence

Clicking 'Finish' in this dialog will create the new test which will then be listed in the GUI main window's Explorer. Properties and Scenario (XML files) 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 4.3.3.

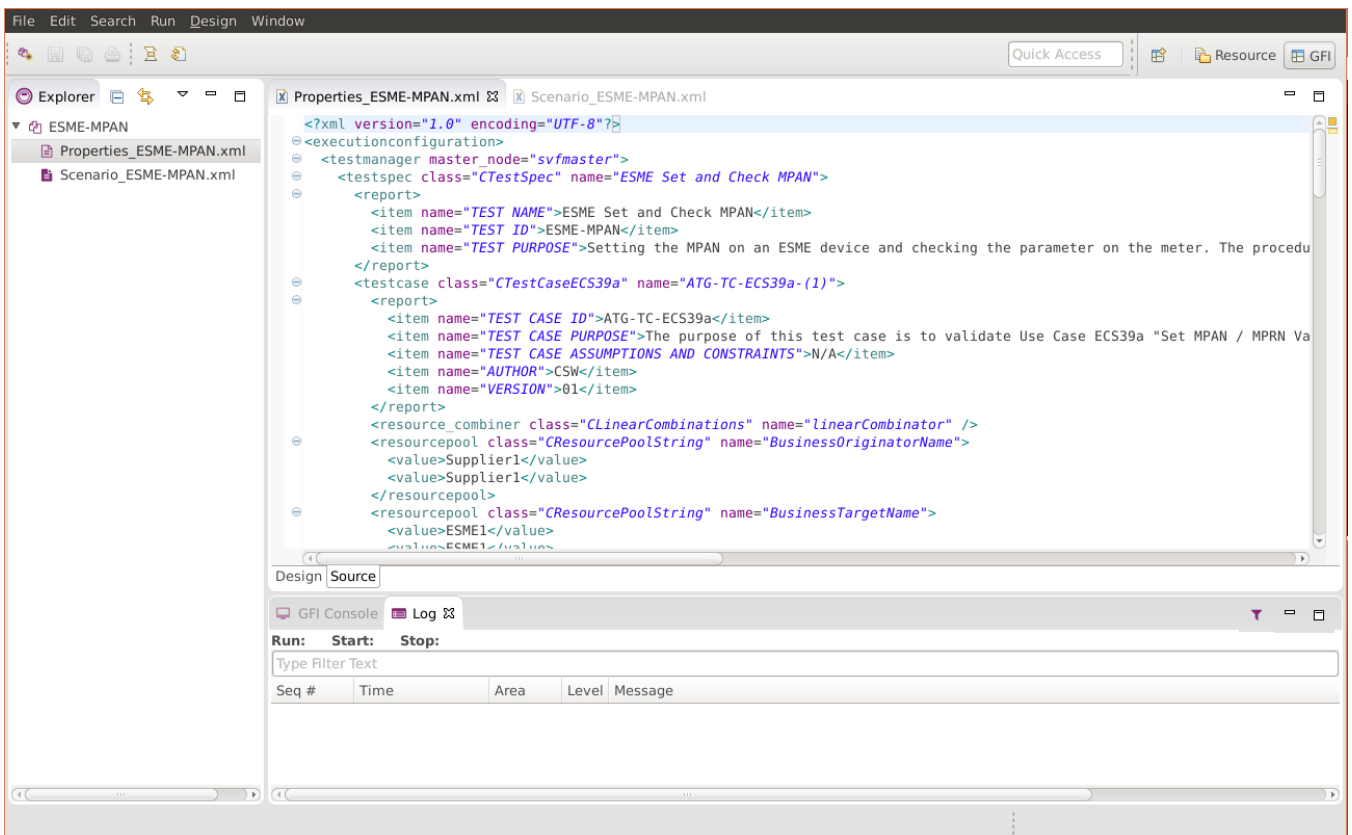


Figure 13 - New Test added

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.

Each input or output in a test case is defined as a resource (as it could be used in any possible way) and included in a <resourcepool> element 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, on 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 4.3.3 a practical example is presented.

4.3.2 Test Execution

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

A Test previously created may be executed through the 'Run' button in the toolbar or the option 'Run' in the Test's context menu.

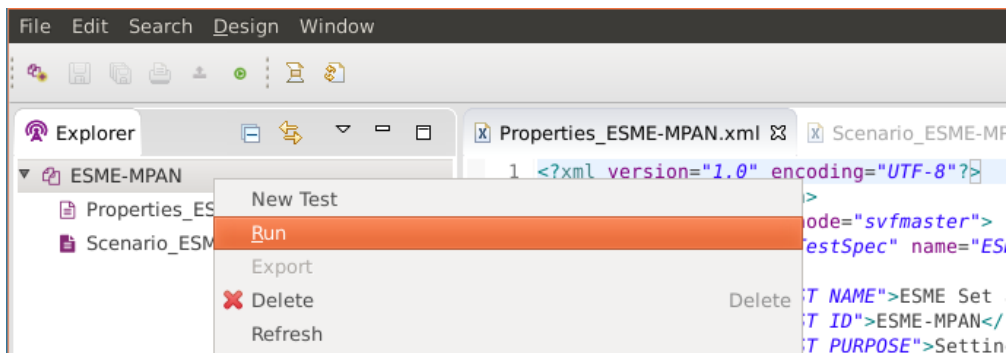


Figure 14 - Run Test from context menu

After clicking OK in the confirmation box the Test is executed.

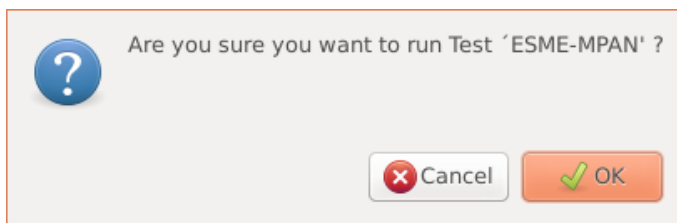


Figure 15 - Test execution confirmation

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

```

GFI Console Log Console
Running Test Project ESME-MPAN
Found 1st argument. Searching for file [/home/gfiuser/runtime-EclipseApplication/MPAN/Scenario_MPAN.xml].
Found 2nd argument. Searching for file [/home/gfiuser/runtime-EclipseApplication/MPAN/Properties_MPAN.xml].

-----
Initializing the Software Database (SDB)...
Starting VSIS SVF Execution...
-----
STARTING EXECUTION TIME: 2015-06-18 19:03:30 (WEST)
-----
  TIME | AREA | LV | MESSAGE
-----
00:00:00.243 | MNGE | 0 | Preparing Test Execution...
00:00:00.244 | RPORT | 4 | Report location: /home/gfiuser/.gui/tmp/test-report.html
00:00:00.258 | ACB | 4 | ACBEmulator: Start processing thread for messages received from the KRP
00:00:00.258 | ACB | 4 | ACBEmulator: Waiting for new messages from KRP
00:00:00.258 | ACB | 4 | ACBEmulator: Start processing thread for messages received from xSME
00:00:00.258 | ACB | 4 | ACBEmulator: Waiting for new messages from xSME
00:00:00.259 | EQUIP | 4 | CHFEmulator: Start processing thread
00:00:00.259 | EQUIP | 4 | CHFEmulator: Waiting for new messages
00:00:00.275 | EQUIP | 4 | GSMEmulator: Start processing thread
00:00:00.275 | EQUIP | 4 | GSMEmulator: Waiting for new messages
00:00:00.276 | EQUIP | 4 | ESMEEmulator: Start processing thread
00:00:00.276 | EQUIP | 4 | ESMEEmulator: Waiting for new messages
00:00:00.277 | MNGE | 0 | Starting Test Execution...
00:00:00.277 | MNGE | 0 | Starting Run 1 of 1...
00:00:00.277 | MNGE | 0 | Executing Run 1 of 1...
00:00:00.277 | MNGE | 2 | Testcase: ATG-TC-ECS39a-(1), Starting...

```

Figure 16 - Console activity

When the execution is finished, the execution log is presented in the Log pane and the report shown in the working area. Also, a new Campaign item regarding the Test's execution is added to the Explorer pane along with all the respective configuration and execution files.

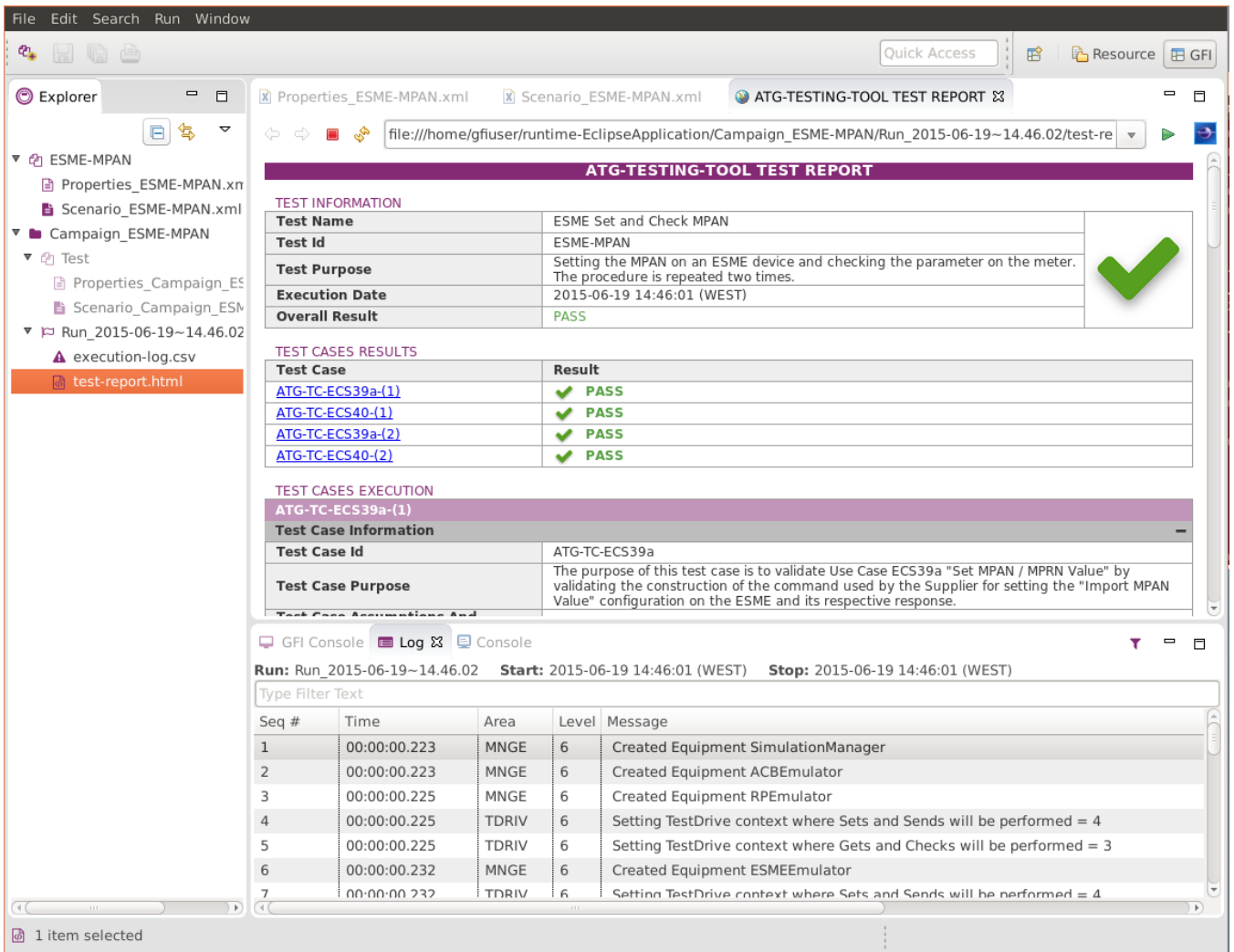


Figure 17 - Execution finished

This concludes the Test execution. As mentioned before, should any further executions of this Campaign be performed, new 'Run' items will be added to the Campaign. Old runs may be deleted but are kept by default for later analysis.

Note: If a Test is executed, a new Campaign is generated. If a Campaign is executed, a new Run (in 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 18 depicts multiple executions of a Campaign.

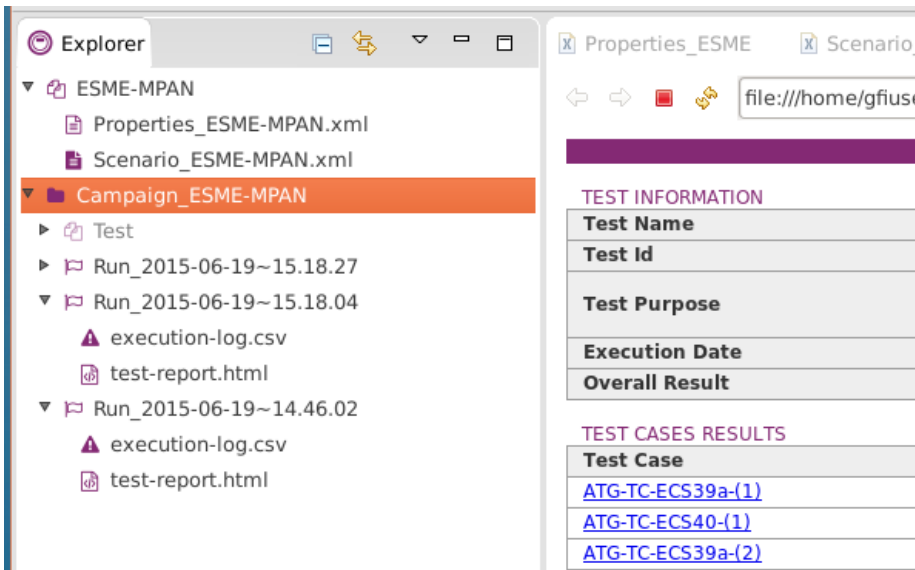


Figure 18 - Multiple Runs

4.3.3 Results Analysis

In the previous section the Test was executed without any changes to the original test cases in the library, so the result was PASS for all of them (the default values were previously tested and known to yield pass results).

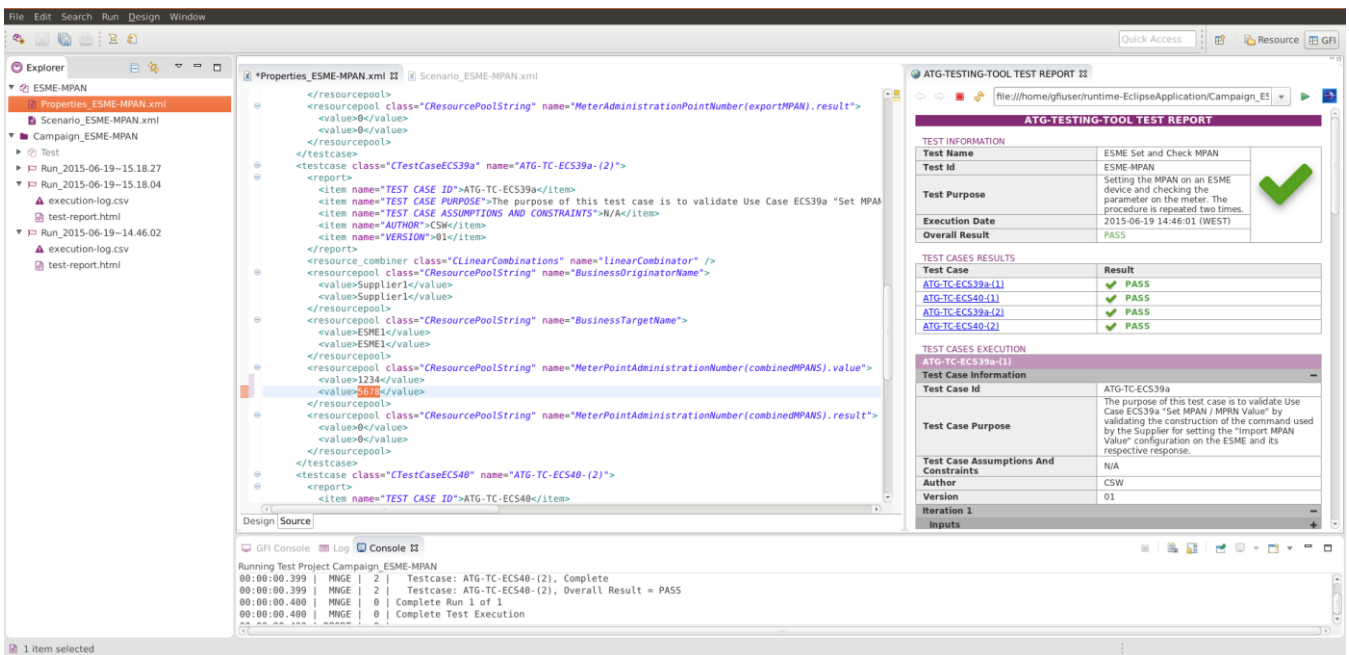


Figure 19 – IDE layout (Properties and Report)

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

This test case runs two iterations as there are two data values per resource pool. 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 go about it:

- Remove the redundant iterations – 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 set the value of the resource pool that relates to the parameter requiring change, in the last iteration. Values set in previous iterations will be overwritten in the last one.

Going with the second option, only the value highlighted in Figure 20 needs to be changed.

```

<testcase class="CTestCaseECS39a" name="ATG-TC-ECS39a-(2)">
  <report>
    <item name="TEST CASE ID">ATG-TC-ECS39a</item>
    <item name="TEST CASE PURPOSE">The purpose of this test case is to validate Use Case ECS39a "Set MPAN
    <item name="TEST CASE ASSUMPTIONS AND CONSTRAINTS">N/A</item>
    <item name="AUTHOR">CSW</item>
    <item name="VERSION">01</item>
  </report>
  <resource_combiner class="CLinearCombinations" name="linearCombinator" />
  <resourcepool class="CResourcePoolString" name="BusinessOriginatorName">
    <value>Supplier1</value>
    <value>Supplier1</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="BusinessTargetName">
    <value>ESME1</value>
    <value>ESME1</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterPointAdministrationNumber(combinedMPANS).value">
    <value>1312345678333</value>
    <value>1312345678111312345678222</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterPointAdministrationNumber(combinedMPANS).result">
    <value>0</value>
    <value>0</value>
  </resourcepool>
</testcase>
<testcase class="CTestCaseECS40" name="ATG-TC-ECS40-(2)">
  <report>
    <item name="TEST CASE ID">ATG-TC-ECS40</item>
    <item name="TEST CASE PURPOSE">The purpose of this test case is to validate Use Case ECS40 "Read MPAN
  
```

Figure 20 – Edit Test Properties - before

After assigning different values to that resource pool, the XML should look like Figure 21.

```
*Properties_ESME-MPAN.xml Scenario_ESME-MPAN.xml
</resourcepool>
<resourcepool class="CResourcePoolString" name="MeterAdministrationPointNumber(exportMPAN).result">
  <value>0</value>
  <value>0</value>
</resourcepool>
</testcase>
<testcase class="CTestCaseECS39a" name="ATG-TC-ECS39a-(2)">
  <report>
    <item name="TEST CASE ID">ATG-TC-ECS39a</item>
    <item name="TEST CASE PURPOSE">The purpose of this test case is to validate Use Case ECS39a "Set MPAN<
    <item name="TEST CASE ASSUMPTIONS AND CONSTRAINTS">N/A</item>
    <item name="AUTHOR">CSW</item>
    <item name="VERSION">01</item>
  </report>
  <resource_combiner class="CLinearCombinations" name="linearCombinator" />
  <resourcepool class="CResourcePoolString" name="BusinessOriginatorName">
    <value>Supplier1</value>
    <value>Supplier1</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="BusinessTargetName">
    <value>ESME1</value>
    <value>ESME1</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterPointAdministrationNumber(combinedMPANS).value">
    <value>1234</value>
    <value>5678</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterPointAdministrationNumber(combinedMPANS).result">
    <value>0</value>
    <value>0</value>
  </resourcepool>
</testcase>
<testcase class="CTestCaseECS40" name="ATG-TC-ECS40-(2)">
  <report>
    <item name="TEST CASE ID">ATG-TC-ECS40</item>
```


Figure 21 - Edit Test Properties - after

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.

ATG-TESTING-TOOL TEST REPORT

TEST INFORMATION

Test Name	ESME Set and Check MPAN	
Test Id	ESME-MPAN	
Test Purpose	Setting the MPAN on an ESME device and checking the parameter on the meter. The procedure is repeated two times.	
Execution Date	2015-06-19 16:10:04 (WEST)	
Overall Result	FAIL	

TEST CASES RESULTS

Test Case	Result
ATG-TC-ECS39a-(1)	✓ PASS
ATG-TC-ECS40-(1)	✓ PASS
ATG-TC-ECS39a-(2)	✓ PASS
ATG-TC-ECS40-(2)	✗ FAIL

TEST CASES EXECUTION

ATG-TC-ECS39a-(1)	
Test Case Information	
Test Case Id	ATG-TC-ECS39a
Test Case Purpose	The purpose of this test case is to validate Use Case ECS39a "Set MPAN / MPRN Value" by validating the construction of the command used by the Supplier for setting the "Import MPAN Value" configuration on the ESME and its respective response.
Test Case Assumptions And Constraints	N/A
Author	CSW
Version	01
Iteration 1	
Inputs	+
Execution	+
Results Summary	-

Figure 22 - Test FAIL

TIME	TYPE	DESCRIPTION	ACTUAL	EXPECTED	RESULT	
00:00:00.385	CHECK	ECS40_RESPONSE.PAYLOAD.ACCESS_RESPONSE_BODY.ACCESS_RESPONSE_SPECIFICATION.REQUEST_RESULTS.METER_ADMINISTRATION_POINT_NUMBER_EXPORT_MPAN.ACCESS_RESPONSE_TYPE	1	equal to	1	PASS
00:00:00.385	CHECK	ECS40_RESPONSE.PAYLOAD.ACCESS_RESPONSE_BODY.ACCESS_RESPONSE_LIST_OF_DATA.REQUEST_RESPONSES.METER_ADMINISTRATION_POINT_NUMBER_COMBINED_MPANS	5678	equal to	13123456781111 312345678222	FAIL
00:00:00.385	CHECK	ECS40_RESPONSE.PAYLOAD.ACCESS_RESPONSE_BODY.ACCESS_RESPONSE_LIST_OF_DATA.REQUEST_RESPONSES.METER_ADMINISTRATION_POINT_NUMBER_EXPORT_MPAN	1312345678333	equal to	1312345678333	PASS
00:00:00.385	CHECK	ECS40_RESPONSE.PAYLOAD.ACCESS_RESPONSE_BODY.ACCESS_RESPONSE_SPECIFICATION.REQUEST_RESULTS.METER_ADMINISTRATION_POINT_NUMBER_COMBINED_MPANS.RESULT	0	equal to	0	PASS
00:00:00.385	CHECK	ECS40_RESPONSE.PAYLOAD.ACCESS_RESPONSE_BODY.ACCESS_RESPONSE_SPECIFICATION.REQUEST_RESULTS.METER_ADMINISTRATION_POINT_NUMBER_EXPORT_MPAN.RESULT	0	equal to	0	PASS
Results Summary						
Overall Iteration Expected Results Matched 96.3% (26/27)						
Result	✗ FAIL					

Figure 23 - Test FAIL detail

Changes need to be made so the check verifies the last MPAN value set in the equipment, it should be the value set in the last iteration of the preceding update test case.

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

Figure 24 shows how the XML Properties file should look like after resource pool is changed to hold the correct expected values (regarding the last test case (ATG-TC-ECS40-(2))).

```

<testcase class="CTestCaseECS40" name="ATG-TC-ECS40-(2)">
  <report>
    <item name="TEST CASE ID">ATG-TC-ECS40</item>
    <item name="TEST CASE PURPOSE">The purpose of this test case is to validate Use Case ECS40 "Read MPAN
    <item name="TEST CASE ASSUMPTIONS AND CONSTRAINTS">GIST Issue #3145: "ECS40 - Notes for attribute Met
    <item name="AUTHOR">CSW</item>
    <item name="VERSION">01</item>
  </report>
  <resource_combiner class="CLinearCombinations" name="linearCombinator" />
  <resourcepool class="CResourcePoolString" name="BusinessOriginatorName">
    <value>ACB</value>
    <value>ACB</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="BusinessTargetName">
    <value>ESME1</value>
    <value>ESME1</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="SupplementaryRemotePartyName">
    <value>Supplier1</value>
    <value />
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterAdministrationPointNumber(combinedMPANS).value">
    <value>5678</value>
    <value>5678</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterAdministrationPointNumber(exportMPAN).value">
    <value>1312345678333</value>
    <value>1312345678333</value>
  </resourcepool>
  <resourcepool class="CResourcePoolString" name="MeterAdministrationPointNumber(combinedMPANS).result">
    <value>0</value>
    <value>0</value>
  </resourcepool>

```

Figure 24 - 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. Meaning that the parameter was updated successfully.

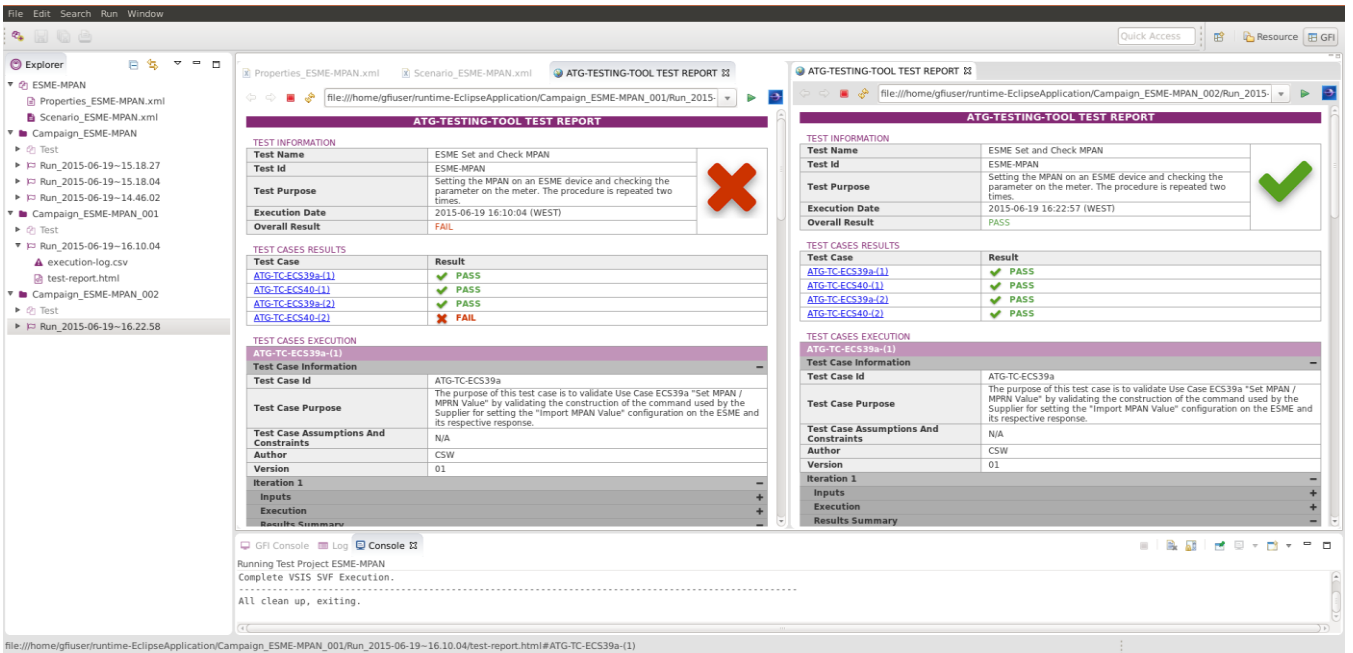


Figure 25 - Test reports side by side

As shown in Figure 25, a new Campaign is added for every Test run – The Test had to be run (not the Campaign) as the test cases in the sequence were subject changes (not allowed in the Campaign).

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

Logging

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

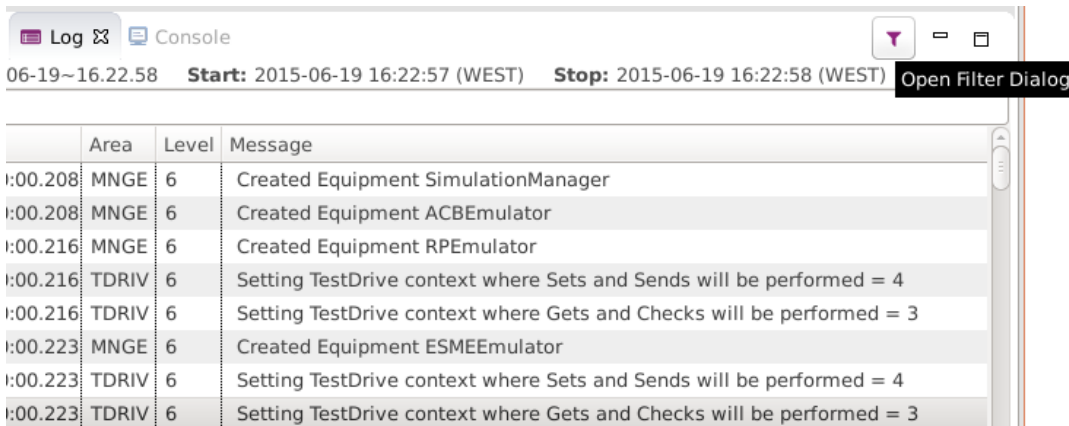


Figure 26 - Open Filter Dialog button

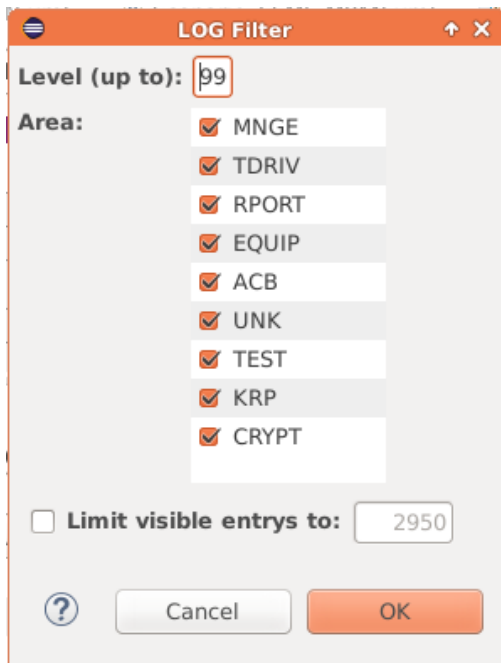


Figure 27 - Log Filter

This will facilitate manual checks on exchanged messages. The filter applies to the log level and the area 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.

Export

It is possible to export the files of a text execution for archiving purposes. This action may be invoked from the 'Run' context menu. Dialogs will be displayed to select the output path for the produced files.

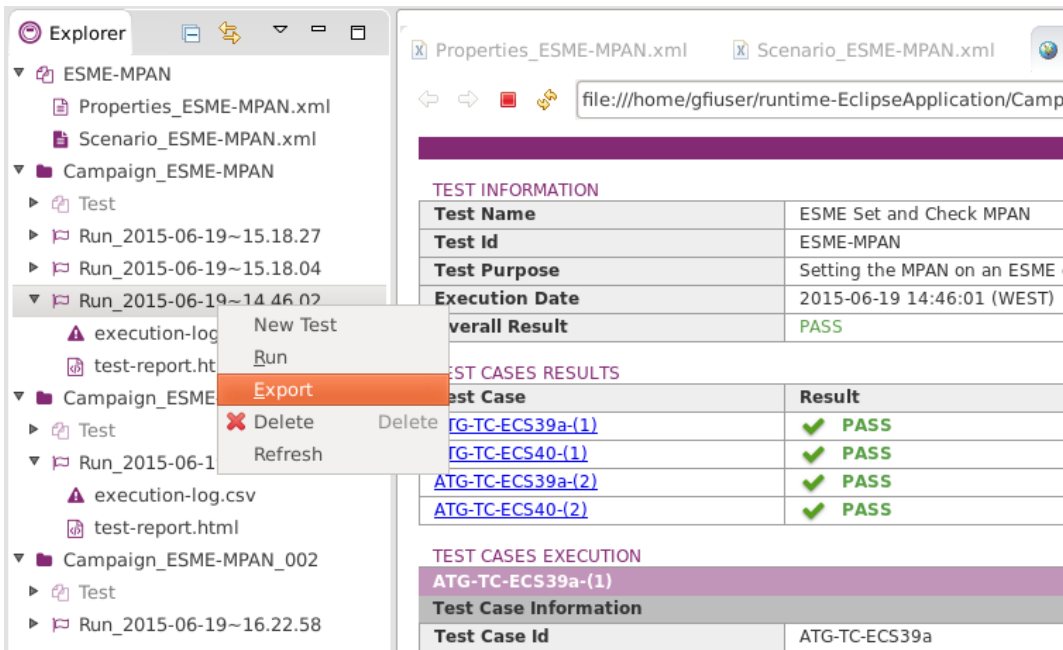


Figure 28 - Export Test Run

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

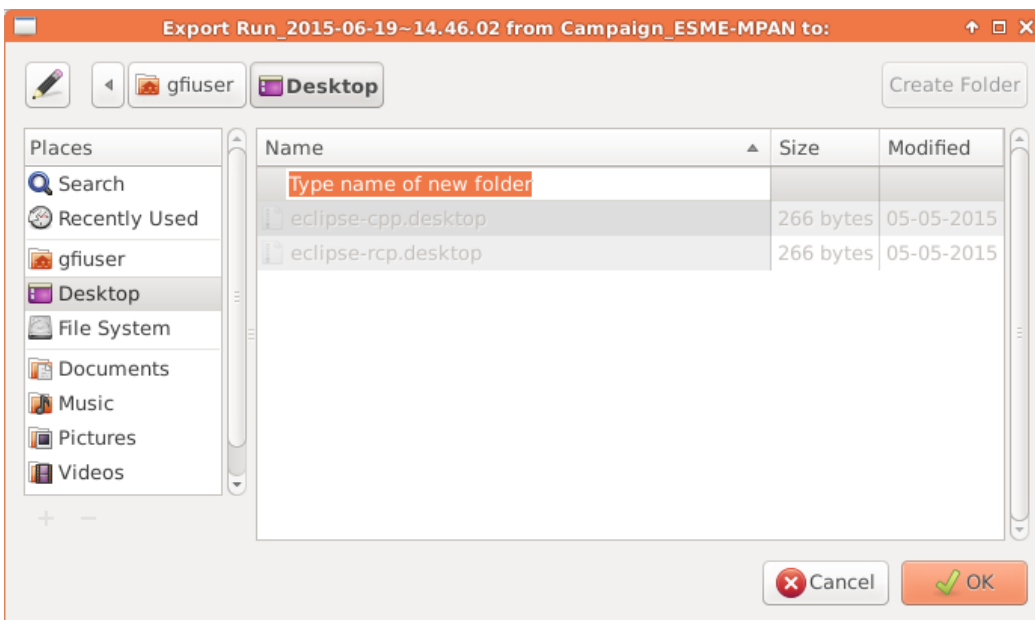


Figure 29 - Select output path

Generated files should then be available in the selected location.

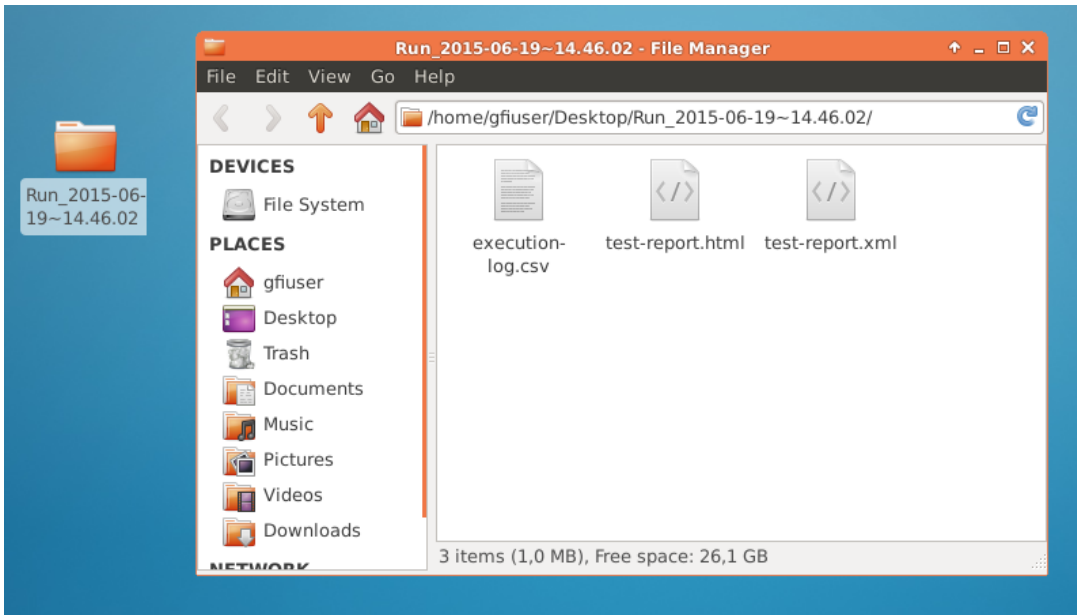


Figure 30 - Exported files

Along with the execution log and the test report (HTML) an XML report file is also generated should any report integration tool require an unformatted report.

5 Glossary

Table 2 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 2 - Definitions

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

Acronym	Description
ACB	Access Control Broker
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
GSME	Gas Smart Metering Equipment
KRP	Known Remote Party
NA	Not Applicable
PWO	Pass With Observations
RD	Reference Document
TBC	To be confirmed
TBD	To be defined
UTRN	Unique Transaction Reference Number
ZCL	ZigBee Cluster Library

Table 3 - Acronyms