



14/12/2016

How to for tests and campaigns

Not for General Release

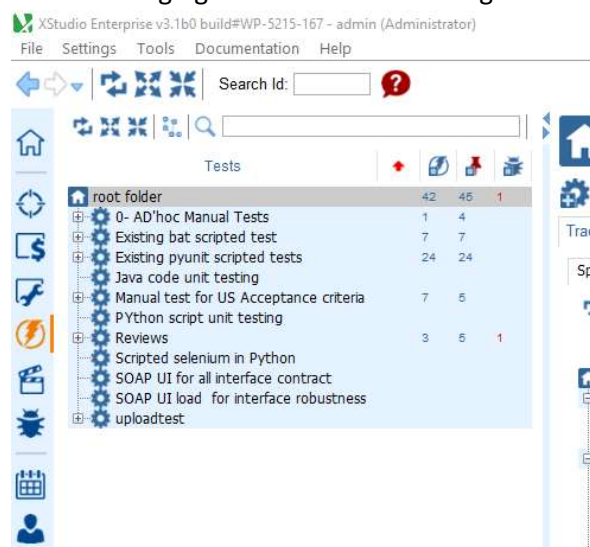
Create test

- Tests are made of test cases (at least one per test)
- Test cases may have procedures (especially for manual tests)
- Procedures are sequential steps with ‘Do’ and ‘Checks’ declaratives
- Tests may have attributes
- Attributes can be valued differently for each test
- Test cases may have Params
- Params may be valued differently for each test case
- You can also personalize test by using costumed fields (declared by right clicking on the root folder of the test tree)
- A test case must be set as “Ready for manual run” and/or “Ready for automated run” in order to be used
- If this is not set, the test can be used in a campaign but can’t be run

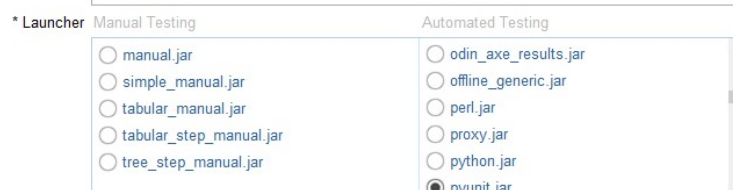
1) Define the category of test you need to create

a. Each category uses its own launcher

i. The following figure shows several categories.



ii. The “PYthon script unit testing” uses the launcher “pyunit”



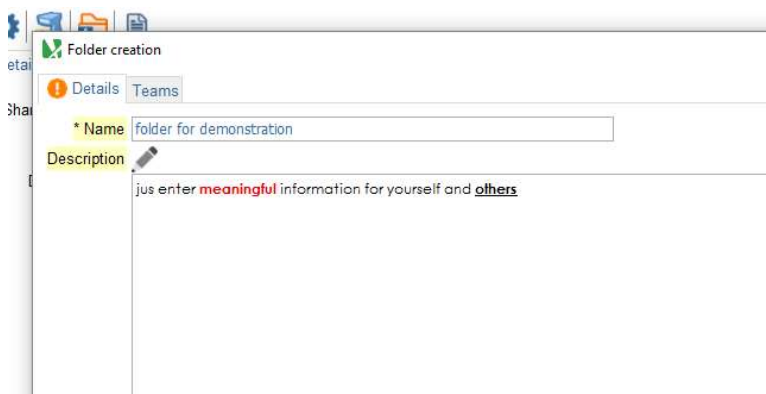
b. You can choose between 5 manual test launchers and 10s of automated frameworks launchers

i. Automation launchers will launch the test scripts on your local machine or a remote PC/server

- ii. Manual launchers will launch a GUI for each test
 - 1. Each manual launcher present a different GUI
 - 2. You need to pick the one that best meets your needs
- c. Any test you create in a test category will be using the launcher attached to that category

2) Under each category, you create folders to gather related tests together

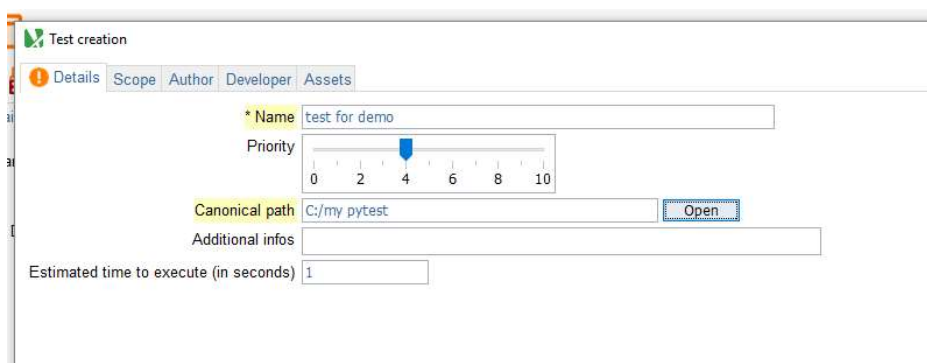
- a. Right click on the category and ‘Create folder’



- b. You can also assign “teams” that will have access to that folder ... if you have started by defining some teams
 - i. Otherwise you will be able to set access rights later

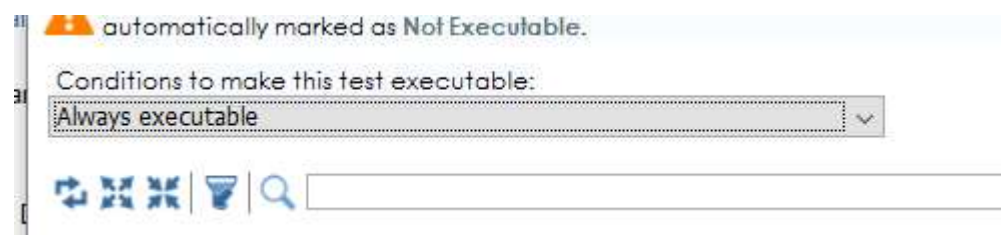


- c. Then right click on the folder and create a test

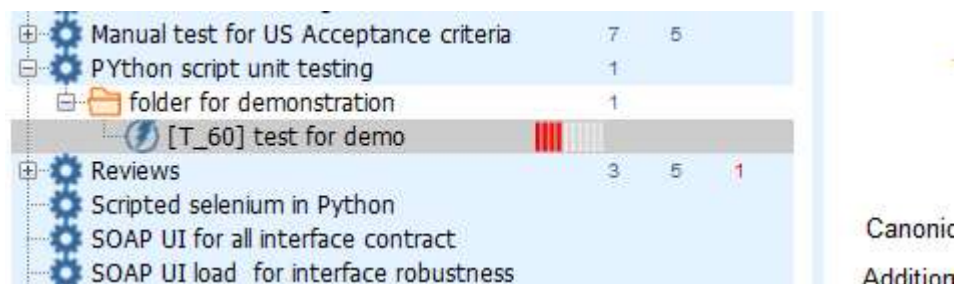


- d. Provide the test name
 - i. It must match the name of your script for automated tests
 - ii. For manual tests, name it with something sufficiently descriptive
- e. Indicate the path to the test
 - i. The path is relative to a root path that you will specify at execution time

- ii. The root path is based on the target machine (the agent in XStudio parlance) where the script resides
 - iii. The root path will be defined in the configuration (defining how to execute the test)
- f. Describe your test with a scope (prerequisites and description)
 - i. This is extremely useful for manual testing and for reports
- g. The author is automatically set as “you”
- h. The developer (of the test) can be set
- i. The assets (resources) required to execute the test can then be set
 - i. You can set the policy governing how the test can be executed whether an asset is present or not



3) The test is created once you ‘submit’



- a. Each test is identified with an id (here: T_60)
- 4) Once created, you can also define additional information
- a. **Coverage** allows to define which requirements or specifications this test covers
 - b. **Dependencies** will allow to define on which other tests this one is dependent on
 - c. **Attributes** will allow to assign attributes and their values to this test
 - i. The attributes will then be provided to the launcher at run time
 - d. **Assets** allows to review and modify the assets this test relies on
 - e. **Attachments** allows to attach files to the test (e.g. any document, script that allows to better understand the test ...)
 - i. Note that attachments are by default stored in the DB, but you can configure XStudio so that it is stored on a share (SAMBA share for example)
 - f. **Campaigns, Results and Bugs** are display-only tab information as they allow to know in which campaigns and sessions this test is involved, what are the test results for that specific test so far, and which defect are linked to the test
 - i. At this stage, there are empty for us as this is a new test
- 5) At that stage, we have created a new test (in this case as test that links to a pyUnit test)
- 6) Alternatively, if you have existing tests, you can also “scan” them and import them into XStudio

- a. So, in such case, you don't have to create them one by one
- b. The scan is done by right clicking on the folder, providing a "configuration" with the root path; Then, XStudio recursively parses the folder structure you defined and finds the test scripts that are already existing

Scan physical tests

This feature will scan your local disk to find physical tests/scripts that you may not have already referenced in XStudio.

The scan will be performed recursively starting from the **Test root path** specified in the configuration selected below. When the scan is completed, it will suggest a list of tests.

You'll then be able to select manually in this suggestion list which ones you want to reference in XStudio.

Be careful picking the right configuration below!

The configuration you select potentially contains information that will be used by the scanner (i.e. the **test root path** to know where to start scanning from, the **file extension** of the scripts to search, etc.). So **it must be a valid configuration!**. In other words, you must be able to run existing tests locally with that configuration.

Configuration

Existing bat scripted test

Configuration scanning bat tests



Id 2

General

* Test root path C:\Users\pacca\Desktop\scripted test for demo

Synchronous executable ☒

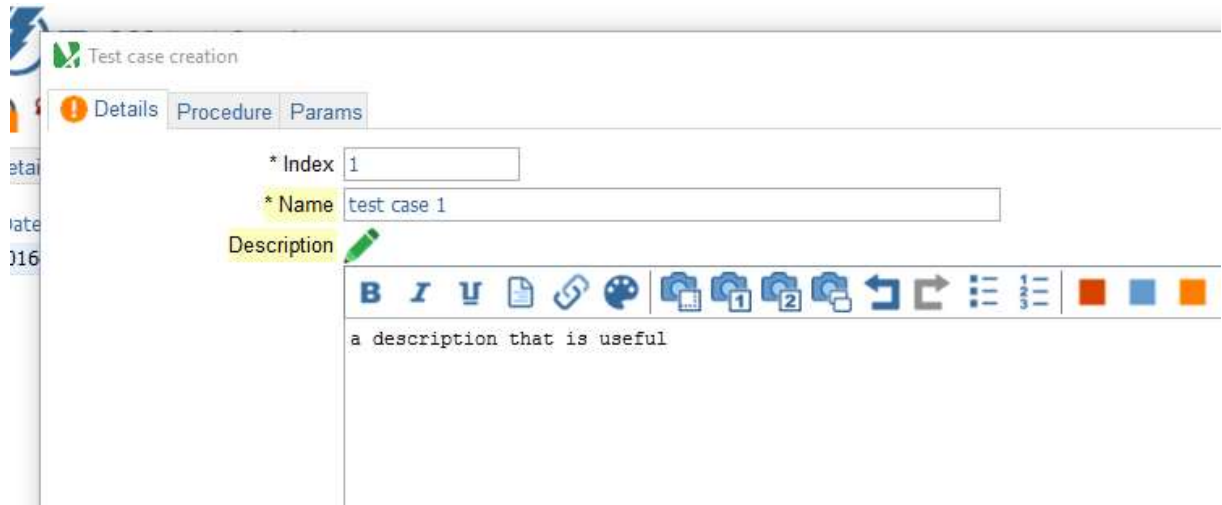
* Asynchronous timeout (in seconds) 600

Scan

CANCEL

Creating test cases

- 1) Each test has at least one test case
- 2) To create a test case, right click on the test and "Create test case"
- 3) Enter a useful name and description
- 4) Note the index that allows to order the test cases in a test



5) Then, state if your test case can be run manually and/or automatically



- a. A test case must be at least “ready for manual run” if you consider it ready
- b. Otherwise this means that you still should describe its procedure
- c. If set as “Ready to automate run”, it must have the script ready and tested

6) For manual test, you must describe a procedure

- a. A procedure has steps
- b. To create a step, click on the green buttons (insert above, insert below)

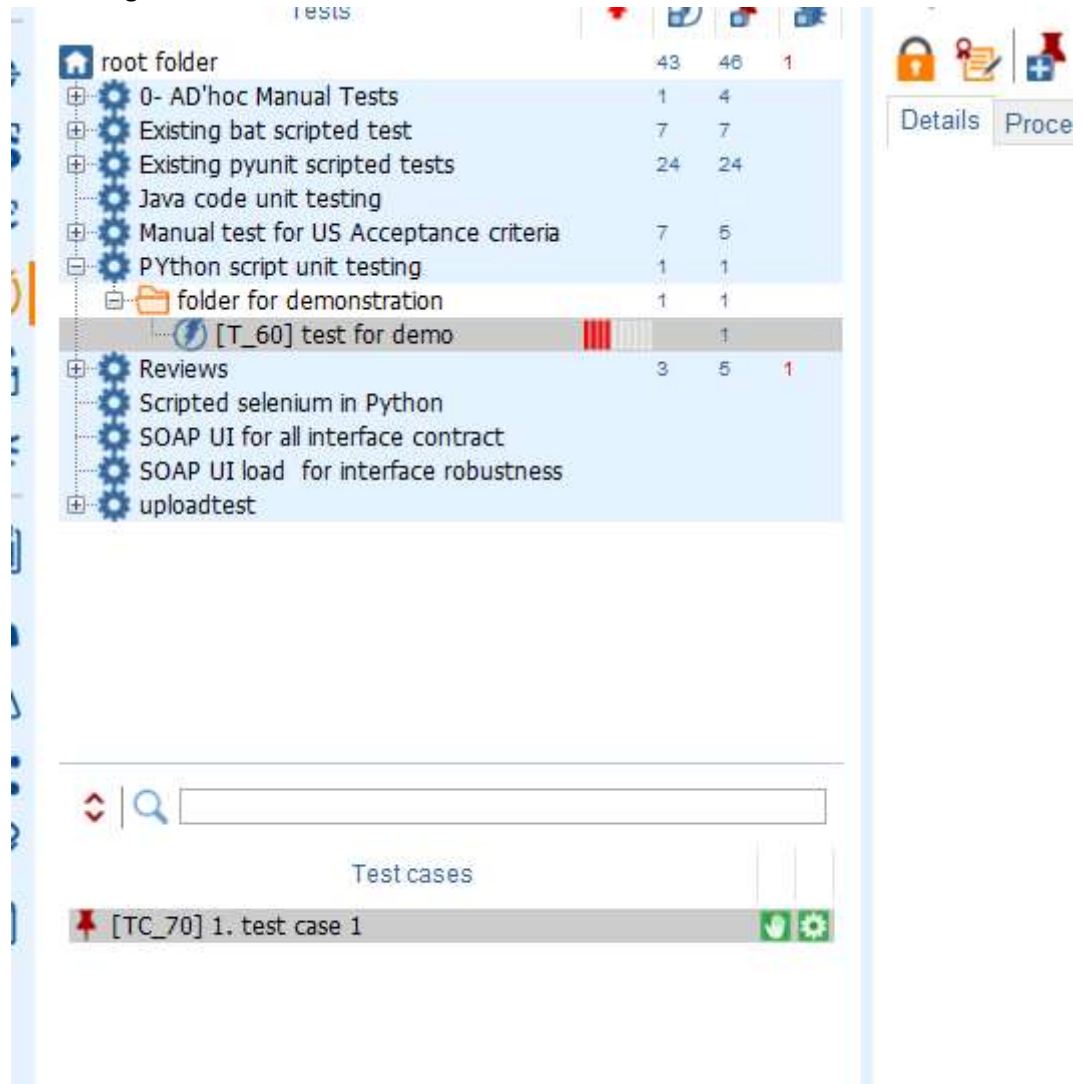


c. This inserts a step line and you describe the ‘Steps’ and ‘Checks’

Steps		Checks	
do this		check this	
do that			

- d. Note that you can also assign params to each test case
 - i. Params are parameters with values
 - ii. Each test case can use different valued params
 1. Studio also allows to generate test cases based on params by doing (pairwise or full) combination

e. You then get a test case for this test



The screenshot shows the XQUAL interface with a list of tests and a search bar. The tests are listed in a table with columns for test name, status, and count. The search bar is located below the table.

Test Name	Status	Count
root folder		43 46 1
0- AD'hoc Manual Tests		1 4
Existing bat scripted test		7 7
Existing pyunit scripted tests		24 24
Java code unit testing		
Manual test for US Acceptance criteria		7 5
PYthon script unit testing		1 1
folder for demonstration		1 1
[T_60] test for demo		1
Reviews		3 5 1
Scripted selenium in Python		
SOAP UI for all interface contract		
SOAP UI load for interface robustness		
uploadtest		

Search bar: [Search]

Test cases

[TC_70] 1. test case 1

f. You can then carry on adding other test cases as per your needs .

Test plans

- Test plans are built using “campaigns”
- Campaigns collect tests
- Campaigns are run through sessions
- Campaigns can collect tests of any category
 - Hence, you can mix manual and automated tests, as well as automated tests using different launchers (e.g.; Pyunit, Robot Framework ...)

You can then plan against your test strategy

- a. Unit test campaign
- b. Functional test campaign
- c. Performance testing
- d. Etc.

or

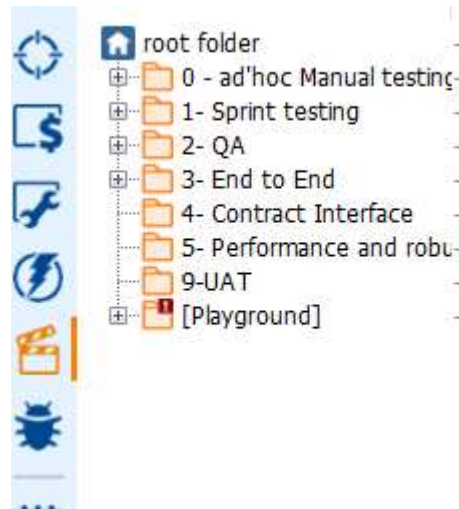
- e. Per-feature campaign including unit testing, integration, functional, quality attribute tests
- f. Per-sprint campaign s
- g. Etc.

Also, if your strategy involves having large campaigns, you may at some point calculate a residual campaign that collects only a subset of the tests (e.g. those that were never tried, or those that failed at least once ...)

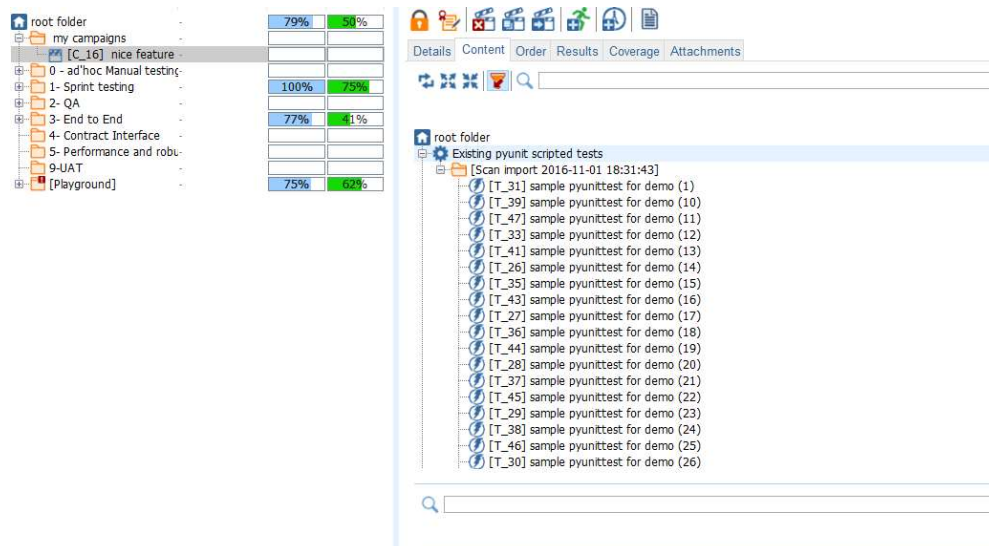
To the contrary, if you create some small campaigns (e.g. per-feature campaigns), you may decide to create a large regression campaign by combining several sessions together. Then you run this campaign covering all aspects of all features that you deem “ready for release”

Create a campaign

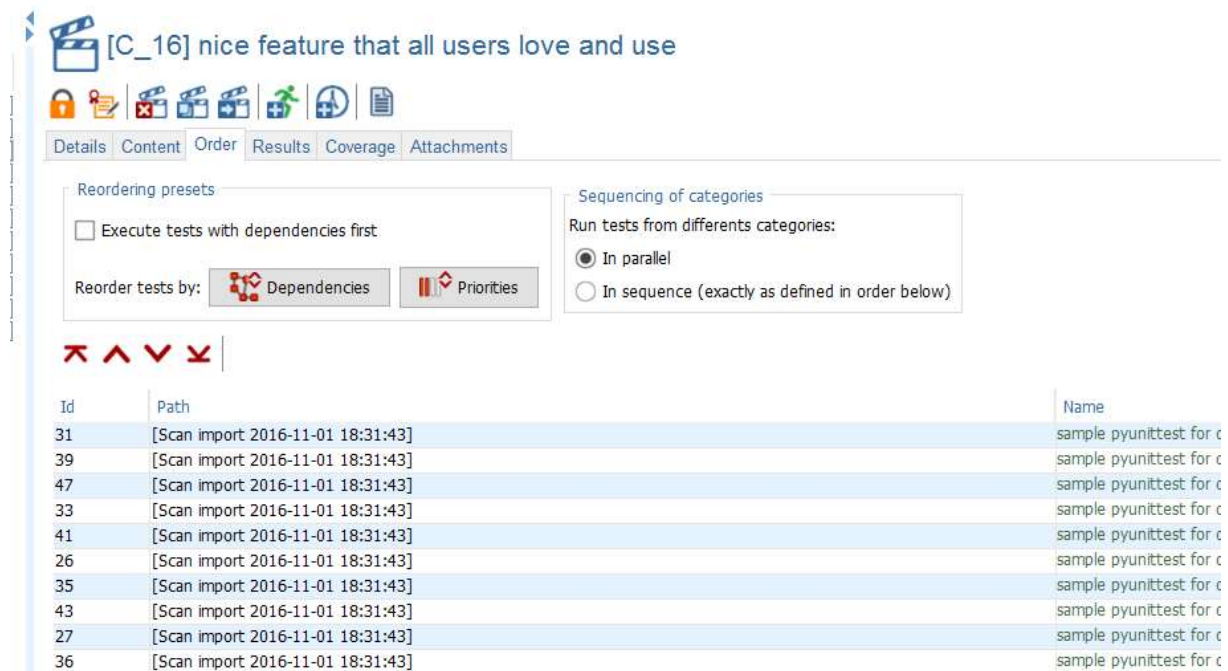
- a. Create a folder under the campaign tree



- b. As always, set a descriptive name, a useful description and, if you have setup teams, you can already associate this folder to the teams that will need to access it
- c. Create a campaign by right-clicking on the folder
- d. Set a useful name (e.g. “nice feature that all users love and use” and a description
- e. Then, identify which tests will be part of your campaign using the “content” tab
 - a. In the tree, choose the tests or the test folder(s) you need



- b. Alternatively, you can select based on sophisticated filters
http://www.xqual.com/documentation/user_guide/filtering.html
- f. If you click on one of the tests, you can see the test cases of that test, hence checking this is what you wanted
- g. If you need to order the test execution for that campaign, you do so by setting it in the ‘Order’ tab



The screenshot shows the XStudio interface with a title bar [C_16] nice feature that all users love and use. Below the title bar is a toolbar with icons for file operations, test execution, and settings. A tabbed interface shows 'Details', 'Content', 'Order', 'Results', 'Coverage', and 'Attachments'. The 'Order' tab is active, showing two panels: 'Reordering presets' and 'Sequencing of categories'. The 'Reordering presets' panel has a checkbox 'Execute tests with dependencies first' and a 'Reorder tests by:' section with 'Dependencies' and 'Priorities' buttons. The 'Sequencing of categories' panel has a section 'Run tests from different categories:' with radio buttons for 'In parallel' (selected) and 'In sequence (exactly as defined in order below)'. Below these panels is a table of tests with columns 'Id', 'Path', and 'Name'.

Id	Path	Name
31	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
39	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
47	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
33	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
41	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
26	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
35	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
43	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
27	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c
36	[Scan import 2016-11-01 18:31:43]	sample pyunittest for c

- Note that you can set the order by the priority you have set when describing the test or based on their dependencies
- In case you have test in different categories (using different launchers), you can also decide if you want to run them in parallel or in sequence (e.g. wait for manual tests to complete, before you run the automated tests)
- Results** and **Coverage** are display-only tabs that results from the configuration you have done on each test
- Attachments** tab allows you to attach a file for that session, if needed

Test sessions

Once you have a campaign, you need to run it; in XStudio parlance this is a session

- Right click the campaign and “Create a campaign session”
- The name is proposed – the campaign name + a time-stamp – you can change it if needed
- You then define which test operator (human) will run the manual tests for that session
- Which agent (machine) will run the tests
 - An agent allows to run a test on any remote PC
 - For evaluation purpose, you can use your current XStudio as the agent
 - Any PC running XStudio is an agent
 - You also can set unattended agents on a remote PC (this is XAgent)
 - This will pick up the tests it is assigned and run then without needing an operator to act
 - All agents are automatically added to XStudio, if they are started at least once
 - By default, any agent can run the tests

- g. But you can define a pool of agent

Execute on

☐ Any agent (can be started manually from any XStudio)

☒ All the agents in the pool

☐ The first available agent in the pool

- e. You also must define against which SUT you run the campaign
- a. You must run a campaign against a single SUT (application/product and specific version of it)
- f. Then you choose the configuration for each test category you included into the campaign

Details Test operator Agents SUT Configuration Overloading Assets Monitoring

If a parent test has status failed

☐ Execute child tests

☐ Set child tests as failed

☒ Set child tests as not executed

If a parent test has status unknown

☒ Execute child tests

☐ Set child tests as failed

☐ Set child tests as not executed

Existing pyunit scripted tests PYTHON script unit testing

Configuration scan script pyunittest

Id 4

General PyUnit

* Test root path C:\Users\pacca\Desktop\scripted test for demo

- a. A configuration specifies some run time parameters (e.g. root path for the automated tests, which version of pyUnit, time between tests and so on)
- b. You can have several configurations and switch them between sessions
- c. You can decide how to orchestrate your remaining tests in case a test fails

Details Test operator Agents SUT Configuration Overloading Assets Monitoring CC Er

If a parent test has status failed

☐ Execute child tests

☐ Set child tests as failed

☒ Set child tests as not executed

If a parent test has status unknown

☒ Execute child tests

☐ Set child tests as failed

☐ Set child tests as not executed

Existing pyunit scripted tests PYTHON script unit testing

- g. At that stage, you can run your session

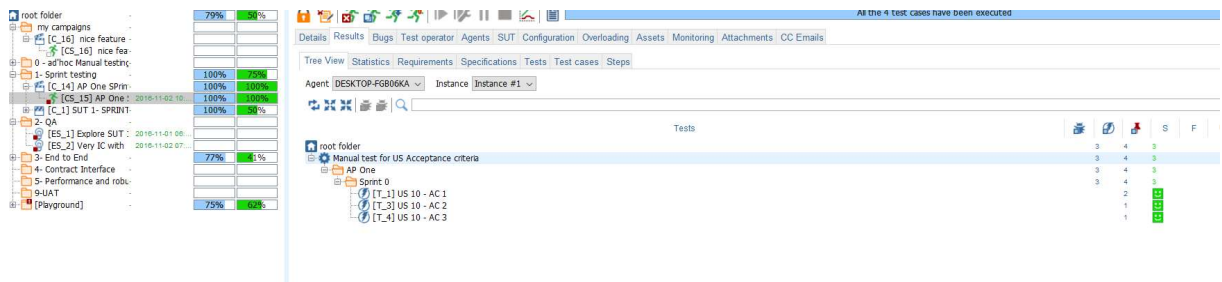
- a. By default, your settings are applied and the session is running on the agent, with the defined configuration, and by the operator you defined
- b. Note that you can still force the execution with a different launcher

Test Session results

Once a session has returned, you can review its results at session level, campaign level, by test, by covered specification, by covered requirement and by covered SUT

In all cases, this is achieved by looking into the “Results” tab of the object you are interest in

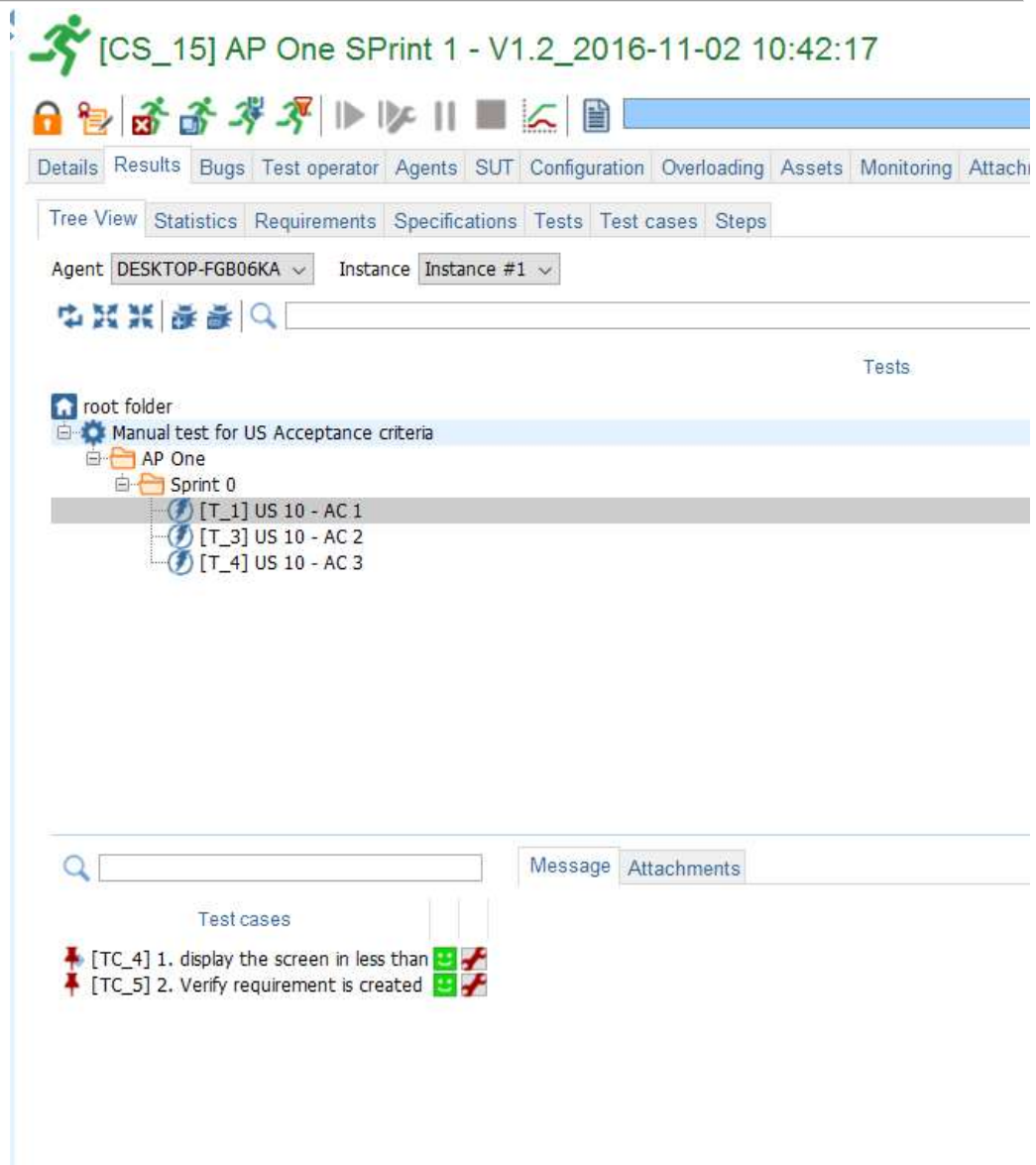
- For a test session, click on the session
- Click on the “results” tab



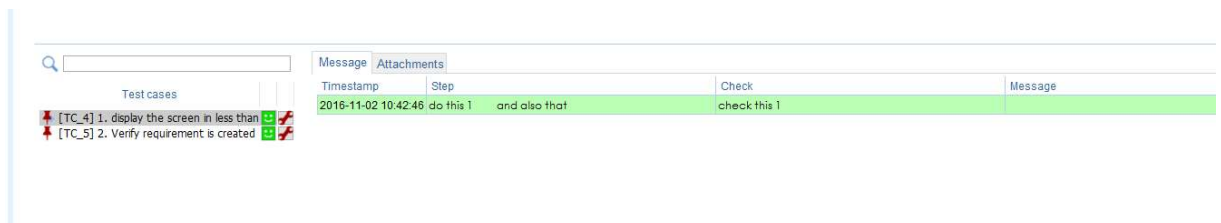
The screenshot displays the XQUAL interface. On the left, a tree view shows various test sessions with progress bars indicating completion status (e.g., 79%, 100%, 77%, 75%). The main area on the right shows the 'Results' tab for a selected session, displaying a table of test cases with status indicators (S, F, P).

Test Case	Status	Pass	Fail	Partial
Manual test for US Acceptance criteria	S	3	4	3
AP One	S	3	4	3
Sprint 0	S	3	4	3
TT_11 US 10 - AC 1	P	2	1	1
TT_31 US 10 - AC 2	P	1	1	1
TT_41 US 10 - AC 3	P	1	1	1

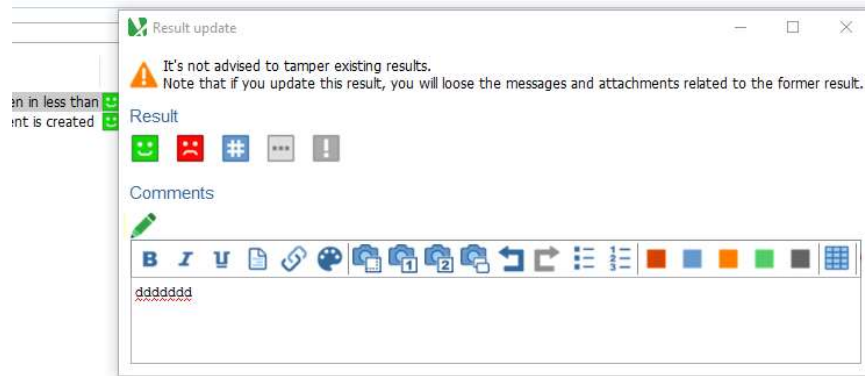
- The ‘Tree view’ tab tells you which tests where tried, passed, failed ..
- If you click on a test, you will see the details of it test cases



- e. If you click on the test case, you see the information that was fed back by the launcher (either by the operator for a manual test , or by the test script, through the launcher)



- f. If you click on the red tool, you can even edit it (well that's not a good practice but some customers require it)



- g. Then you can review results by requirements, specifications, tests, test cases and steps

Other interesting features

- **Exploratory sessions:** allows to run tests without having them pre-defined – an operator will do tests and take notes, that can then be later automatically translated as scripted tests
- **Running a test on its own without building a campaign:** this is useful when creating new tests and needing to ensure they are correct and working – in such case, you run the test and XStudio creates a campaign and session in the “playground” folder (under the campaign tree)
- **Residual campaigns:** once you have run several sessions for a campaign, you may get only a few tests that are still failing. You can then generate a new campaign that will only contain the subset of tests you need (there are 7 combinations)
- **Merged sessions into a campaign:** if you have several campaigns that are mostly passed, you can merge them into one campaign that will allow to do a full regression testing