Visure
the requirements company

www.visuresolutions.com

Test Management Extension User Guide

December 2011

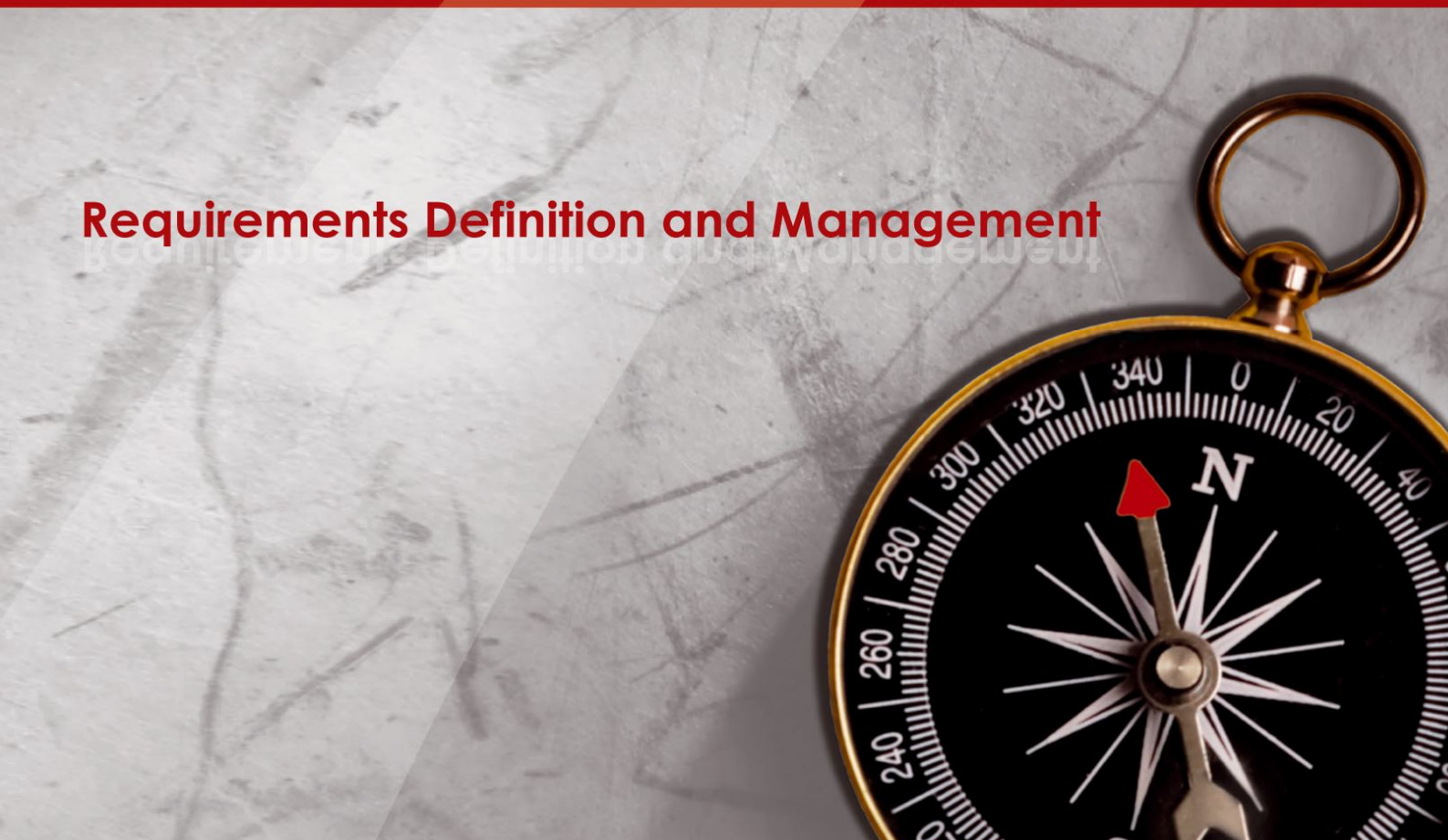


TABLE OF CONTENTS

1 Introduction

This document provides a description of the Test Management Extension in IRQA.

The purpose of this extension is to add capabilities to IRQA to support Test Management. These capabilities include:

- Test Session Creation
- Test Session Execution (via test scripts)
- Test Status update
- Test Scenarios View Columns

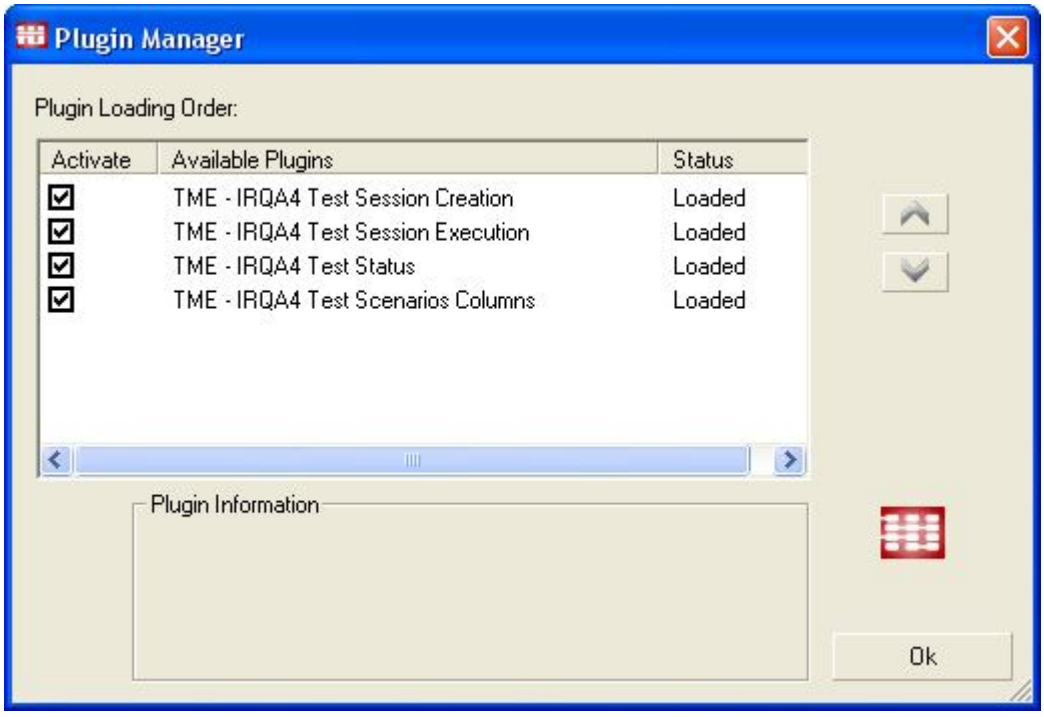
2 Installation

The Test Management Extension is made up of five files:

- TME_TestSession.irqa
- TME_TestSessionExecution.irqa
- TME_TestStatus.irqa
- TME_TestScenariosColumns.irqa
- TMEConfig.xml

All files need to be copied to the Plugins folder inside the IRQA Installation folder (usually [HDD]:\Program files\IRQA 4\) or the project's "Corporate Plugin Folder". The files need to be directly copied to the Plugins folder (no subfolder).

Open the IRQA 4 Client, then open a project and navigate to the menu "Project/Plugins" to open the "Plugin Manager". Make sure the TME plugins are activated. If they are not activated, check them, press OK and restart the IRQA 4 Client.



2.1 IRQA Support

The test management extension can be used with IRQA 4.3 and up.

2.2 MFC Security Update

In order to use the extension, the "Microsoft Visual C++ 2005 Service Pack 1 Redistributable Package" update (KB2538242), from the Microsoft Security Bulletin MS11-025, must be installed.

You can get it from this website: http://technet.microsoft.com/en-us/security/bulletin/ms11-025

3 Test Management Extension Description

The Test Management Extension is made of four main modules:

- **Test Session Creation:** Module that lets users instantiate a test session from a test plan.
- **Test Session Execution:** Module that lets users execute test scripts associated to a test session and update test results.
- **Test Status Columns:** Module that adds test status columns to the Requirement view.
- **Test Scenarios Columns:** Module that adds test scenario properties columns to the test scenarios view.

3.1 Test Session Creation

The test session creation module lets users instantiate test cases, part of a test plan, into test run objects, part of a test session.

As stated in the "Standard glossary of terms used in Software Testing" from the International Software Testing Qualifications Board (ISTQB), a test session can be defined by:

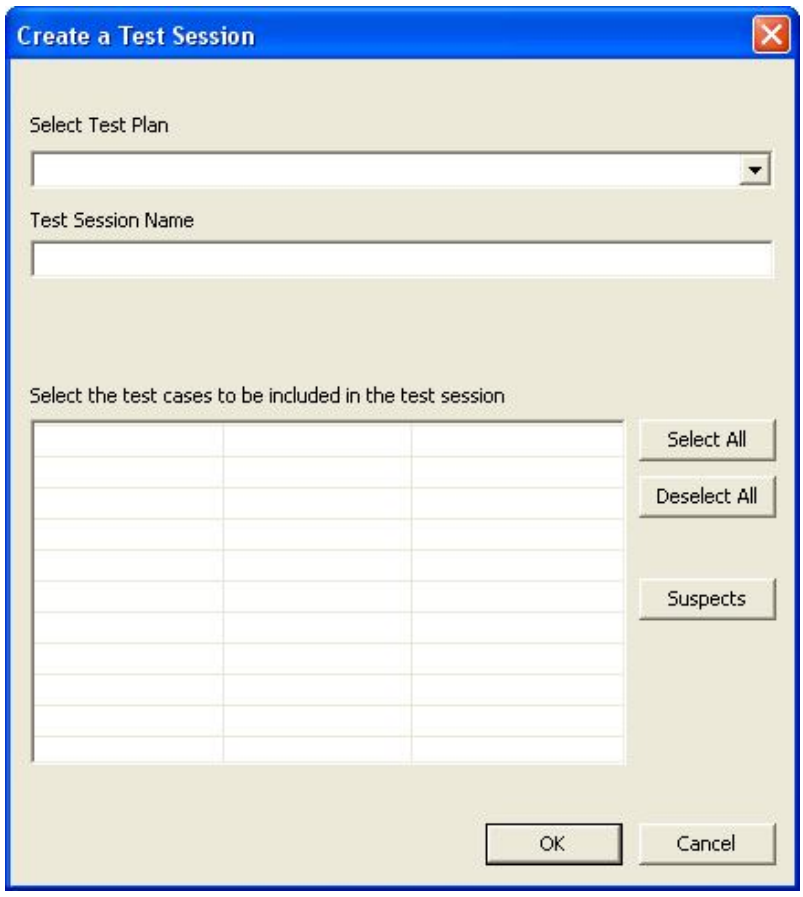
> *An uninterrupted period of time spent in executing tests. In exploratory testing, each test session is focused on a charter, but testers can also explore new opportunities or issues during a session. The tester creates and executes test cases on the fly and records their progress.*

A test run can be defined by:

> *Execution of a test on a specific version of the test object.*

The test session creation module adds a new toolbar to the test scenarios view.

This toolbar contains one button that, when clicked, brings a dialog to create a test session based on a test plan.

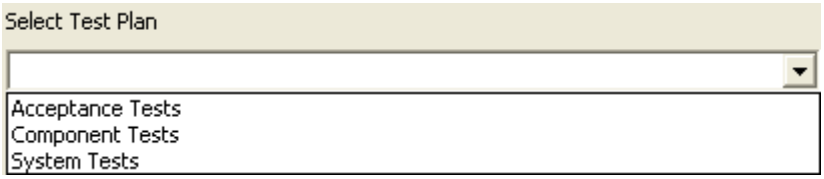


To create a Test Session, users need to first select a Test plan. Test plans are based on test scenario blocks. The "Select Test Plan" drop-down menu gives a list of all test scenario blocks in the project. The list of test scenarios blocks does not include the test scenarios blocks used to store the test run objects.

For-example, from this list of project blocks:

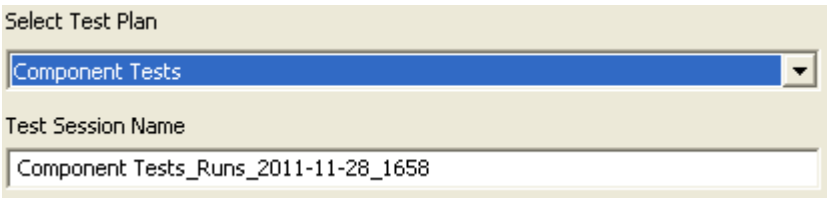
Name	Type of elements
Acceptance Tests	Test Scenarios
Component Requirements	Requirements
Component Tests	Test Scenarios
Component Tests_Runs_2011-11-02_1237	Test Scenarios
Customer Requirements	Requirements
Defects	Requirements
Electrical Requirements	Requirements
Marketing Requirements	Requirements
Mechanical Requirements	Requirements
Runs	Test Scenarios
Software Requirements	Requirements
Stakeholder Requirements	Requirements
Standards	Requirements
System Requirements	Requirements
System Tests	Test Scenarios
Use Cases	Services

The available test plans are:

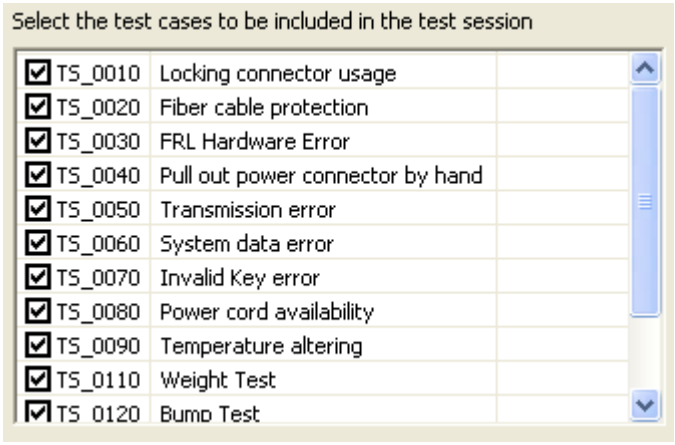


The "Runs" and "Component Tests_Runs_2011-11-02_1237 blocks are test runs blocks that contains test run objects that track the results of the execution of a test scenario defined in a test plan (see section 4.2.1.2 on how to configure the module to recognize Runs block).

Once the user selects a test plan, the module automatically generates a test session name. This test session name can be modified if needed. However, be aware to keep the string used to recognize Runs blocks as part of the name (see section 4.2.1.2 on how to configure the module to recognize Runs block).



The selection of the test plan also populates the list of test cases that will be included in the test session. The list contains all test cases associated to the test plan (i.e. test scenarios associated to the selected block).



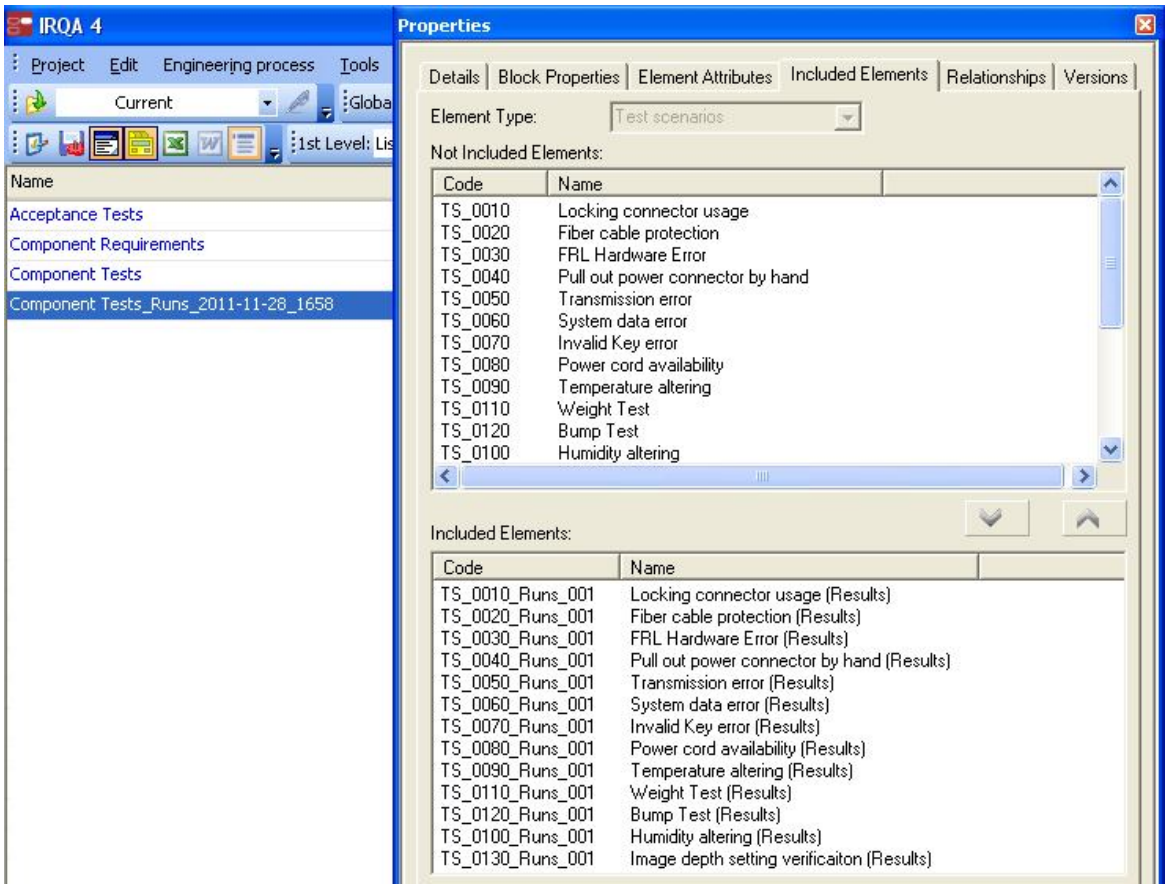
Users can select any test cases to be included in the Test Session. If the suspect links are enabled in the project, the Suspects button is enabled and when clicked, selects all test cases that have a suspect link to a requirement.

When the button "OK" is clicked, it creates one test run object (that will be used to store execution results) for every selected test case.

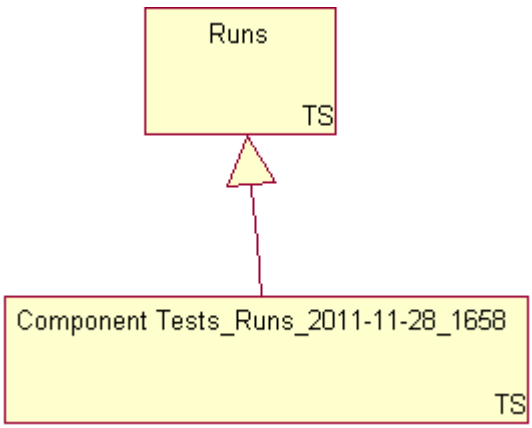
Code	Name	Test Status
TS_0010 (1)	Locking connector usage	
TS_0010_Runs_001	Locking connector usage (Results)	Pending
TS_0020 (1)	Fiber cable protection	
TS_0020_Runs_001	Fiber cable protection (Results)	Pending
TS_0030 (1)	FRL Hardware Error	
TS_0030_Runs_001	FRL Hardware Error (Results)	Pending
TS_0040 (1)	Pull out power connector by hand	
TS_0040_Runs_001	Pull out power connector by hand (Results)	Pending
TS_0050 (1)	Transmission error	
TS_0050_Runs_001	Transmission error (Results)	Pending
TS_0060 (1)	System data error	
TS_0060_Runs_001	System data error (Results)	Pending
TS_0070 (1)	Invalid Key error	
TS_0070_Runs_001	Invalid Key error (Results)	Pending
TS_0080 (1)	Power cord availability	
TS_0080_Runs_001	Power cord availability (Results)	Pending
TS_0090 (1)	Temperature altering	
TS_0090_Runs_001	Temperature altering (Results)	Pending
TS_0100 (1)	Humidity altering	
TS_0100_Runs_001	Humidity altering (Results)	Pending
TS_0110 (1)	Weight Test	
TS_0110_Runs_001	Weight Test (Results)	Pending
TS_0120 (1)	Bump Test	
TS_0120_Runs_001	Bump Test (Results)	Pending
TS_0130 (1)	Image depth setting verification	
TS_0130_Runs_001	Image depth setting verificaiton (Results)	Pending

The format of the test run object (Code, Name, traceability to tests, partition, etc.) is configured via Test Management Extension configuration file. See section 4.2.2 for more details.

These test run object are associated to the test session block (as specified by the test session name) that is created by the extension.



The test session creation module also creates a hierarchical link between the test session block and the predefined "Runs" block (see section 4.2.1.2 on how to configure the module to find the predefined "Runs" block). This allows test run objects to inherit attributes defined in the "Runs" block (e.g.: Test Status).

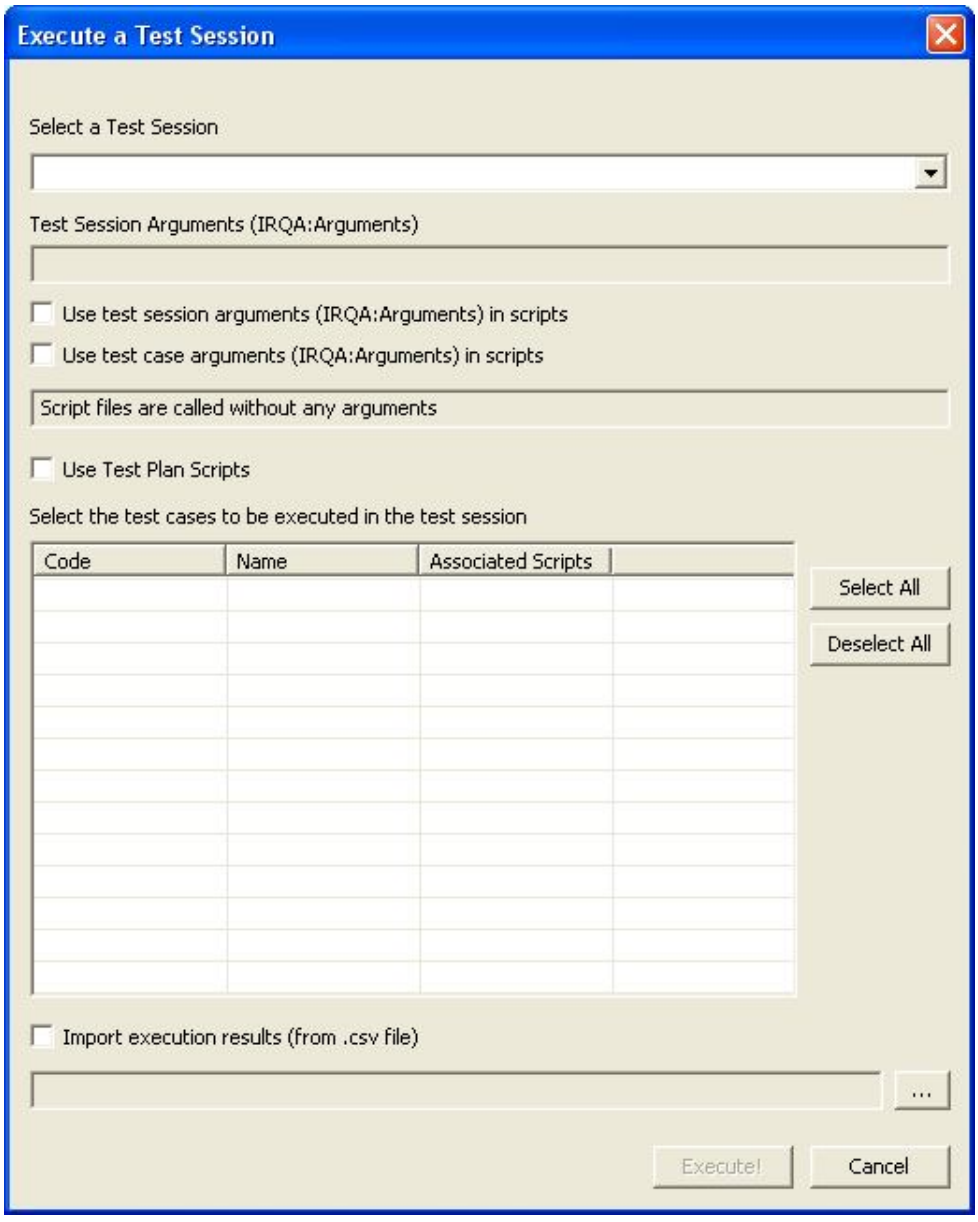


3.2 Test Session Execution

The test session execution module lets users execute test scripts associated to test run objects (part of a test session) and update test results. The update of the test result of a test run can be done manually by the users or automatically by the module.

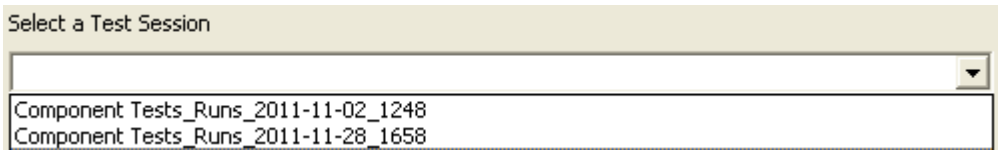
The test session execution module adds a new toolbar to the test scenarios view.

This toolbar contains one button that, when clicked, brings a dialog to execute the scripts associated to test run objects that are part of a selected test session.

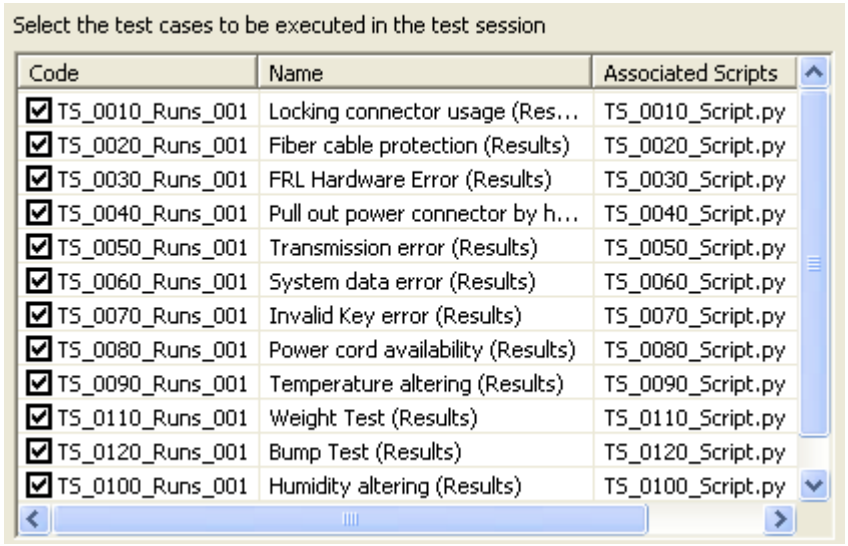


3.2.1 Test scripts execution

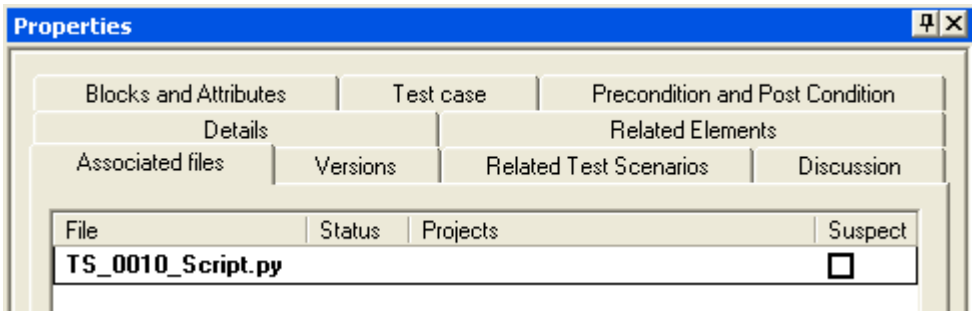
To execute test scripts, users need to first select a Test session (see section 3.1 on how to create a test session). The "Select Test Session" drop-down menu gives a list of all test session in the project.



The selection of the test session populates the list of test run objects that are part of the test session. Users can select which test runs will be executed.



The list displays up to four columns. The code and name of the test run object, the associated scripts and any attribute that will be used as argument to the scripts (see section 3.2.2). The associated scripts are any associated files to the test run that are detected as being scripts. Currently, any python script file with the ".py" extension, is detected as a valid script to be executed (see section 3.2.4 on python scripts).



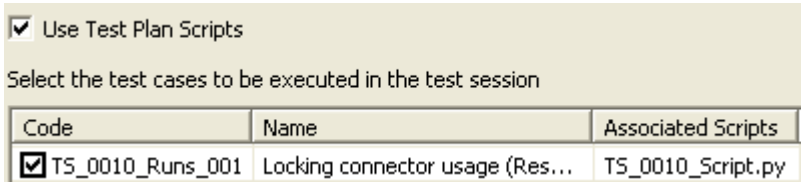
Multiple scripts can be attached to one test. These are executed in sequence (in the order of appearance in the Associated Files tab).

The test session execution module can execute scripts associated to the test run object or scripts associated to the parent test cases. The option "Use test plan scripts" allows the users configuring this behavior.

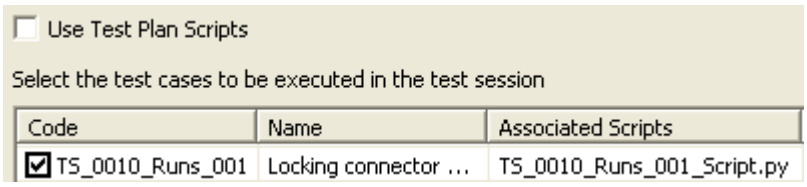
For-example:

Code	Name	Associated Files
TS_0010 (1)	Locking connector usage	TS_0010_Script.py
TS_0010_Runs_001	Locking connector usage (Results)	TS_0010_Runs_001_Script.py

When the option is checked, the scripts associated to the parent test case (here TS_0010) are used:



When the option is unchecked, the scripts associated to the test run object (here TS_0010_Runs_001) are used:



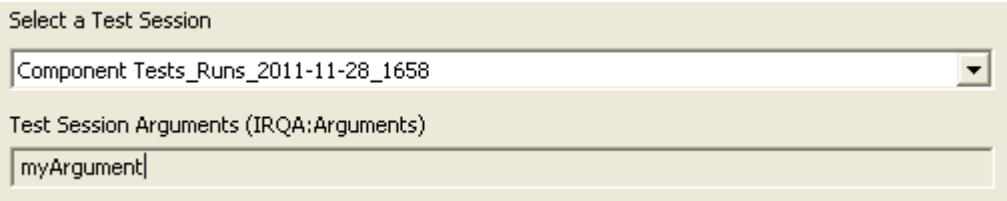
When clicking on OK, the test execution module executes all valid scripts of any of the selected test run objects.

3.2.2 Attributes as script arguments

A dedicated attribute can be used to specify arguments for the test scripts. There are two types of attributes/arguments. The first type is a test session argument. The test session argument is used in every script call. The test session argument is a predefined block property (see section 4.2.3.3 on how to configure the extension to retrieve the right block property) of type TEXT.

Name	IRQA:Arguments
Component Tests_Runs_2011-11-02_1248	myArgument
Component Tests_Runs_2011-11-28_1658	myArgument

In this example, the block property "IRQA:Arguments" is used as the test session argument (applied to every scripts).



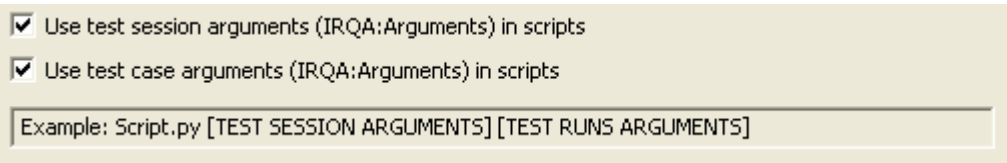
The second type of arguments is the test run object argument. This argument is set via a test run TEXT attribute. This argument is applied to the individual scripts associated to the test run object.

Code	Name	IRQA:Arguments
TS_0010 (1)	Locking connector usage	
TS_0010_Runs_001	Locking connector usage (Results)	myTestRunArgument
TS_0020 (1)	Fiber cable protection	
TS_0020_Runs_001	Fiber cable protection (Results)	myTestRunArgument2

In this example, the attribute "IRQA:Arguments" is used as argument for each individual test run object (applied to every scripts associated to the test run object).

Code	Name	Associated Scri...	IRQA:Arguments
TS_0010_Runs_001	Locking c...	TS_0010_Scrip...	myTestRunArgument
TS_0020_Runs_001	Fiber cab...	TS_0020_Scrip...	myTestRunArgument2

The options "Use test session arguments" and "Use test case arguments" are used to enable or disable the use of the arguments when calling the scripts. Users can see how the arguments will be used via the script call example shown on the dialog.

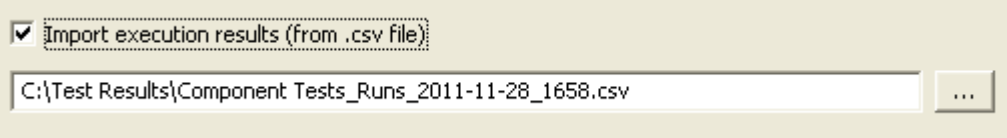


3.2.3 Test run results

Test run results can be manually updated after the execution of the tests. In this scenario, a tester manually or automatically (using test scripts) execute the tests and then manually update each test run object with the test results.

In another scenario, the test scripts can populate a .csv (comma-separated values) file with the test results (e.g.: pass/fail) and the test session execution module can parse this file and automatically update the test run results.

To use the results import facility, the test session execution module must be pointed to the csv file that will be populated by the test scripts as shown here:



The execution scripts are responsible to create the result file (one per session). The format of the .csv file must be:

```
Test_Run_Code, Test_Result
```

The "Test_Run_Code" is the code of the test run that is being executed. The "Test_Result" is the result of the execution. The format of the result needs to correspond to the test result attribute format used for the test run objects (see section 4.2.1.4 on how to configure the module with the right test result attribute). Each new test run entry (with test run code and test result) needs to be created on a new line in the csv file.

E.g.:

```
TS_0010_Runs_001,Pass
TS_0020_Runs_001,Pass
TS_0030_Runs_001,Fail
```

After the execution of all the test scripts, the test session execution module will parse the csv execution results file and update the test results attribute of all the test run objects that were executed accordingly.

3.2.4 Python scripts

The test session execution module can execute any python script. Python script are detected as being valid (i.e. can be executed) if their extension is ".py". In order to run the script properly, python must be installed. Python can be found here (http://www.python.org/download/).

The test session execution calls the python scripts on the command-line, using this syntax:

```
"[PythonPathAndExe]" "[PythonScript]" "[SelectedSession]" "[TestCode]" "[TestName]" "[ParentCode]" [SessionArg] [TestArg]
```

where:

- [PythonPathAndExe]: Python path + python exe as specified in the configuration file (see section 4.2.3.2)
- [PythonScript]: the name of the python script file that is being executed
- [SelectedSession]: the name of the test session that is being executed
- [TestCode]: The code of the test run object that is being executed
- [TestName]: The name of the test run object that is being executed
- [ParentCode]: The code of the parent of the test run object that is being executed
- [SessionArg]: The test session argument (if included)
- [TestArg]: The test run argument (if included)

E.g.:

```
"C:\Python27\python.exe" "TS_0010_Script.py" "Component Tests_Runs_2011-11-28_1658" "TS_0010_Runs_001" "Locking connector usage (result)" "TS_0010" mySessionArgument myTestArgument
```

Depending on the usage, it is up to the script to ignore or make use of any of the arguments passed by the test session execution module.

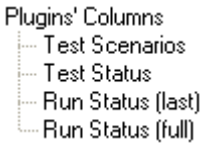
If results need to be imported back by the test session execution module, a csv file will need to be generated by the python scripts to store the test results. See section 3.2.3 for more details.

3.3 Test Status Columns

The test status columns module adds four columns to the requirement view to help users visualize traceability from requirements to test results. These four columns are:

- Test Scenarios : Display traceability to test scenarios
- Test Status : Display traceability to test scenarios with their test status
- Run Status (last): Display traceability to test scenarios and their last associated test run with their run status
- Run Status (full): Display traceability to test scenarios and all their associated test runs with their run status

The test status columns can be added via the Grid View Configuration of any requirements view, under the "Plugins' Columns" section:



3.3.1 Test Scenario Column

The "Test Scenario" column (which name can be modified via the configuration file, see section 4.2.4.2) displays traceability between the requirement and any test scenario. The column displays:

"Test Scenario Code" – "Test Scenario Name"

Code	Name	Test Scenarios
MechReq_0110	Fiber connector robustness	TS_0010 - Locking connector usage
MechReq_0120	Mechanical cladding on fiber	TS_0020 - Fiber cable protection TS_0025 - Fiber cable cladding

3.3.2 Test Status Column

The "Test Status" column (which name can be modified via the configuration file, see section 4.2.4.3) displays traceability between the requirement and any test scenario. The column also adds the test status information appended to each test scenario. To retrieve the test status, the column is using the predefined test status attribute (see section 4.2.1.3 to configure which test status attribute to use) that is associated to the test scenario. Furthermore, if the test status uses one of the predefined values, the text displayed in the column is colored. The predefined values (and their associated color) are:

- Pass or Passed (colored in GREEN)
- Fail or Failed (colored in RED)
- Pending (colored in YELLOW)
- Any other statuses are colored in BLACK

If multiple test scenarios are associated to the requirement, all tests are colored in RED if at least one of the tests failed, are colored in YELLOW if at least one test is pending (and none of the tests failed), are colored in GREEN if all tests were passed.

The column displays:

"Test Scenario Code" – "Test Scenario Name" [Test Result]

Code	Name	Test Status
FUNC_0015	CCC System modes	TST_040 - Self diagnostic test : [Failed]
FUNC_0020	Fully computerised control	TST_020 - Correct appliance connection : [Passed] TST_030 - Incorrect appliance connection : [Pending]
FUNC_0030	Full dual control	TST_010 - Connect appliance : [Passed]

In this last example, the test scenarios and their test status were:

Code	Name	Test Status
TST_010	Connect appliance	Passed
TST_020	Correct appliance connection	Passed
TST_030	Incorrect appliance connection	Pending
TST_040	Self diagnostic test	Failed

3.3.3 Run Status (last) Column

The "Run Status (last)" column (which name can be modified via the configuration file, see section 4.2.4.4) displays traceability between the requirement, their associated test scenarios and the last instance of a test run object associated to the test scenarios. This helps users viewing the status of the last execution of a test case.

The column also adds the run status information appended to the test run object. To retrieve the run status, the column is using the predefined run status attribute (see section 4.2.1.4 to configure which run status attribute to use) that is associated to the test run. Furthermore, if the run status uses one of the predefined values, the text displayed in the column is colored. The predefined values (and their associated color) are:

- Pass or Passed (colored in GREEN)
- Fail or Failed (colored in RED)
- Pending (colored in YELLOW)
- Any other statuses are colored in BLACK

If multiple test scenarios with test runs are associated to the requirement, all tests are colored in RED if at least one of the test runs failed, are colored in YELLOW if at least one test run is pending (and none of the test runs failed), are colored in GREEN if all test runs were passed. The column is colored in BLACK if none of the test scenarios have at least one associated test run.

The column displays:

"Test Scenario Code" – "Test Scenario Name"
"Last Test Run Code" – "Last Test Run Name" [Run Status]

Code	Name	Run Status (last)
MechReq_0020	Enviromental conditions	TS_0100 - Humidity altering TS_0100_Runs_002 - Humidity altering (Results) : [Fail]
MechReq_0030	Weight	TS_0110 - Weight Test TS_0110_Runs_001 - Weight Test (Results) : [Pass] TS_0120 - Bump Test TS_0120_Runs_001 - Bump Test (Results) : [Pending]
MechReq_0040	Air shipment	TS_0110 - Weight Test TS_0110_Runs_001 - Weight Test (Results) : [Pass]

If a test scenario doesn't have any test run object, it is displayed in BLACK:

Code	Name	Run Status (last)
SysReq_0010	Maximum weights and dimensions	SYS_TS_0010 - Modify preference settings

Note: Test run objects must be created as children (from a hierarchical relationships point of view) of the test scenarios for the column to display them (see section 4.2.2.3 on how to enable parent-children hierarchical relationships).

3.3.4 Run Status (full) Column

The "Run Status (full)" column (which name can be modified via the configuration file, see section 4.2.4.5) displays traceability between the requirement, their associated test scenarios, all test run objects associated to the test scenarios and the run status. This helps users viewing the full history of test execution based on the requirements.

The column displays:

"Test Scenario Code" – "Test Scenario Name"
"Test Run Code #1" – "Last Test Run Name" [Run Status]
"Test Run Code #2" – "Last Test Run Name" [Run Status]
"Test Run Code #3" – "Last Test Run Name" [Run Status]

Code	Name	Run Status (full)
MechReq_0020	Enviromental conditions	TS_0100 - Humidity altering TS_0100_Runs_001 - Humidity altering (Results) : [Fail] TS_0100_Runs_002 - Humidity altering (Results) : [Fail] TS_0100_Runs_003 - Humidity altering (Results) : [Pending]
MechReq_0030	Weight	TS_0110 - Weight Test TS_0110_Runs_001 - Weight Test (Results) : [Pass] TS_0110_Runs_002 - Weight Test (Results) : [Pending] TS_0120 - Bump Test TS_0120_Runs_001 - Bump Test (Results) : [Pass] TS_0120_Runs_002 - Bump Test (Results) : [Pass]
MechReq_0040	Air shipment	TS_0110 - Weight Test TS_0110_Runs_001 - Weight Test (Results) : [Pass] TS_0110_Runs_002 - Weight Test (Results) : [Fail]

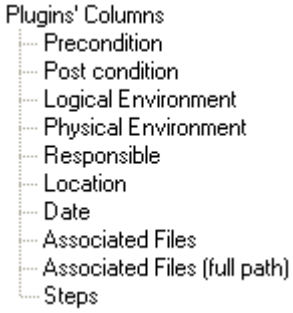
Note: Test run objects must be created as children (from a hierarchical relationships point of view) of the test scenarios for the column to display them (see section 4.2.2.3 on how to enable parent-children hierarchical relationships).

3.4 Test Scenarios View Columns

The test scenarios view columns modules add ten columns to the test scenarios view to help users see test scenarios properties. These columns are:

- Precondition : Display the precondition property
- Post condition : Display the post condition property
- Logical Environment : Display the logical environment property
- Physical Environment : Display the physical environment property
- Responsible : Display the responsible property
- Location : Display the location property
- Date : Display the date property
- Associated Files : Display the list of all associated files (only name of the file)
- Associated Files (full path) : Display the list of all associated files (with their full path)
- Steps : Display the list of all steps

The test status columns can be added via the Grid View Configuration of any test scenarios view, under the "Plugins' Columns" section:



All columns are in read-only mode. All columns are exportable to Excel.

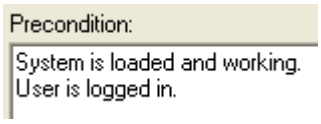
3.4.1 Test Scenario properties columns

The columns Precondition, Post condition, Logical Environment, Physical Environment, Responsible, Location and Date:

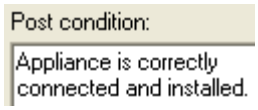
Code	Name	Precondition	Post condition	Logical Environment	Physical Environment	Responsible	Location	Date
TST_020	Correct appliance connection	System is loaded and working User is logged in.	Appliance is correctly connected and installed.	Configuration Y	Hardware X	John Smith	Madrid, Spain	2011-12-06

Directly correspond to the associated test scenario property:

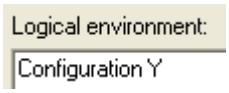
Precondition (from the "Precondition and Post condition" property tab)



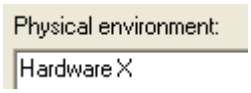
Post condition (from the "Precondition and Post condition" property tab)



Logical Environment (from the "Test Case" property tab)



Physical Environment (from the "Test Case" property tab)



Responsible (from the "Test Case" property tab)

Location (from the "Test Case" property tab)

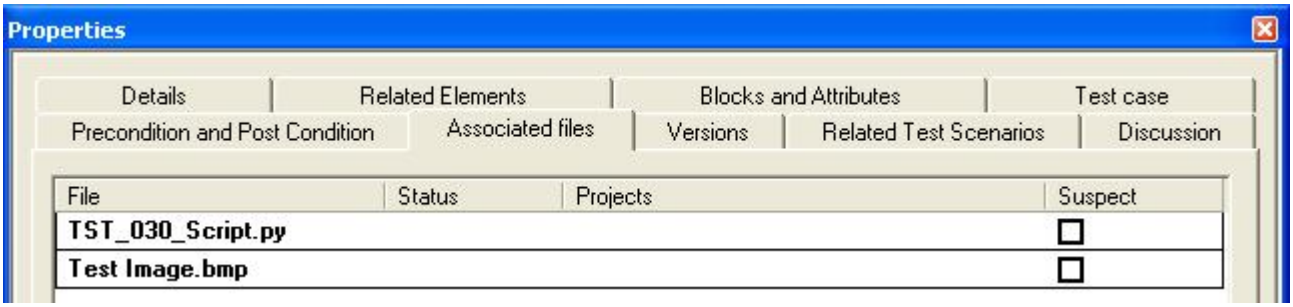
Date (from the "Test Case" property tab)

3.4.2 Associated files columns

The columns "Associated Files" and "Associated Files (full path)" displays the list of all files associated to a test scenario.

Code	Name	Associated Files	Associated Files (full path)
TST_020	Correct appliance connection	TST_020_Script.py	C:\Test Scripts\TST_020_Script.py
TST_030	Incorrect appliance connection	TST_030_Script.py Test Image.bmp	C:\Test Scripts\TST_030_Script.py C:\Test Scripts\Test Image.bmp

The associated files are set via the "Associated files" property tab:



3.4.3 Steps columns

The steps column displays the test scenarios steps in a textual format. The format used is:

- Step #X [Actor: Y] [Direction: <-- OR --> OR <-->]
  [Event: Z]
  Description...

For example, the steps:

Id	Actor	Event	Direction
1	**User**		---->
	The user presses the 'Connect appliance' option in the screen.		
2	**User**		<----
	The system provides the possibility to select the initial working mode of the appliance.		
3	**User**		---->
	The user selects the working mode for the appliance being connected.		
4	**System**		<--->
	The system checks that the appliance is ready to be connected.		
5	**User**		<----
	The system enables the Accept button.		
6	**User**		---->
	The user presses the Accept button.		

Are displayed as:

Code	Name	Steps
TST_020	Correct appliance connection	- Step #1 [Actor: User] [Direction: Destination System -->] The user presses the 'Connect appliance' option in the screen. - Step #2 [Actor: User] [Direction: Destination System -->] The system provides the possibility to select the initial working mode of the appliance. - Step #3 [Actor: User] [Direction: Source System <--] The user selects the working mode for the appliance being connected. - Step #4 [Actor: System] [Direction: Reflexive <-->] The system checks that the appliance is ready to be connected. - Step #5 [Actor: User] [Direction: Destination System -->] The system enables the Accept button. - Step #6 [Actor: User] [Direction: Source System <--] The user presses the Accept button.

4 Test Management Extension Configuration File

The Test Management Extension is configured via the TMEConfig.cfg.xml file. The configuration options are used by the individual TME module.

Below is an example of the configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config>
  <TME>
    <RunBlock>Runs</RunBlock>
    <TestStatusAttribute>Test Status</TestStatusAttribute>
    <RunStatusAttribute>Run Status</RunStatusAttribute>
  </TME>
  <TestSession>
    <Partition>default</Partition>
    <HierarchicalLink>1</HierarchicalLink>
    <Motive>Execute</Motive>
    <UseSameCodeFormat>1</UseSameCodeFormat>
    <CodeFormatSuffix>_Runs_</CodeFormatSuffix>
    <NameFormat>%s (Results)</NameFormat>
    <IncludeDescription>1</IncludeDescription>
  </TestSession>
  <TestSessionExecution>
    <PythonPath>c:\Python27\Python.exe</PythonPath>
    <IRQAArgument>IRQA:Arguments</IRQAArgument>
  </TestSessionExecution>
  <TestStatusColumns>
    <TestScenariosColumn>
      <Name>Test Scenarios</Name>
      <Enabled>1</Enabled>
    </TestScenariosColumn>
    <TestStatusColumn>
      <Name>Test Status</Name>
      <Enabled>1</Enabled>
    </TestStatusColumn>
    <RunStatusLastColumn>
      <Name>Run Status (last)</Name>
      <Enabled>1</Enabled>
    </RunStatusLastColumn>
    <RunStatusFullColumn>
      <Name>Run Status (full)</Name>
      <Enabled>1</Enabled>
    </RunStatusFullColumn>
  </TestStatusColumns>
</Config>
```

Note: XML is a case-sensitive language, make sure to use tags with their correct case.

4.1 XML Configuration file modifications

If the "TMEConfig.cfg.xml" configuration file is modified, IRQA needs to be restarted in order to reconfigure the Test Management Extension.

4.2 XML Configuration Tags

This section describes the XML configuration tags that are used to configure the Test Management Extension and their effect on the individual modules.

All configuration tags must be added within the <Config> tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<Config>
...
</Config>
```

4.2.1 Test Management Extension general settings

The Test Management Extension general settings are used across the individual modules.

```
<TME>
  <RunBlock>Runs</RunBlock>
  <TestStatusAttribute>Test Status</TestStatusAttribute>
  <RunStatusAttribute>Run Status</RunStatusAttribute>
</TME>
```

4.2.1.1 <TME>

This is the entry tag of the general settings. All general settings must be added below.

4.2.1.2 <RunBlock>

This tag defines the name of the predefined Run block used in the project.

Default value (if tag not used): **Runs**

4.2.1.2.1 Effect on modules

The Test Session Creation module uses this setting when creating the hierarchical link between the test session block and the run block. This allows users to define attributes that all test session blocks inherit (from the run block). The run block name is also used to filter the "Select Test Plan" drop-down menu, removing any runs or test session blocks (see section 3.1).

The Test Session Execution module uses this setting to filter the "Select Test Session" drop-down menu, removing any block that are not runs (see section 3.2).

The Test Status Columns module uses this setting to retrieve the run status of only the test run objects.

4.2.1.3 <TestStatusAttribute>

This tag defines the name of the predefined Test Status attribute used in the project.

Default value (if tag not used): **Test Status**

4.2.1.3.1 Effect on modules

The Test Status Columns module uses this setting to retrieve the test status of the test scenarios object that are directly link to the requirement. This value is used by the "Test Status" column (see section 3.3.2).

4.2.1.4 <RunStatusAttribute>

This tag defines the name of the predefined Run Status attribute (i.e. the test status of the runs) used in the project.

Default value (if tag not used): **Run Status**

4.2.1.4.1 Effect on modules

The Test Status Columns module uses this setting to retrieve the run status of the test run objects. This value is used by the "Run Status (last)" column (see section 3.3.3) and the "Run Status (full)" column (see section 3.3.4).

The Test Session Execution module uses this setting when importing results from the csv file (see section 3.2.3).

4.2.2 Test Session Creation settings

The Test Session Creation settings are used to configure the Test Session Creation module.

```
<TestSession>
  <Partition>default</Partition>
  <HierarchicalLink>1</HierarchicalLink>
  <Motive>Execute</Motive>
  <UseSameCodeFormat>1</UseSameCodeFormat>
  <CodeFormatSuffix>_Runs_</CodeFormatSuffix>
  <NameFormat>%s (Results)</NameFormat>
  <IncludeDescription>1</IncludeDescription>
</TestSession>
```

4.2.2.1 <TestSession>

This is the entry tag of the Test Session Creation settings. All settings must be added below.

4.2.2.2 <Partition>

This tag defines in which partition the test run objects are created.

Default value (if tag not used): **default**

4.2.2.2.1 Effect on module

The Test Session Creation module uses this setting as the partition in which the test run objects are created. The partition used here needs to be predefined in the project and it must be able to contain "Test Scenarios" object. Furthermore, the user needs to have write access to it. If these conditions are not met, test run objects will not be created.

4.2.2.3 <HierarchicalLink>

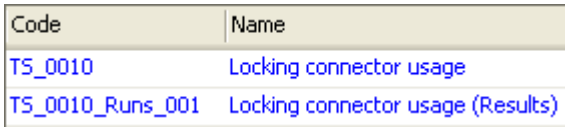
This tag defines whether the test run objects are created as children of the test plan's test cases.

Default value (if tag not used): **1** (i.e. created as children)

4.2.2.3.1 Effect on module

When the tag is set to "1", the test creation module creates the test run object as a child of the test case.

When the tag is set to "0", the test creation module does not create the test run object as a child of the test case.



Note: If the hierarchical links are not used (tag set to "0"), it prevents the "Run Status (last)" and "Run Status (full)" of the Test Status Columns module (see section 3.3.3 and 3.3.4) from working since they are using these hierarchical relationships.

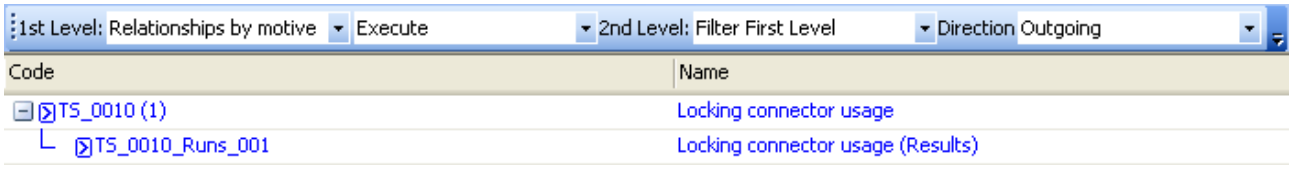
4.2.2.4 <Motive>

This tag defines which motive is used to create the traceability relationship between the test run objects and the test plan's test cases.

Default value (if tag not used): **Execute**

4.2.2.4.1 Effect on module

The Test Session Creation module uses this setting when creating the traceability relationship between the test run object and the test plan's test case.



If the motive doesn't exist, it is created automatically.

If the motive already exists, and traceability restrictions are enabled, users needs to make sure that it is possible to create a traceability relationship between the test run object and the test plan's test case. If it is not the case, then no traceability links are created.

4.2.2.5 <UseSameCodeFormat>

This tag defines whether the test run code is based on the test plan's test case code format or the default test scenario code format.

Default value (if tag not used): **0** (i.e. using the default test scenario code format)

4.2.2.5.1 Effect on module

The Test Session Creation module uses this setting when assigning a code to the test run object. If it is set to "1", the code of the test plan's test case is used, to which the string "_XXX" is appended (XXX are digits that are incremented for each test run).

For example:

Test Case: **TS_0010**
Test Run: **TS_0010_001**

If the tag is set to "0", the default test scenario code format is used (set via the "Configuration" -> "Code" -> "Test Scenarios" menu in IRQA).

For example:

Test Case: **TS_0010**
Test Run: **00010**

4.2.2.6 <CodeFormatSuffix>

This tag defines whether a suffix is appended to the test run code. The test run code format is defined via the <UseSameCodeFormat> tag (see 4.2.2.5).

Default value (if tag not used): **""** (i.e. empty value, the suffix value is ignored)

4.2.2.6.1 Effect on module

The Test Session Creation module uses this setting in collaboration with the <UseSameCodeFormat> tag.

If the <CodeFormatSuffix> is set to a value and <UseSameCodeFormat> is enabled, then the test run code format is: [Test Scenario Code] + [CodeFormatSuffix] + [XXX]

For-example:

<CodeFormatSuffix> is set to **"_Runs_"**
<UseSameCodeFormat> is set to **"1"**

The Test Case: **TS_0010**
generates this test run code: **TS_0010_Runs_001**

If the <CodeFormatSuffix> is set to a value and <UseSameCodeFormat> is disabled, then the test run code format is: [Next default Test Scenario Code] + [CodeFormatSuffix]

For-example:

<CodeFormatSuffix> is set to **"_Runs_"**
<UseSameCodeFormat> is set to **"0"**

The Test Case: **TS_0010**
generates this test run code (when default format is to use of 5 digits): **00010_Runs**

If the <CodeFormatSuffix> tag is empty, the suffix is ignored.

4.2.2.7 <NameFormat>

This tag defines the format of the test run object name.

Default value (if tag not used): **"%s (Runs)"**

4.2.2.7.1 Effect on module

The Test Session Creation module uses this setting when assigning a name to the test run object. To reuse the same name as the test plan's test case object, the symbol "%s" can be used.

For example:

<Nameformat> is set to **"%s (Results)"**

Test Case Name: **Locking connector usage**
Test Run Name: **Locking connector usage (Results)**

If the tag is empty or not used, the default value **"%s (Runs)"** is used.

4.2.2.8 <IncludeDescription>

This tag indicates the module if the test plan's test case description is copied inside the test run description.

Default value (if tag not used): **"1"**

4.2.2.8.1 Effect on module

The Test Session Creation module uses this setting to determine if the test plan's test case description is copied in test run description.

If the tag is set to "1", the description is copied.
If the tag is empty or set to "0", the description is NOT copied.

4.2.3 Test Session Execution settings

The Test Session Execution settings are used to configure the Test Session Execution module.

```
<TestSessionExecution>
  <PythonPath>c:\Python27\Python.exe</PythonPath>
  <IRQAArgument>IRQA:Arguments</IRQAArgument>
</TestSessionExecution>
```

4.2.3.1 <TestSessionExecution>

This is the entry tag of the Test Session Execution settings. All settings must be added below.

4.2.3.2 <PythonPath>

This tag defines the path AND the python executable that is used to execute the scripts.

Default value (if tag not used): **"c:\Python27\Python.exe"**

4.2.3.2.1 Effect on module

The Test Session Execution module uses this setting when calling the python scripts.

See section 3.2.4 on how python scripts are called.

If the tag is empty, the default value is used.

Note: The module doesn't check if the path is valid and if the executable exists. Users need to make sure they are valid.

4.2.3.3 <IRQAArgument>

This tag defines the IRQA attribute that can be used as a script argument. The IRQA attribute needs to be a TEXT attribute and defined by the user. Attribute are not automatically created by the module and ignored if not found.

Default value (if tag not used): **"IRQA:Arguments"**

4.2.3.3.1 Effect on module

The Test Session Execution module uses this setting when calling the python scripts with arguments.

The attribute name specified in the <IRQAArgument> tag is used as a test run argument (Test Scenario attribute) AND the test session argument (block property).

See section 3.2.2 on how arguments are used.

If the tag is empty, the default value is used.

4.2.4 Test Status Columns settings

The Test Status Columns settings are used to configure the Test Status Columns module.

```
<TestStatusColumns>
  <TestScenariosColumn>
    <Name>Test Scenarios</Name>
    <Enabled>1</Enabled>
  </TestScenariosColumn>
  <TestStatusColumn>
    <Name>Test Status</Name>
    <Enabled>1</Enabled>
  </TestStatusColumn>
  <RunStatusLastColumn>
    <Name>Run Status (last)</Name>
    <Enabled>1</Enabled>
  </RunStatusLastColumn>
  <RunStatusFullColumn>
    <Name>Run Status (full)</Name>
    <Enabled>1</Enabled>
  </RunStatusFullColumn>
</TestStatusColumns>
```

4.2.4.1 <TestStatusColumns>

This is the entry tag of the Test Status Columns settings. All settings must be added below.

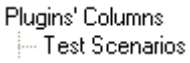
4.2.4.2 <TestScenariosColumn>

This section contains the tags used to set the name of the Test Scenario Column (see section 3.3.1) and whether it is enabled or not.

4.2.4.2.1 <Name> and <Enabled>

The tag <Name> is used to set up the name of the Test Scenario Colum. This name is used as the column name in the Requirements View. If the tag is empty, the default name is used.

Default value (if tag not used): **"Test Scenarios"**



The tag <Enabled> tells the module to enable the column or not. If the column is disabled (value of "0"), the column is not listed under the Plugins' Columns in the Requirement View. If the tag is enabled (value of "1" or if value is empty), the column is listed.

Default value (if tag not used): **"1"**

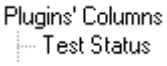
4.2.4.3 <TestStatusColumn>

This section contains the tags used to set the name of the Test Status Column (see section 3.3.2) and whether it is enabled or not.

4.2.4.3.1 <Name> and <Enabled>

The tag <Name> is used to set up the name of the Test Status Colum. This name is used as the column name in the Requirements View. If the tag is empty, the default name is used.

Default value (if tag not used): **"Test Status"**



The tag <Enabled> tells the module to enable the column or not. If the column is disabled (value of "0"), the column is not listed under the Plugins' Columns in the Requirement View. If the tag is enabled (value of "1" or if value is empty), the column is listed.

Default value (if tag not used): **"1"**

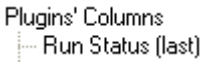
4.2.4.4 <RunStatusLastColumn>

This section contains the tags used to set the name of the Run Status (last) Column (see section 3.3.3) and whether it is enabled or not.

4.2.4.4.1 <Name> and <Enabled>

The tag <Name> is used to set up the name of the Run Status (last) Colum. This name is used as the column name in the Requirements View. If the tag is empty, the default name is used.

Default value (if tag not used): **"Run Status (last)"**



The tag <Enabled> tells the module to enable the column or not. If the column is disabled (value of "0"), the column is not listed under the Plugins' Columns in the Requirement View. If the tag is enabled (value of "1" or if value is empty), the column is listed.

Default value (if tag not used): **"1"**

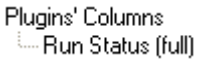
4.2.4.5 <RunStatusFullColumn>

This section contains the tags used to set the name of the Run Status (full) Column (see section 3.3.4) and whether it is enabled or not.

4.2.4.5.1 <Name> and <Enabled>

The tag <Name> is used to set up the name of the Run Status (full) Colum. This name is used as the column name in the Requirements View. If the tag is empty, the default name is used.

Default value (if tag not used): **"Run Status (full)"**



The tag <Enabled> tells the module to enable the column or not. If the column is disabled (value of "0"), the column is not be listed under the Plugins' Columns in the Requirement View. If the tag is enabled (value of "1" or if value is empty), the column is listed.

Default value (if tag not used): **"1"**

the requirements company

Avda. de los Labradores, 1, 4th Floor
28760 Tres Cantos, Madrid (Spain)
Tel: +34 91 806 17 13 Fax: +34 91 804 39 50
info@visuresolutions.com
www.visuresolutions.com