Read the ExtensiveAutomation Documentation

Release 1.0

ExtensiveAutomation

Aug 10, 2019

About

CHAPTER 1

Extensive Automation

Welcome on the Extensive Automation project.

1.1 Concepts

ExtensiveAutomation is a test and deployment automation framework for integration, regression and end-to-end usages. The framework provided a rich and collaborative workspace environment. The installation is also simplified and some others great tools are integrated on it (selenium, sikuli)

The project have several purposes since the creation:

- Unify all testing tools in one environnement
- Provide a complete testing dashboard
- Make testing easy
- Testing everything from anywhere.
- Provide a user friendly environment.
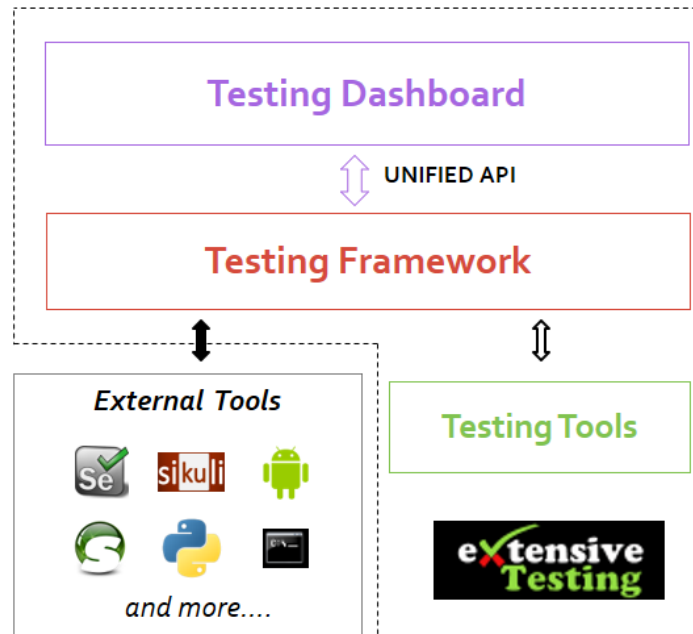
1.2 Usages

This solution is designed to make a lot of thing like:

- test automation in integration environment
- regression test automation
- end to end test automation
- deployment automation of virtual or physical machines
- deployment automation of software application

Note: The solution is written in Python and also all tests.

1.3 License

This environment is open source, and is freely available under the LGPL 2.1 version. All source code is available in github (https://github.com/ExtensiveAutomation).

1.4 Author

Started since 2010, the solution was developed by Denis Machard. This project is an effort, driven in my spare time.

If you want to sponsor me then I am accepting donations via PayPal. The money received will be used to cover web site costs, domain name reservation and more :wink:.

1.5 Contact

If you have any questions, you can contact me with:

- email d.machard@gmail.com
- twitter https://twitter.com/Extensive_Auto
- from the google forum https://groups.google.com/forum/#!forum/extensive-automation-users
- github https://github.com/ExtensiveAutomation

CHAPTER 2

Download

A complete release is generated every two months or less and each component can also evolve at its own rate. The solution is made of several components.

The solution can be downloaded from the website https://www.extensiveautomation.org

The server is available throught a `tar.gz` file, the following package are also integrated:

- Test framework
- Scheduler
- Adapters and libraries

The graphical client is available in 2 modes:

- portable version
- installation in program files

The toolbox is available in 3 modes:

- portable version
- installation in program files
- command line

Note: Client and toolbox can be run on Windows and Linux, in 64bits only.

Warning: The server must be executed in Linux (CentOS7 or RedHat7).

Warning: Administrators rights can be necessary to execute properly the client or the toolbox.

Changes Logs

3.1 Current version

Note: **Version 19.0.1 available since 08/09/2018**

- Bug fix on deployment server, pip command integration

Version 19.0.0 available since 08/05/2018

- New name ExtensiveAutomation for the solution
- All tests files are stored in XML by default (no more zlib compression)
- Bugs fixes and improvement in REST API
- Initial docker support
- Python 2.6 no more supported on server side
- Cache preview from the client during test writing
- Simplification of tests parameters with "text" and "json"
- SQL queries optimization on server side
- Begin to support python 3.5 on server side
- The client is no more embedded in the server side by default
- New feature to set a security banner on login page of the web interface and app client
- Update of selenium in 3.13.0
- New major version for the app client
- New major version for the toolbox

A detailed release notes is available in the installation package.

3.2 Previous versions

Version 18.0.0 released on 02/11/2018

- API XmlRPC is removed from the server
- Big improvment of the API REST
- New major client based on the API Rest
- Full support of Qt5.9 for the toolbox and graphical client
- Full support of python 3.6 for the toolbox and graphical client
- Code cleanup
- Several bugs fixed
- Update of selenium in 3.9.0
- The toolbox is no more embedded in the server side by default

Version 17.1.0 released on 10/22/2017

- Improvment in the REST api
- New features in the test framework library
- Several bugs fixed
- Improvment of the graphical interface of the client
- Experimental Ubuntu Support for the graphical client

Version 17.0.0 released on 06/04/2017

- 64bits support by default for the client and toolbox
- New major features in the test framework library
- New swagger for the REST api
- Update of selenium 3 et 2 in the toolbox
- Several bugs fixed

Version 16.1.0 released on 03/30/2017

- Several bugs fixed
- Improvment of the graphical interface of the client
- Installation improvment

Version 16.0.0 released on 25/02/2017

- Several bugs fixed
- Improvment of the REST api
- Changes on core server
- New features in the test framework library
- Optimization in server side to reduce the number of SQL requests
- Improvment of the graphical interface of the client
- 64bits Support for the graphical client and toolbox

Version 15.0.3 released on 04/11/2016

- Several bugs fixed
- New plugins for the graphical client
- Improvment of the REST API
- New features in the test framework library
- New interop module in test library

Version 14.0.0 released on 27/08/2016

- Several bugs fixed
- New features in the test framework library
- New major features on the REST api
- No more new feature in the XmlRPC api, just bug fix
- New features in the web interface
- Python2.7 no more supported on windows for the graphical client and toolbox
- Integration of the REST api in the graphical client
- Improvment of the graphical interface of the client
- New HP QC ALM plugin for the graphical client

Version 13.0.0 released on 23/06/2016

- Several bugs fixed
- New REST api on the server side
- New features in the test framework library
- Improvment in the core server
- Plugins support for the client and toolbox
- Improvment of the graphical interface of the client

Version 12.1.0 released on 29/04/2016

- Several bugs fixed
- New features in the test framework library
- Minors update on the XmlRPC API
- Improvment of the graphical interface of the client

Version 12.0.0 released on 12/02/2016

- Several bugs fixed
- New features on the XmlRPC API
- New features in the test framework library
- New features in the web interface

Version 11.2.0 released on 22/11/2015

- Several bugs fixed
- New features in the test framework library

- Improvment of the scheduler
- New public repository for the test framework library
- Offline installation support
- Minor changes on the XmlRPC api

Version 11.1.0 released on 18/10/2015

- Several bugs fixed
- New features on the XmlRPC API
- New features on the web interface

Version 11.0.0 released on 14/09/2015

- Several bugs fixed
- New features in the web interface
- Merge of agents and probes in the toolbox
- Update in the XmlRPC API
- Python 3.4 support for the graphical client and toolbox

Version 10.1.0 released on 12/07/2015

- Several bugs fixed
- CentOS 4 et 5 no more supported
- New features in the test framework library
- New features in the web interface

Version 10.0.0 released on 28/05/2015

- Several bugs fixed
- New features in the web interface
- Minor changes in the core server
- Update of the documentations
- Improvment of the graphical interface of the client

Version 9.1.0 released on 22/03/2015

- Several bugs fixed
- New features in the test framework library
- Product installation improved
- Improvment of the graphical interface of the client

Version 9.0.0 released on 05/01/2015

- Several bugs fixed
- New features in the test framework library
- Python 2.4 no more supported
- New features in the web interface
- Improvment of the graphical interface of the client

Version 8.0.0 released on 25/10/2014

- Several bugs fixed
- Improvment of the graphical interface of the client
- New features in the test framework library
- Minors changes in the XmlRPC API
- New features in the web interface

Version 7.1.0 released on 20/09/2014

- Several bugs fixed
- Documentations updated
- Optimization in server side to prepare a test
- New features in the core
- New features in the test framework library
- Improvment of the graphical interface of the client

Version 7.0.0 released on 08/08/2014

- Several bugs fixed
- Improvment in the scheduler
- Reverse proxy added on the front of the server
- Websockets support, activated by default
- New documentations
- tcp/443 used by default on all components
- SSL proxy support
- SSL used by default for agents and probes
- Improvment of the graphical interface of the client

Version 6.2.0 released on 02/06/2014

- Several bugs fixed
- Agents update
- Minors changes in the XmlRPC API
- New features in the test framework library
- Improvment of the scheduler

Version 6.1.0 released on 25/04/2014

- Several bugs fixed
- New features in the web interface
- New features in the test framework library
- Agents improvments

Version 6.0.0 released on 23/03/2014

- Several bugs fixed

- New packages for adapters and libraries
- New features in the XmlRPC API
- New features in the test framework library
- No more link with the twisted library
- SSL support on XmlRPC api
- Proxy socks4 support
- Agents Support

Version 5.2.0 released on 12/01/2014

- Several bugs fixed
- New minors features in the core server

Version 5.1.0 released on 08/12/2013

- New features in the web interface
- Several bugs fixed
- New features in the test framework library

Version 5.0.0 released on 15/09/2013

- Several bugs fixed
- New major features in the test framework library
- Improvment of the scheduler

Version 4.2.0 released on 08/04/2013

- Several bugs fixed
- New features in the web interface

Version 4.1.0 released on 10/03/2013

- Several bugs fixed
- New features in the web interface
- CentOS 6 Support
- Improvment of the scheduler

Version 4.0.0 released on 30/01/2013

- Several bugs fixed
- New features in the test framework library
- SSL support for the web interface
- New authentification method with sha1 and salt
- New features in the XmlRPC API

Version 3.2.0 released on 29/09/2012

- Several bugs fixed
- New features in the test framework library

Version 3.1.0 released on 14/07/2012

- Several bugs fixed
- New features in the test framework library
- Improvment of the scheduler
- New features in the XmlRPC API

Version 3.0.0 released on 09/06/2012

- Several bugs fixed
- New features in the XmlRPC API
- Improvment of the scheduler
- New repositories for adapters and backups

Version 2.2.0 released on 28/03/2012

- New majors features in the XmlRPC API
- Several bugs fixed
- New features in the test framework library

Version 2.0.0 released on 27/02/2012

- New features in the XmlRPC API
- Documentation added for the test framework and adapters
- Several bugs fixed
- Probes support

Version 1.2.0 released on 14/01/2012

- Improvment of the scheduler
- New features in the XmlRPC API
- New features in the test framework library
- Interface web added
- Several bugs fixed

Version 1.0.0 released on 13/12/2011

- First official version
- CentOS 5 support
- Several bugs fixed

Version 0.1.0 released on 17/05/2010

- First beta

CHAPTER 4

Client

The client allows to write and execute automatic tests but also to analyze the results in real time or deferred. It also allows you to share tests in a simple and effective way. To use the client, you must have a user account and be able to connect to the test server (tcp/443).

The client can also be used to develop extensions (adapters and libraries) to communicate with the system to be tested or piloted.

Finally the graphical interface changes according to the level of access:

- tester level: write / execute tests, and analyze the results
- admin level: access to all features
- monitor level: read only access

The interface is divided into 3 main parts:

- the workspace
- the analyzer
- the server explorer
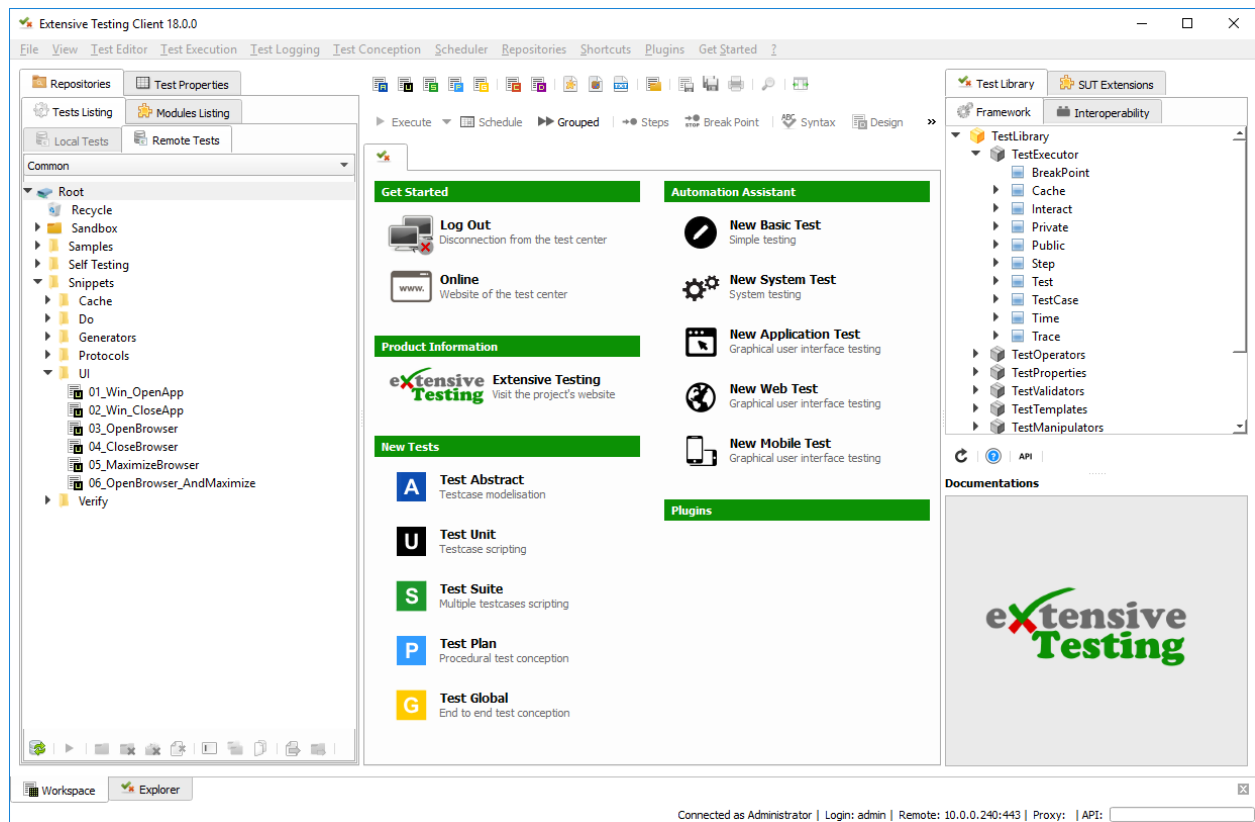
Note: The client is available on Windows and Linux, in 64bits mode

4.1 The workspace

The workspace is composed of 3 main parts:

- access to all file repositories

- access to test design

- online documentation
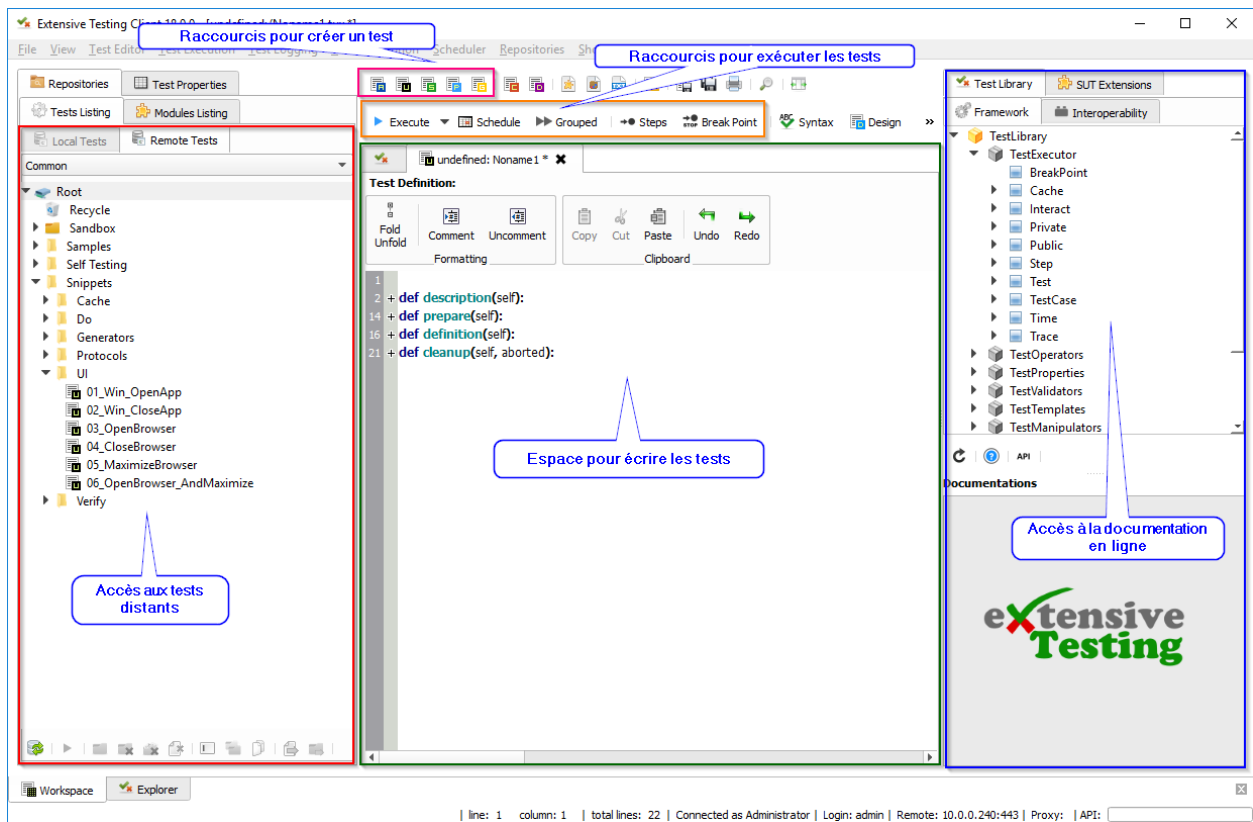
4.1.1 Deposit of tests

The client provides access to the two test repositories: remote and local.

The `remote repository` allows to store its tests on the test server, so to share them with other users. The tree consists of files and directories. Test management can be done from the client. The tests can be organized by project if necessary.
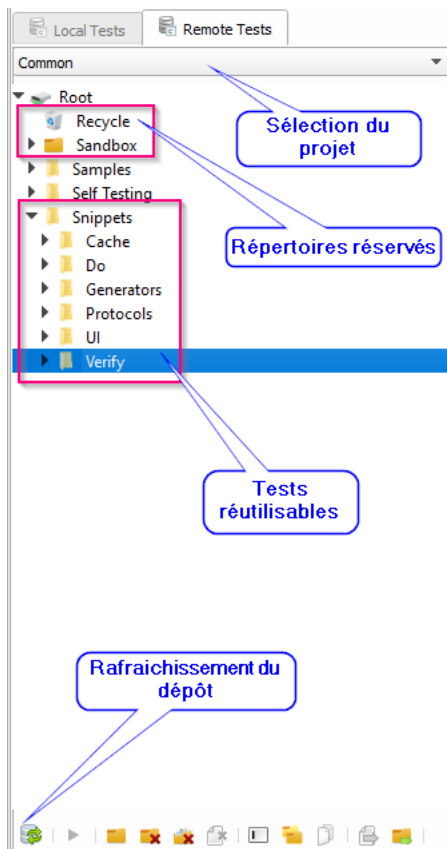
Note: The `Common` project contains re-usable tests and various examples.

Note: The directories `Recycle` and `Sandbox` are reserved directories, delete them is impossible.

Note: It is possible to open a test by dragging and dropping the file to the writing space.

The **local repository** gives the possibility to store his tests on his post, so not shared. This feature is not enabled by default because it is not in the philosophy of the solution to use it. Nevertheless the deposit can be activated through the user's preferences.

Warning: Some features are missing in the local repository, its use is not recommended!

4.1.2 Depositing extensions

The client allows access to the depots of the extensions (adapters and libraries) and can also be used to develop new ones, which will be stored there too. These extensions are organized by version.
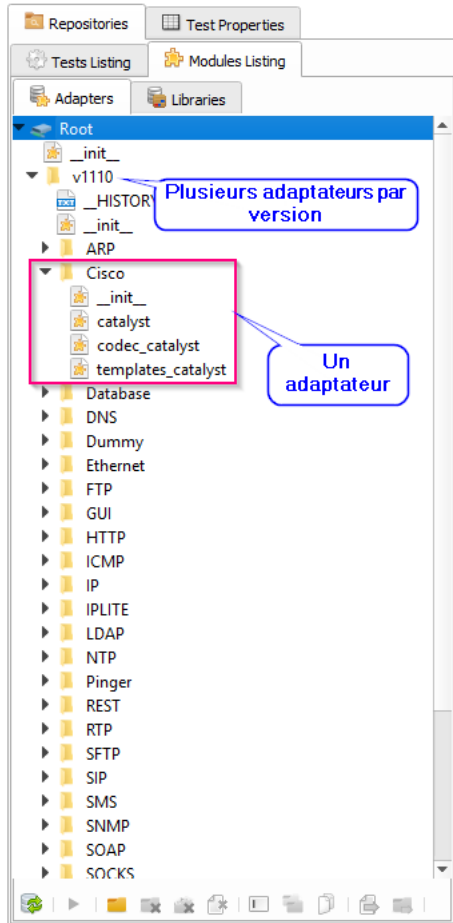
Note: The extensions are developed in `Python`.

4.1.3 Tests properties

The tests can be enriched with a number of properties. Available properties are:

- the description of the test (author, date of creation, etc . . .)

- incoming and outgoing variables

- the definition of agents and probes used by the test

The `Test properties> Test Data> Inputs` window contains the list of variables accessible from the test. Adding variables can be done by right clicking 'Add parameter'.
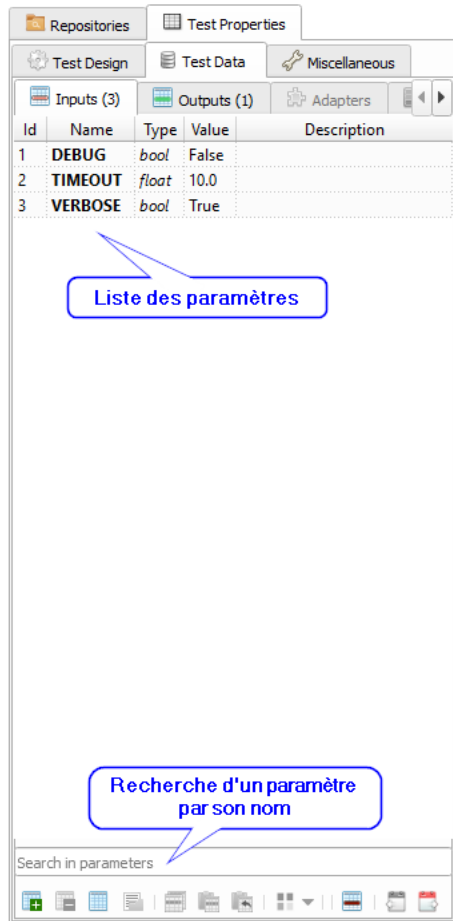
Note: To insert a parameter into a test, just drag & drop.

Note: It is possible to choose the version of the adapters and libraries to use for the test
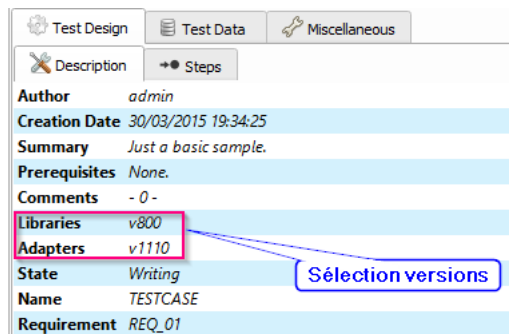
4.1.4 Graphic design

The design of a test in graphical form is possible with the `abstract` type test. This design mode does not require any knowledge in development.
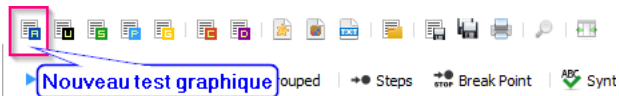
Right click on the drawing area to choose the element to add.

Test Abstract Definition:

Right click to draw your test!

Start

Step #1
start

ACTION6 = HTTP::Client #1
doGET
string: /

OK

KO

Step #1
setPassed

Step #1
setFailed

Template
ACTION14 = getValue
variables: ACTION6

info
variables: ACTION14
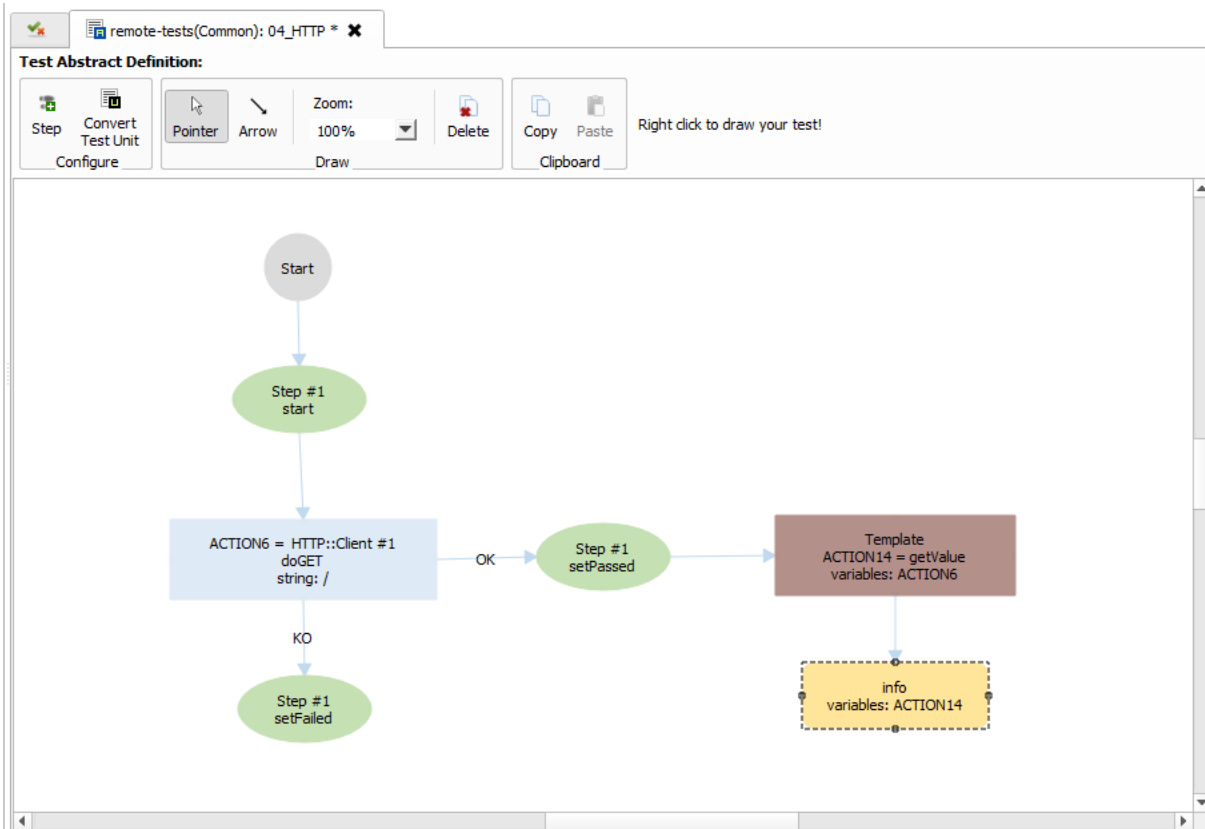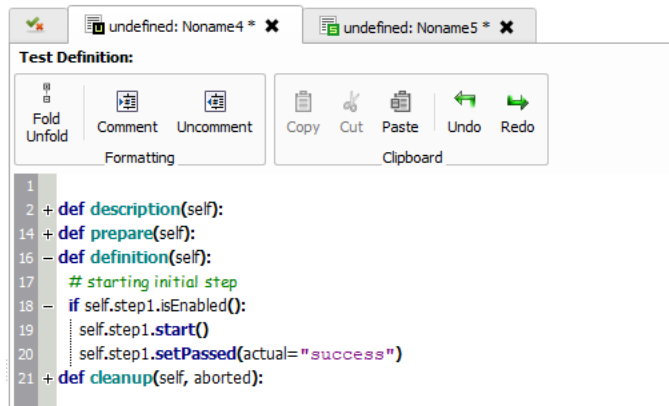
4.1.5 Textual design

The design of a `scripting` test is possible with `unit` and `suite`. This design mode requires knowledge in development, i.e. python.
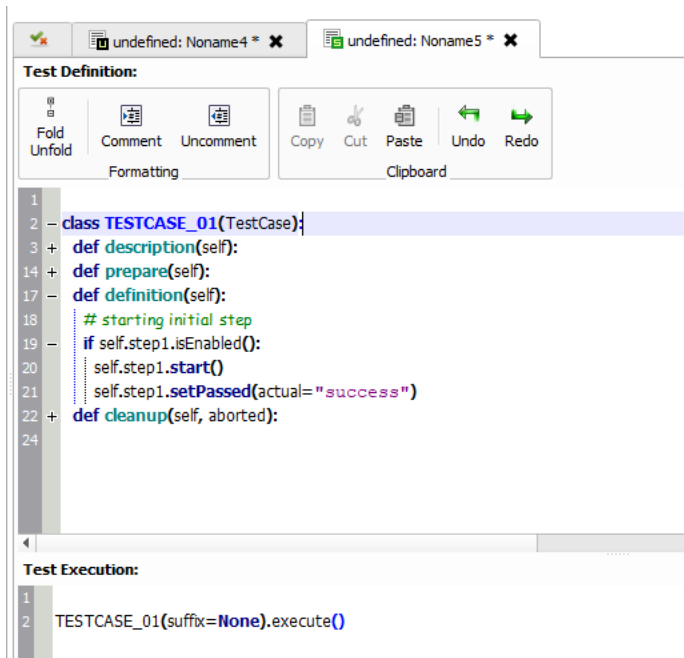
The `unit` test is a test case. It is divided into 4 sections automatically called by the framework.

The `suite` test represents one or more test cases. This type of test allows you to run the same test case by changing the input parameters.

Note: The `Ctrl + F` shortcut allows you to search for text in your tests.

4.1.6 Assisted design

The design wizard allows you to write tests without knowledge in development. It covers the following actions:

- Call to the basic functions of the test framework
- SSH test
- Application test with screenshot (based on the Sikuli project)
- Website test (based on the Selenium project)
- Android mobile app test

The wizard is to describe the actions to perform, and if desired export them to a test unit or suite.
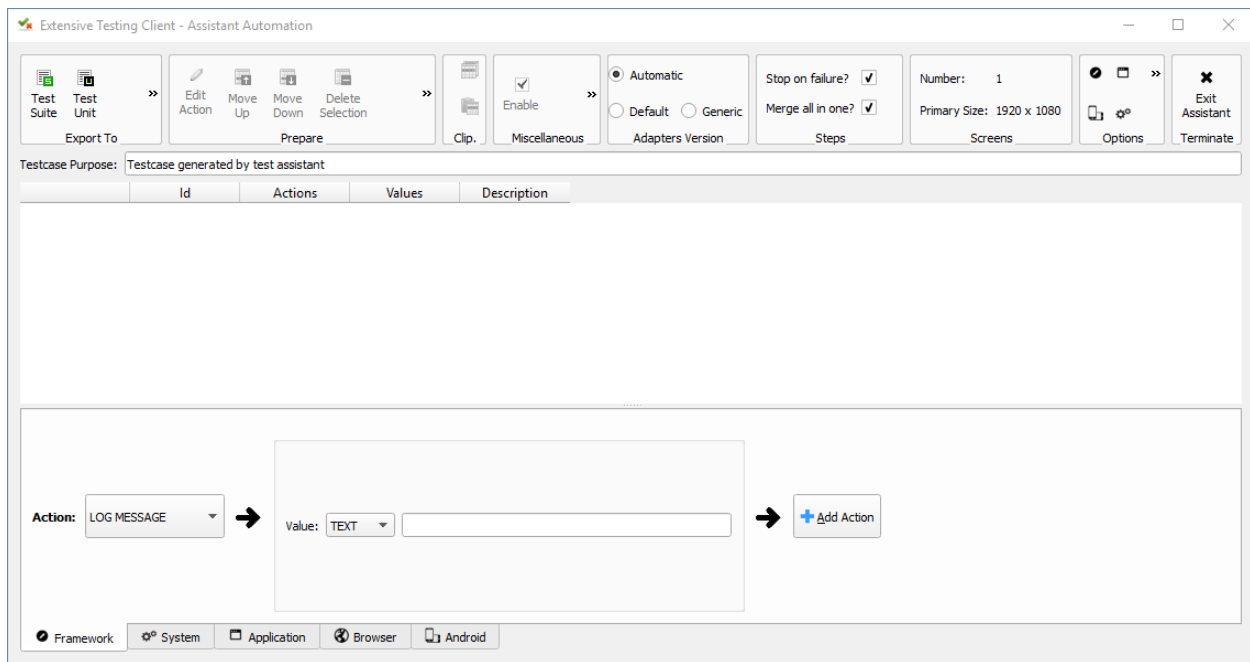
4.1.7 Conditional design

Conditional design allows you to build scenarios or test campaigns. This approach does not require developing knowledge. To perform this type of test, it is necessary to create a new `plan` or `global` test.

The test `plan` makes it possible to write test scenarios by including tests of the type `abstract`, `unit` or `suite`.
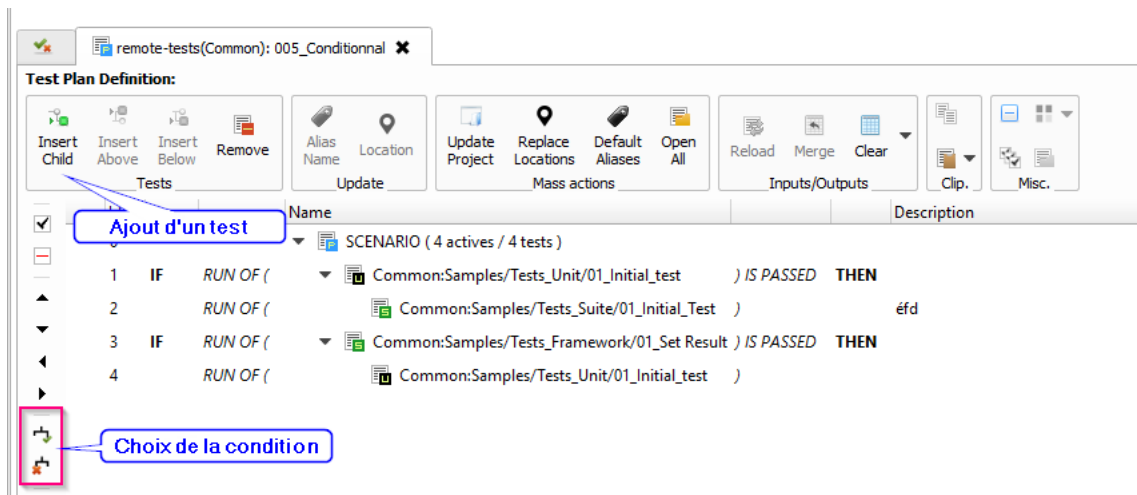
The `global` test is used to describe test campaigns by including tests `plan`, `abstract`, `unit` or `suite`.

Note: It is possible to override the test parameters.

4.1.8 Online documentations

The online documentation is generated by the server, it describes the set of available functions in the test framework and the different extensions.
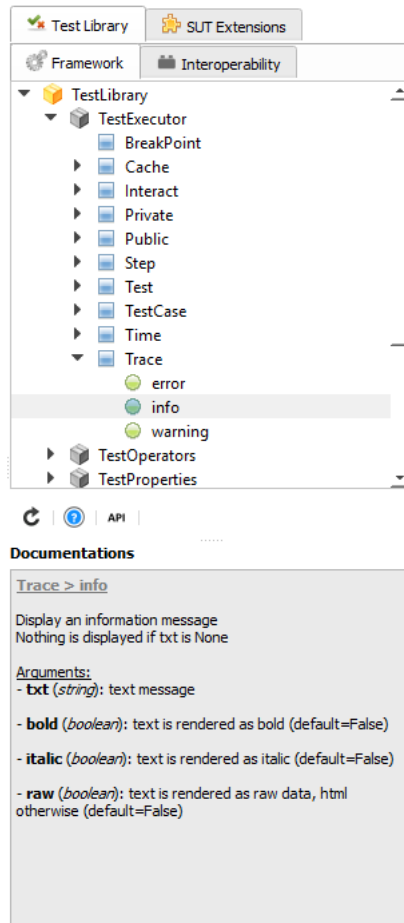
Note: A drag & drop from the documentation on a test automatically inserts the skeleton of the function.

4.2 The analyzer

The analyzer makes it possible to follow the execution of a test in real time or deferred. It makes it possible to display all the events of the test and to facilitate the analysis of the errors.

4.2.1 Visualization of events

Different types of events are possible (column event type):

- DEBUG
- INFO
- WARNING
- ERROR
- SEND
- RECEIVED
- STEP-STARTED
- STEP-PASSED
- STEP-FAILED
- MATCH-STARTED
- MATCH-INFO
- MATCH-STOPPED
- MATCH-EXCEEDED

Note: Filtering on the ERROR event allows you to quickly see why the test is in error.

Note: The SEND | RECEIVED filter is used to display messages sent or received by the system to be tested / piloted.

4.2.2 Detailed view

Selecting an event from the list displays the detailed view. The detailed view displays the content of the event and more.

4.3 Server Explorer

4.3.1 Visualization of the results

The complete history of test results is available from the client. They are sorted by date and time of execution. The client can display the reports and download the logs generated during the execution of the test.

4.3.2 Visualization of test reports

Test reports are visible directly from the client. Two types of reports are available:

- advanced report
- simple report

Note: The reports are exportable in html, xml and csv formats.

4.4 Settings

Client behavior can be changed through the user's preferences.
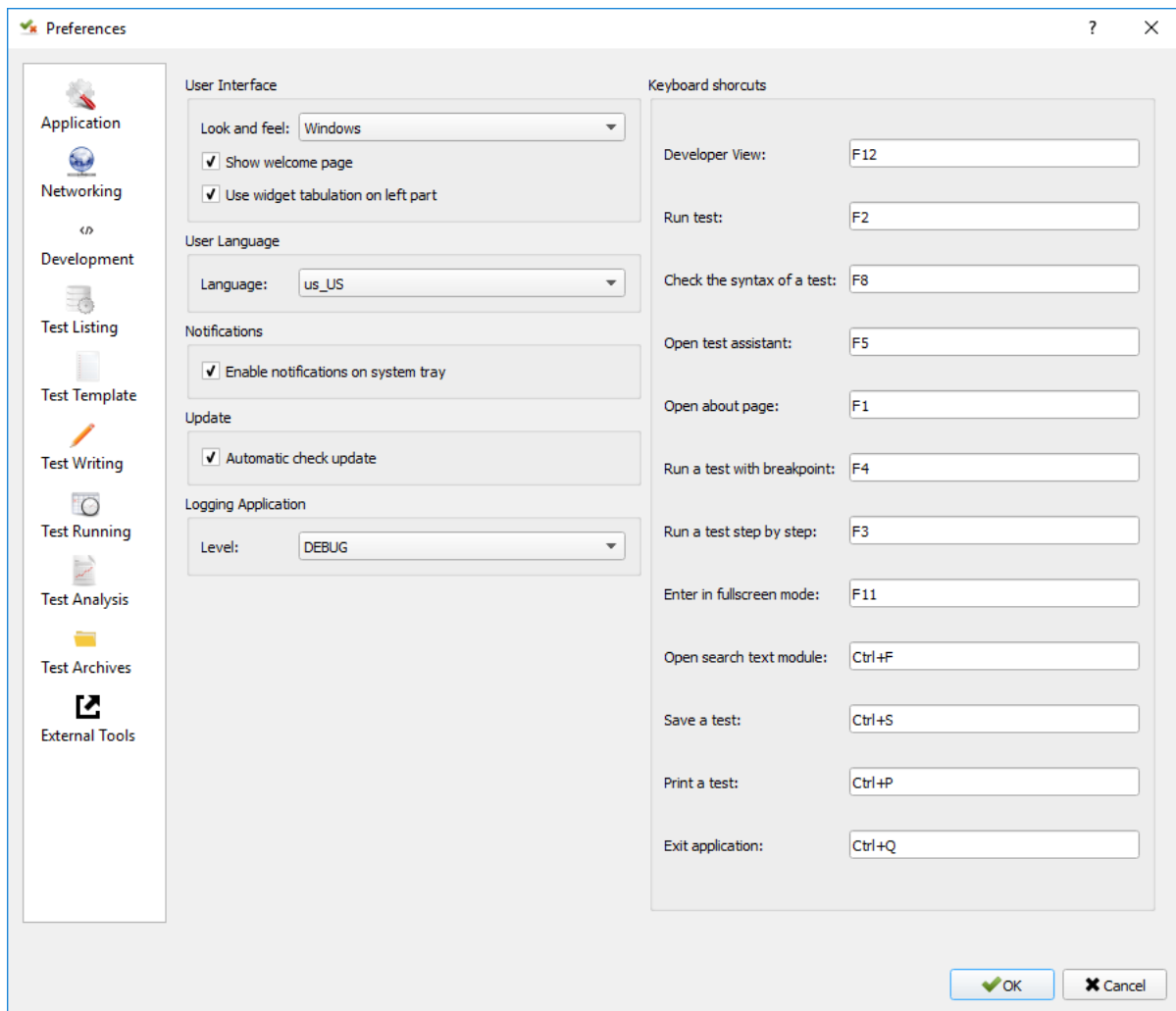
Note: Preferences are stored in the `settings.ini` file.

4.5 Complements

It is possible to add plugins in the client. Plugins are to be added to the `Plugins` directory.
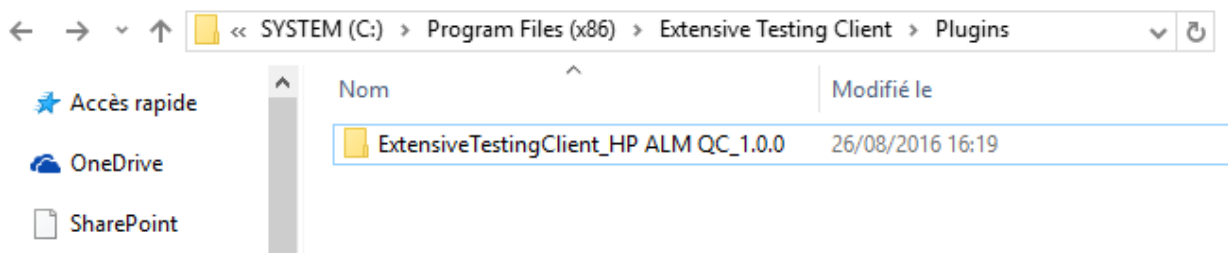
Plugins are accessible in the `Plugins` menu after restarting the client.
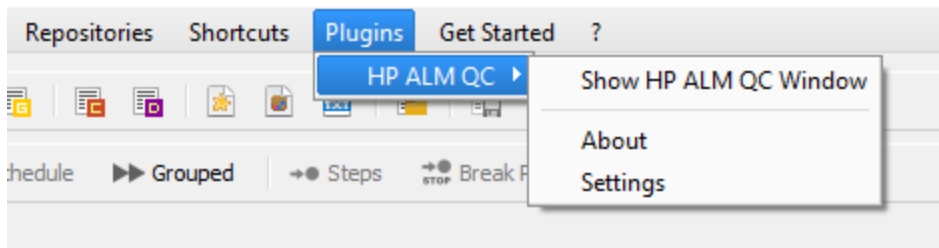
Note: It is necessary to restart the client to take into account the plugins deployed.

4.5.1 HP ALM plugin

The HP ALM plugin allows you to export tests and results from the Extensive Client to HP ALM QualityCenter. This approach makes it possible to be independent with respect to QC.

The configuration of the plugin is done in the page " Settings", it is necessary to configure at least:

- username
- the password
- the domain
- the project

To export a test, you must generate the test design from the client and click on the HP ALM plugin available on the toolbar.
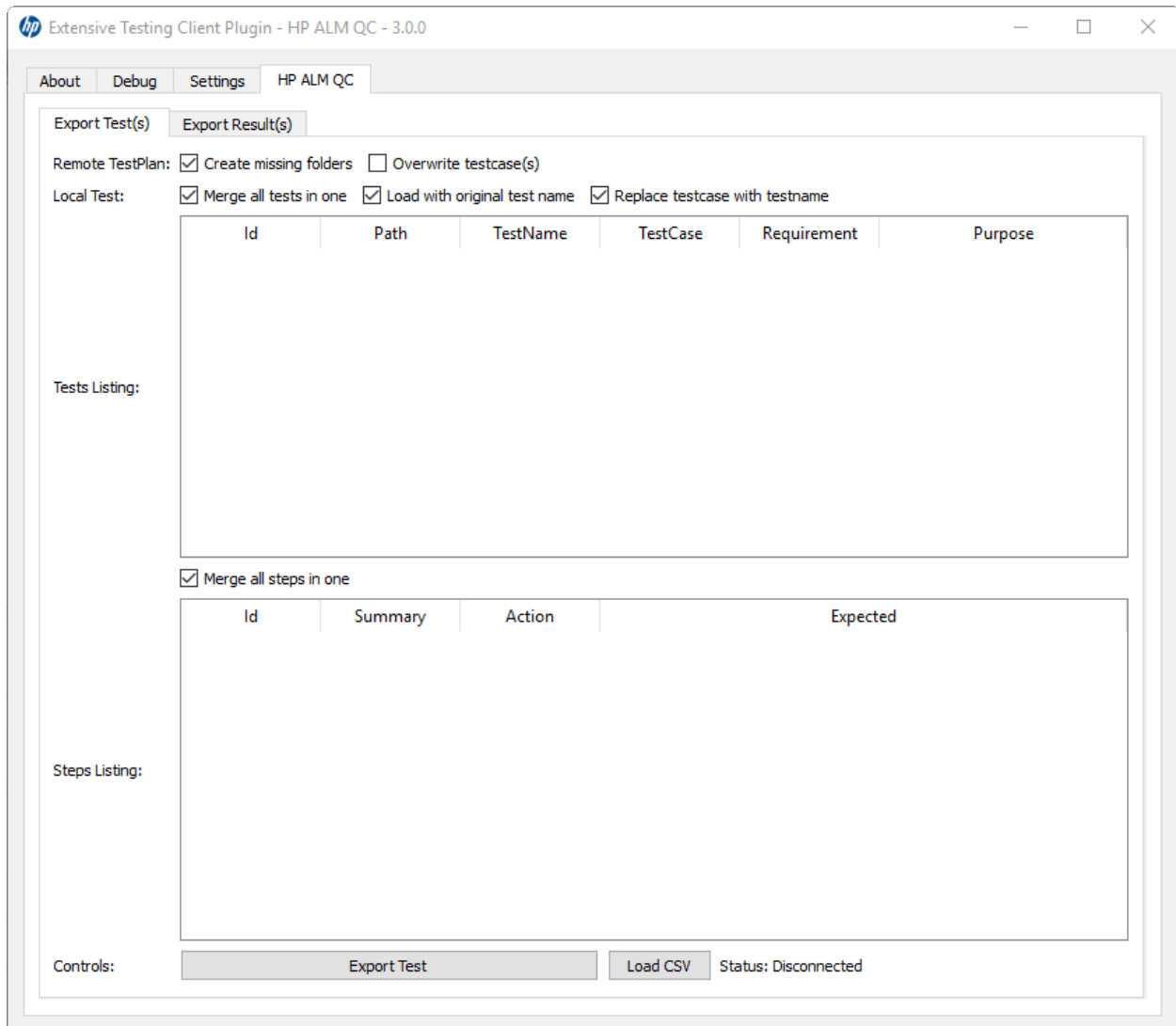
The export of the results can be done from the archive exploration window, The plugin must be available in the toolbar when a test report is loaded.

Note: The plugin is compatible with an HP ALM QC> = 12, the REST API is used.

4.5.2 Jenkins plugin

The Jenkins plugin does not do much in this version . . . It just provides a link to the web interface of its favorite Jenkins.

4.5.3 Shell Recorder Plugin

The Shell Recorder plugin allows you to import a sequence of shell commands into the design wizard and generate the associated test. It allows to replay easily a sequence of commands.

The first step is to import an ssh session (from a putty terminal for example) from the clipboard or by directly importing a text file containing the sequence of shell commands.

The plugin automatically detects the prompt in the sequence to parse the associated commands and results. If the prompt is not detected, it can be changed manually.
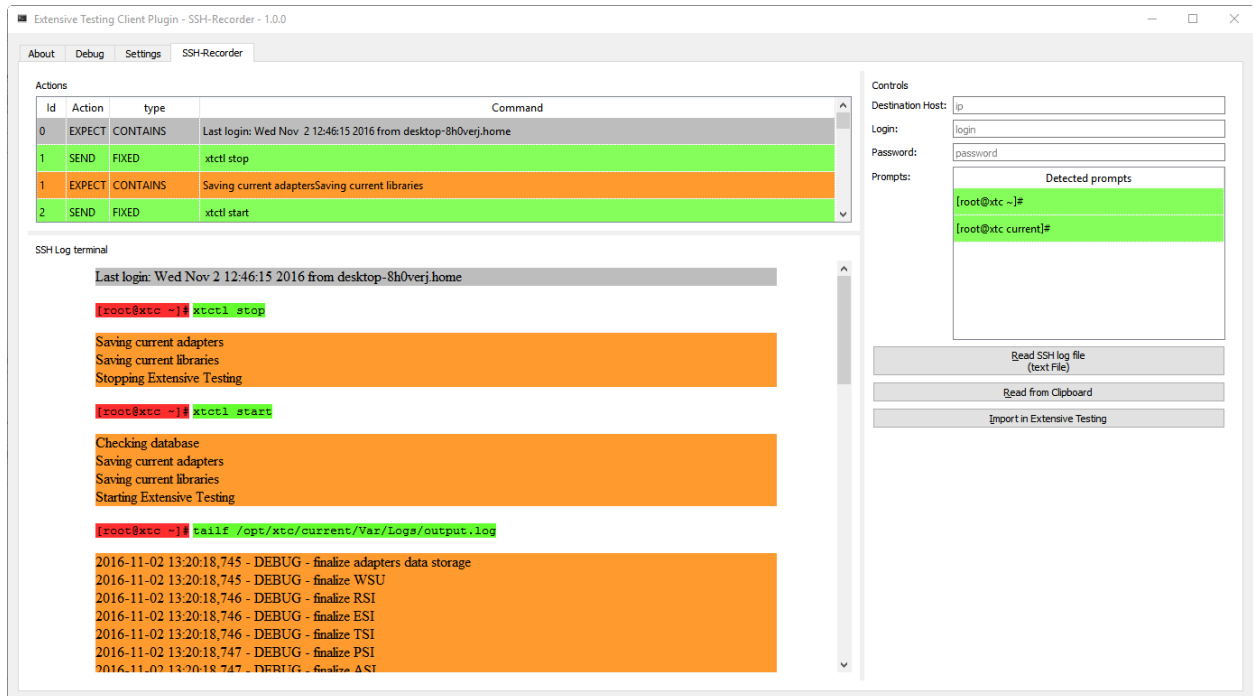
4.5.4 SeleniumIDE Plugin

The use of the `SeleniumIDE` plugin involves basic use. It can convert a file saved with the SeleniumIDE plugin of firefox in the design assistant.

Tip: It is more efficient to use the live assistant to be in tune with the philosophy of the solution.

Toolbox

The toolbox allows you to start agents or probes on dedicated workstations.

- Agents are required to run tests with Selenium on dedicated workstations or to deport the execution of a test.
- Probes can be used to retrieve logs automatically during the execution of a test.
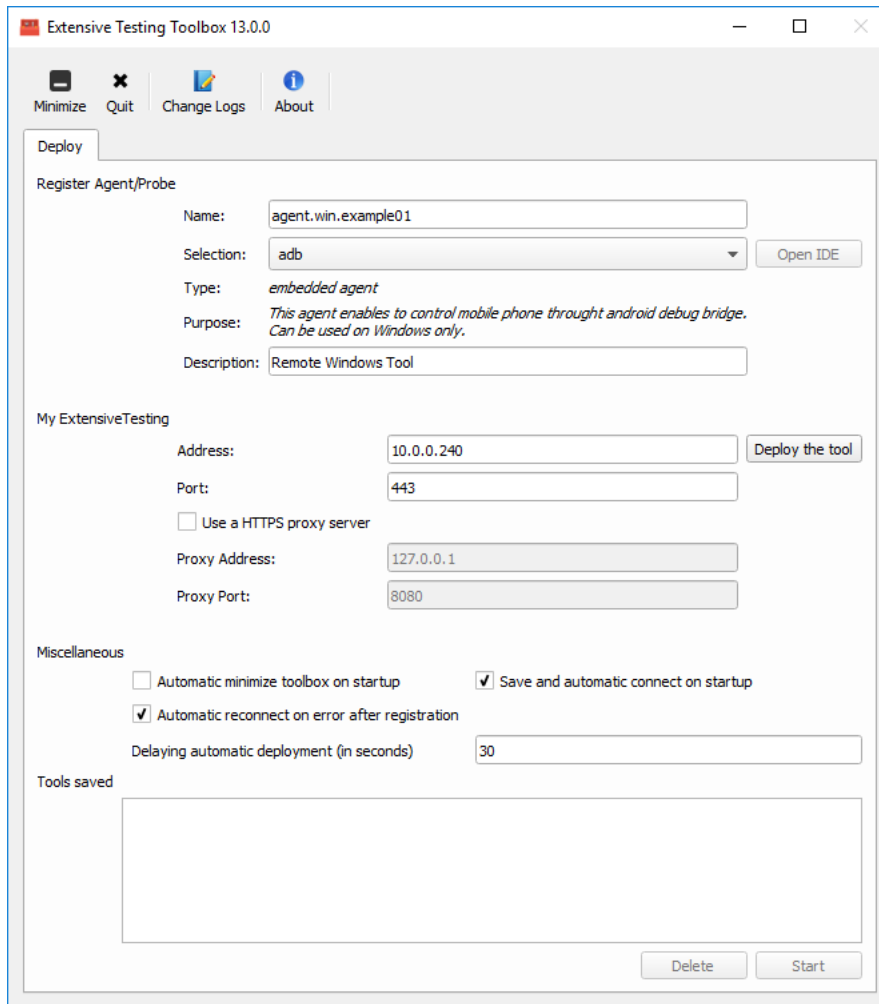
5.1 Deployment

This window allows you to choose the agent or probe to start. The type of agent or probe to start can be chosen in the drop-down list. Finally, an agent or probe needs to be registered with the test server in order to use it.

An agent will allow you to perform a distributed run of your tests. For example, an agent deployed on several machines will allow to run the same test on different environment to test or pilot.

The complete list of available agents and probes are described in the *Server Add-ons> Agents or Probes* chapter.

Note: The agent name or probe must be unique for successful registration.

Tip: For a better visibility of the agents or probes available, it is advisable to respect the following formalism for the names:

[Agent | probe] [Environment] [prénom_testeur] [name] [instance-number]. . .

Example: agent.win.denis.socket01
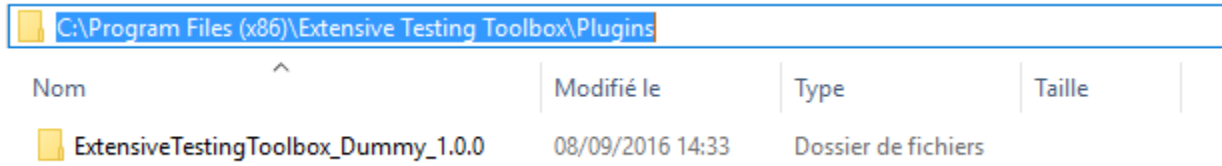
Example of a deployed and running agent:
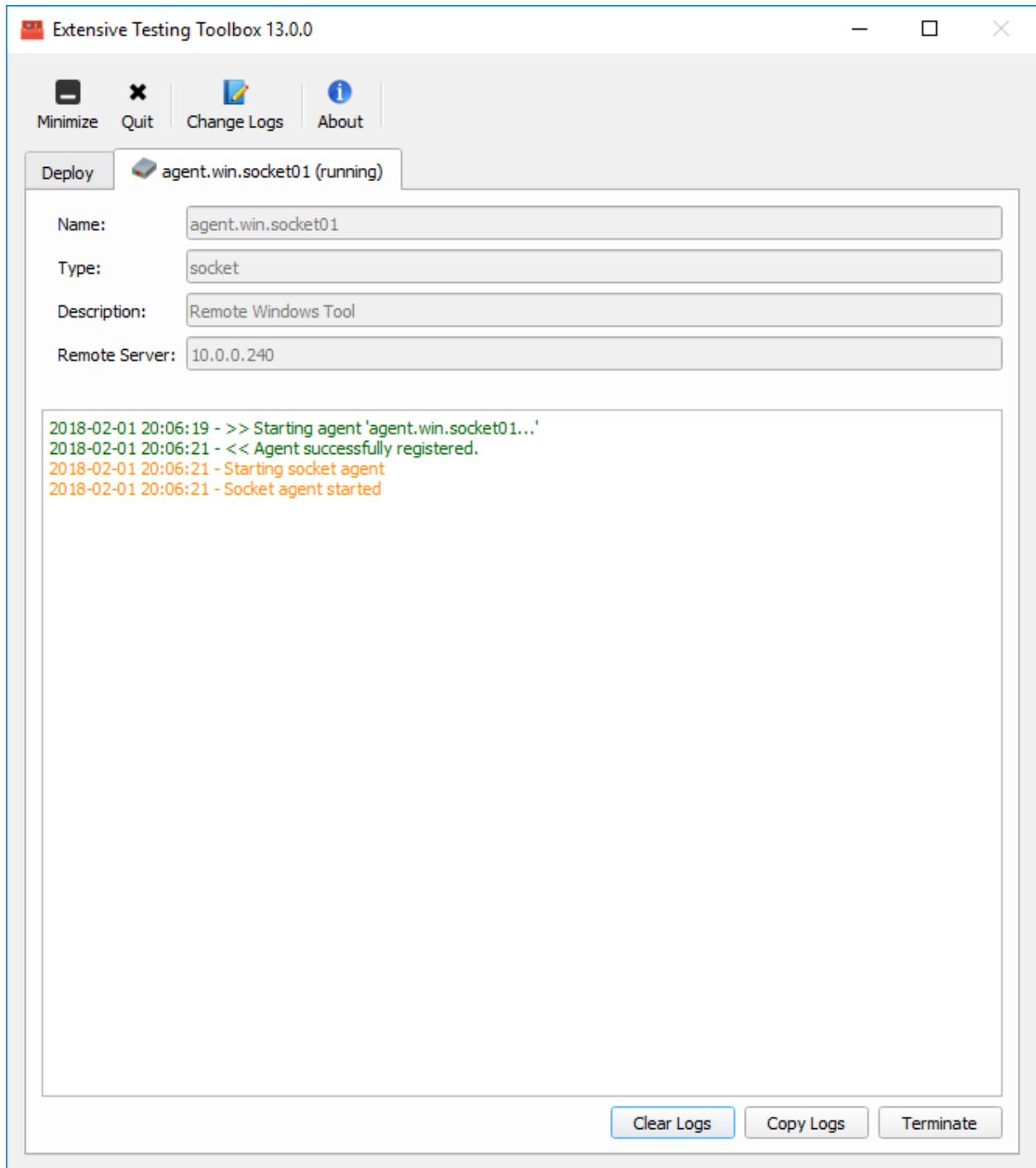
5.2 More

The toolbox can be enriched with new plugins.

To do this, follow the procedure described in the chapter *Contributions> Development plugins> Toolboxes*. Plugins are to be dropped into the Plugins directory.

After restarting the toolbox, the add-in appears in the list of "external" agents
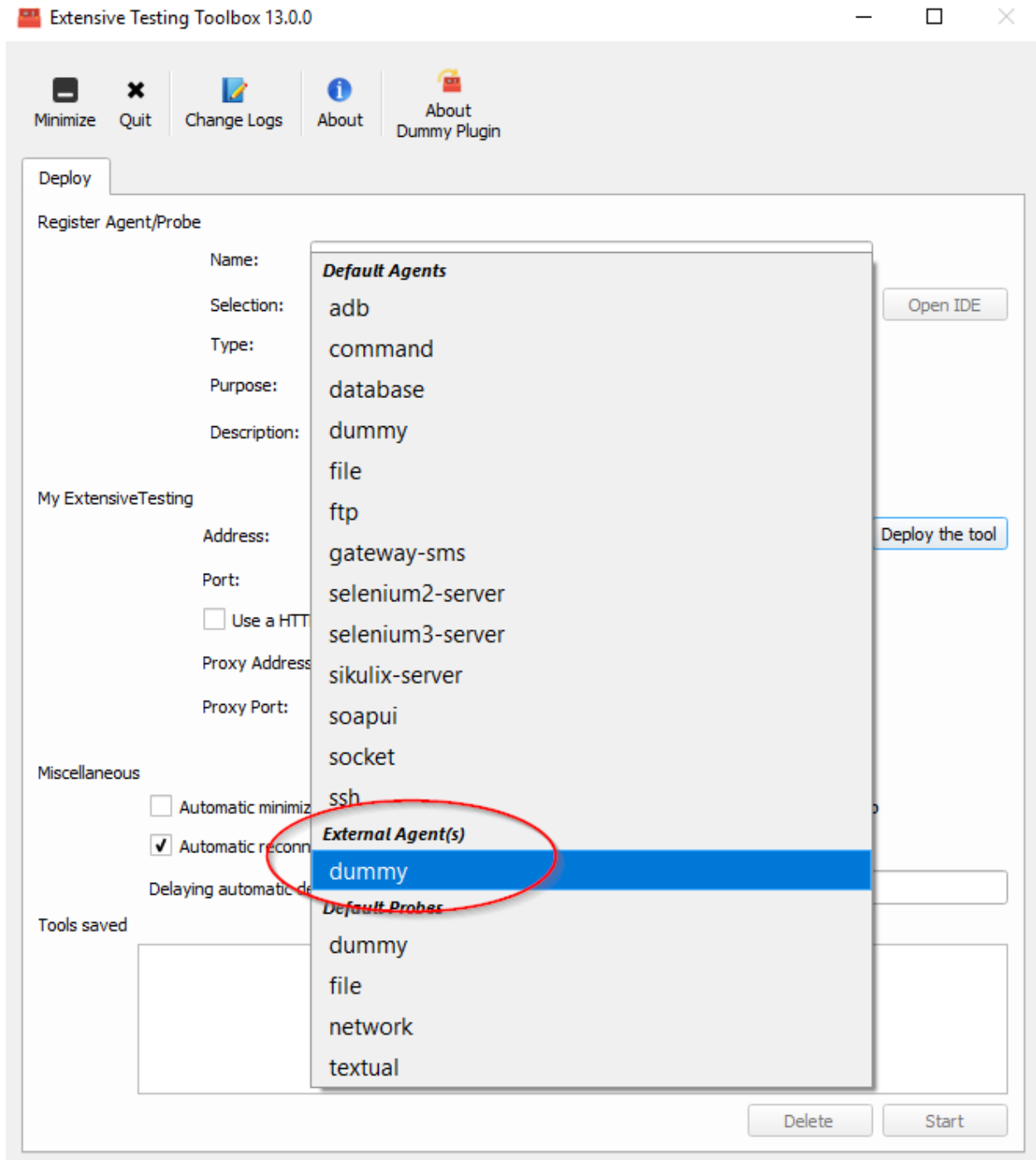
Web interface
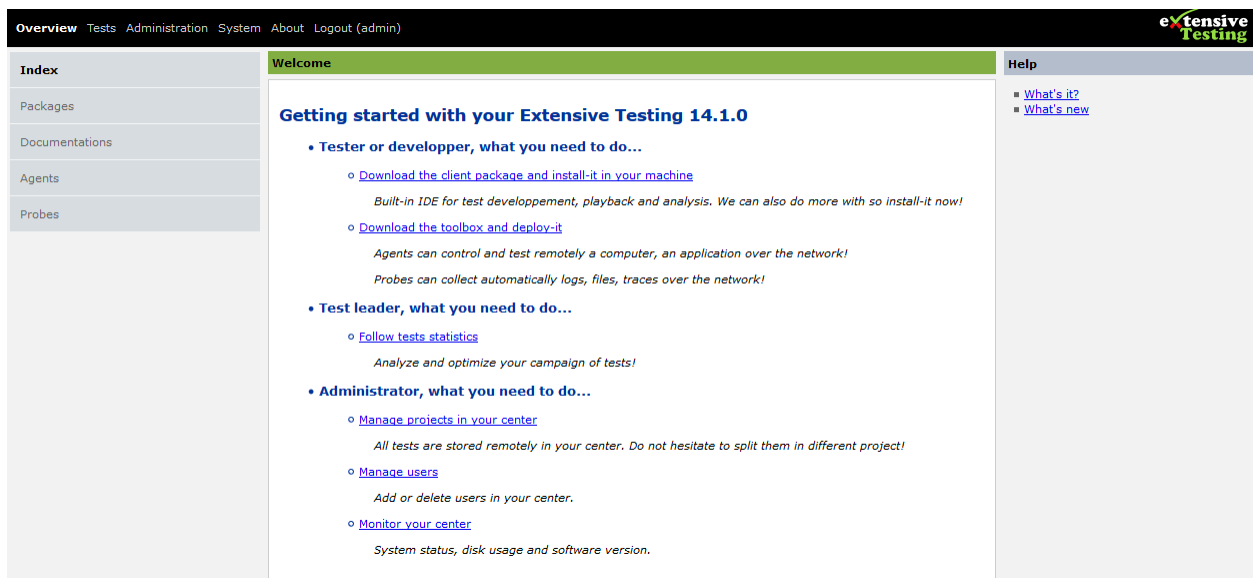
6.1 Tests part

6.1.1 Shared variables

The shared variables enable to describe your dataset. JSON format must be used. These variables are reachable from all tests.

6.2 Admin part

6.2.1 Users

Users account must be created to use properly the product. The creation of users can be done through the web interface or from the rest api.

Some parameters must be provided:

- username
- password
- privilegies (admin, monitor, tester)
- authorized projects

Note: Tests results can be received by email if the account is configured with an email.

6.2.2 Projects

Tests files can be organized per project. The adding or removing of a project can be done from the web interface or the REST api.

Note: The Common project exists by default and can be read from all users, this project cannot be removed.
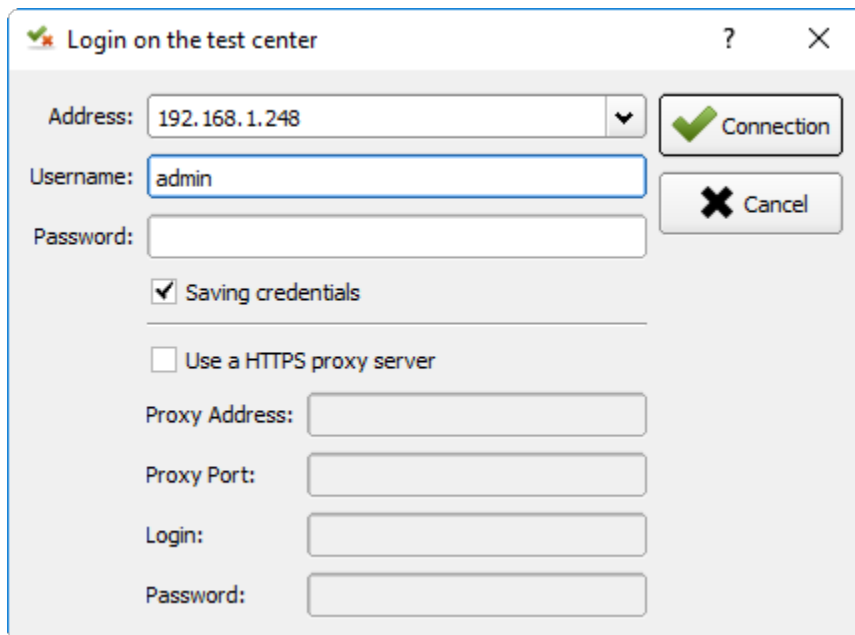
6.3 System part

The system part enable to see the status of the server.

Get started

7.1 Connection to the server

After the oppening of the client, the first step is to connect to the remote server. To do that, you need an account and the remote address of your automation server.

The connection window is available from the `Get Started > Connect` menu or from the welcome tabulation. If the connection is successfull, the user can see all remote tests available.

Note: The `admin` user can be used in the discover of the solution.

7.2 Write a test

The first utilization consists to create a very simple testcase and display a parameter.

1. Create a test of the type `Unit`

2. Add the parameter MON_PARAMETRE of the type `str` with the value "bonjour"

Id	Name	Type	Value
1	DEBUG	bool	False
2	MON PARAMETRE	str	bonjour
3	TIMEOUT	float	10.0
4	VERBOSE	bool	True

Inputs (4)   Outputs (1)   Ac

3. Change the test in the `definition` section to display the value of the parameter.

```
def definition(self):
    # starting initial step
    if self.step1.isEnabled():
        self.step1.start()

        Trace(self).info(txt=input('MON PARAMETRE'), bold=False, italic=False, multiline=False, raw=False)

        self.step1.setPassed(actual="success")
```

Note: It is possible to check the syntax of the test before execution by clicking on the button `Syntax`.

 Syntax
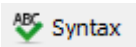
4. Save the test in the repository with the name "Test_A" in the `Sandbox` directory

Local Tests   Remote Tests

Common

- Root
  - Recycle
  - Sandbox
    - Test_A
  - Samples
  - Self Testing
  - Snippets
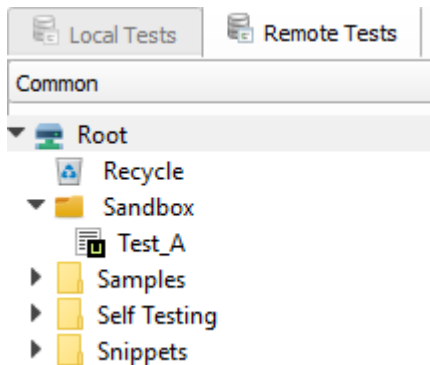
7.3 Write a scenario

Note: This mini guide assumes that you have followed the chapter `Writing a script test`.

The following example explains how to create its first scenario with an overload of test variables.

1. Create a `Plan` type test.

   ![Plan type test icon]

2. Insert test "Test_A" in the scenario. Click on the `Insert Child` button and select the *Test_A* test.
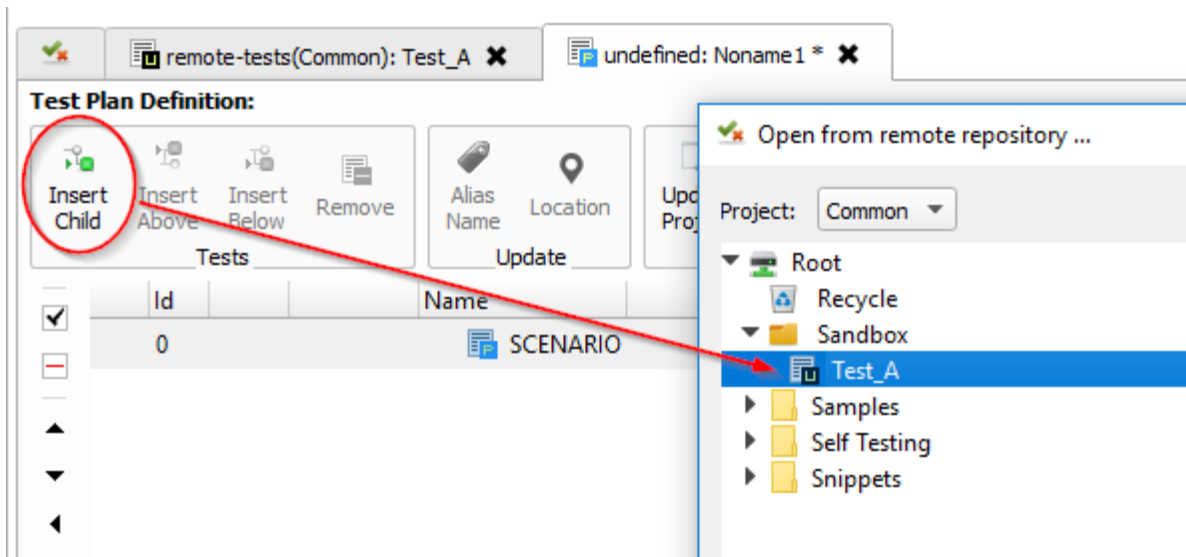
3. After insertion, click on the *Test_A* test and insert the same test again.

4. Save the scenario in the test repository with the name "Scenario_A" in the `Sandbox` directory.

5. Add the parameter MON_PARAMETRE with the value "goodbye" at the scenario level.

Tip: Do not hesitate to define an alias for the name of the test to make the scenario more readable.
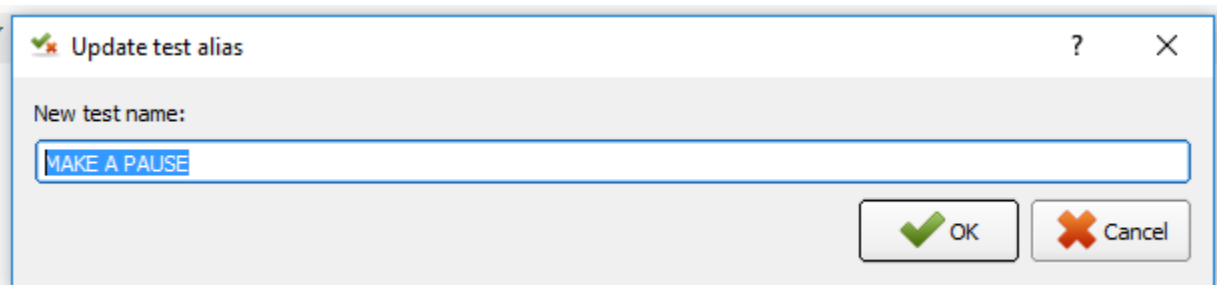
7.4 Execute a test

Note: This mini guide assumes that you have followed the chapters *Writing a script test* and *Writing a scenario*.

You can run a test by clicking the Execute button. Open the *Test_A* and *Scenario_A* tests and run them.
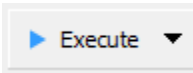
7.5 Analyse des résultats

Note: This mini guide assumes that you have followed the chapters *Writing a script test* and *Writing a scenario*.

The first analysis window shows the execution of the test "Test_A" and in particular the message "hello".

The 2nd analysis window shows the execution of the "Scenario_A" test and in particular the "goodbye" message.

This first usage shows how to run a test and a scenario as well as the overloading of the test variables.

7.6 Best practices

Tip: To keep readability in script type tests, do not use *try / except*. The framework catches all the exceptions at its level.

Tip:

It is essential to take the time to declare the test steps because they allow

- quickly understand the test without the script.
- to have relevant and understandable test reports.

Tip: To facilitate the maintenance of your tests and make them reusable, you should not have hard value in your test. It is necessary to systematically put them in test parameters, it is done for.

Tests examples

8.1 Testcase (unit)

This example shows how to use a test case. A test case consists of 4 sections automatically executed by the test framework as well as associated test parameters.
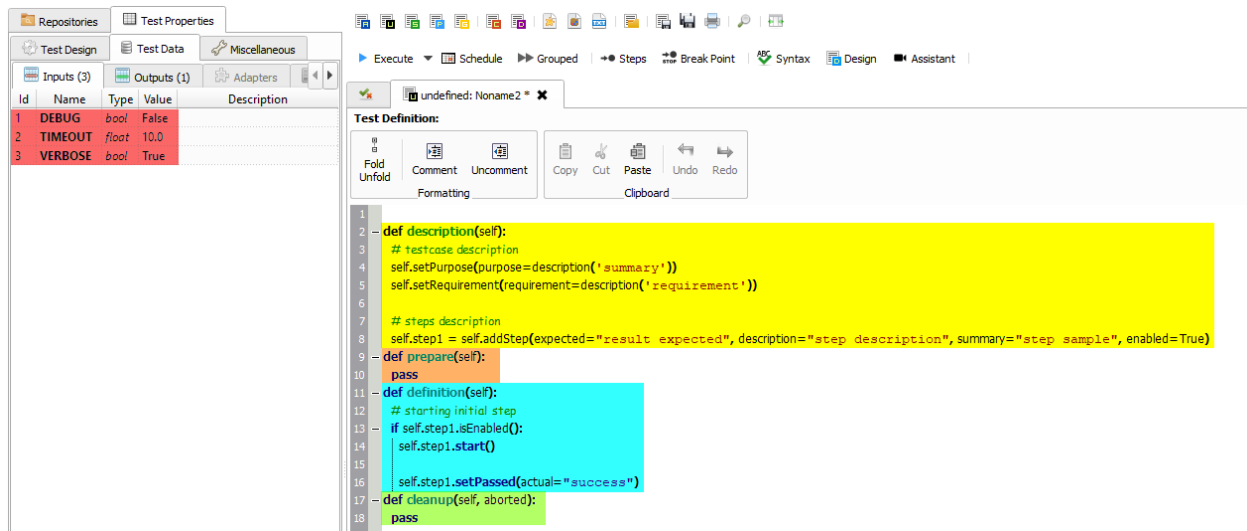
8.2 Testcase (suite)

A test suite allows you to run several test cases afterwards. The example shows how to loop on a test case while modifying the incoming data.

It is therefore possible to add as many arguments as necessary to the `execute()` function and add them identically to the level of the 4 sections.

Note: It is possible to add a prefix at the test case level using the `prefix` argument.

8.3 variables

The variables can be used from a test, there are several types. The example below shows how to retrieve a parameter from its test.

A test parameter can be retrieved at the test level using the *input* function. The name of the parameter to be recovered is to be specified.

8.4 Scenario

A scenario allows you to run several test cases one after the other with result conditions between them. It is possible to override the test parameters at the scenario level.

8.5 Test campaign

A campaign allows you to run multiple scenarios. It is possible to overload the test parameters at the campaign settings level.

8.6 Rest API

To write a REST API test, it is recommended:

- to use the reusable test /Snippets/Protocols/04_Send_JSON
- describe the target server in JSON (ip / destination port, http support)

Example:

The test calls the `httpbin.org` service in https and calls the `ip` service to get the client's actual ip in json.

0				SCENARIO (8 actives / 8 tests)		
5	IF	*RUN OF (*		PREPARE ENVIRONMENT	*) IS FAILED*	**THEN**
16		*RUN OF (*		ABORT ON ERROR	*)*	
30	IF	*RUN OF (*		RETURNS ORIGIN IP	*) IS FAILED*	**THEN**
31		*RUN OF (*		ABORT ON ERROR	*)*	
32		*RUN OF (*		DISPLAY EXTERNAL IP	*)*	
33	IF	*RUN OF (*		RETURNS GET DATA	*) IS FAILED*	**THEN**
34		*RUN OF (*		ABORT ON ERROR	*)*	
35		*RUN OF (*		DISPLAY USERAGENT	*)*	
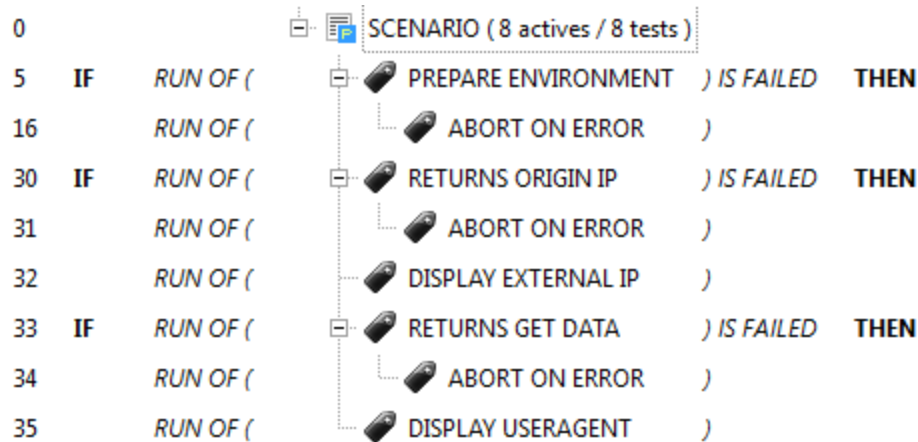
The scenario breaks down into several stages:

1. Preparation of the environment: description of the tested environment (address, network port, etc.) The environment is configured in the *ENVIRONMENT* parameter of the *PREPARE ENVIRONMENT* test (Id = 5)

```
{
  "PLATFORM": {
      "CLUSTER": [
          { "NODE": {
                      "COMMON": {
                          "HOSTNAME": "httpbin"
                      },
                      "INSTANCES": {
                          "HTTP": {
                              "REST": {
                                  "HTTP_DEST_HOST": "httpbin.org",
                                  "HTTP_DEST_PORT": 443,
                                  "HTTP_DEST_SSL": true,
                                  "HTTP_HOSTNAME": "httpbin.org",
                                  "HTTP_AGENT_SUPPORT": false,
                                  "HTTP_AGENT": null
                              }
                          }
                      }
                  }
              }
          ]
  },
  "DATASET": [    ]
  }
```

2. If the environment preparation does not work then the scenario is stopped by calling the test reusable `Snippets/Do/02_Terminate` (Id = 16)

3. A REST request is sent and the expected response is described using the reusable test `/Snippets/Protocols/04_Send_JSON` (Id = 30). If this step does not work then we cancel the test (Id = 31)

The response received is verified by the framework and what was described by the tester in the `HTTP_RSP_BODY` parameter

```
  origin                  [!CAPTURE:EXTERNAL_IP:]

The configuration indicates that the response must verify that the `origin` key is present and
save the value in the cache with the ``EXTERNAL_IP`` key

4. The value received in the response is displayed with the reusable test ``Snippets/Cache/02_Log_
↪Cache`` (Id = 32)
```

Note: The example presented below is available in full in the test samples /Samples/Web_API/
001_httpbin_rest.tpx.

8.7 SSH controls

To write an SSH test, it is advisable:

- to use the reusable test /Snippets/Protocols/01_Send_SSH
- to describe the target server in JSON (ip, account, password at least)

0				SCENARIO (5 actives / 5 tests)		
7	IF	RUN OF (		INITIALIZE ENVIRONMENT	) IS FAILED	**THEN**
9		RUN OF (		ABORT PREPARE ON ERROR	)	
2		RUN OF (		CHECK SERVER STATUS	)	
3	IF	RUN OF (		SHOW SERVER VERSION	) IS PASSED	**THEN**
8		RUN OF (		LOGS CAPTURED INFORMATIONS)		

The test is broken down into several stages:

1. Loading the description (ip, account, password) of the target machine into the cache

2. Calling the /Snippets/Protocols/01_Send_SSH generic test to retrieve the server version The version (if found on the screen) is saved in the cache with the *SERVER_VERSION* key If the version is not found, the test goes into error.

```
# checking server version
xtctl version
.*Server version: [!CAPTURE:SERVER_VERSION:]\n.*
```

3. View the version from the cache.

Note: The complete example is available in the test samples /Self Testing/SYSTEM/000_System.tpx.

8.8 Web browsers

To write a web application test, you must:

- deploy a selenium agent on a machine with a firefox, chrome, internet explorer or edge browser

- have access to the source code of the web page from his browser
- have knowledge of xpath
- know the basics of HTML

The recommended approach for writing web tests is as follows:

- identify the number of pages displayed to script (and the possible reuse of these pages)
- identify the different sequence of pages to create the scenarios
- identify user paths

To perform this type of test, you must declare the agent that will be used
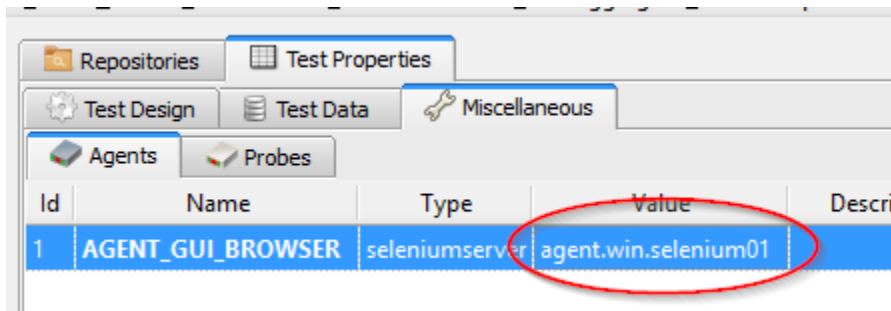
The writing of the tests is done through the assistant. It allows to describe the different stages and generate the equivalent unit test. The sequence of pages are to be described in the flat tests. The user path is to be defined in a global test.

The solution also recommends using only xpath to identify HTML elements.
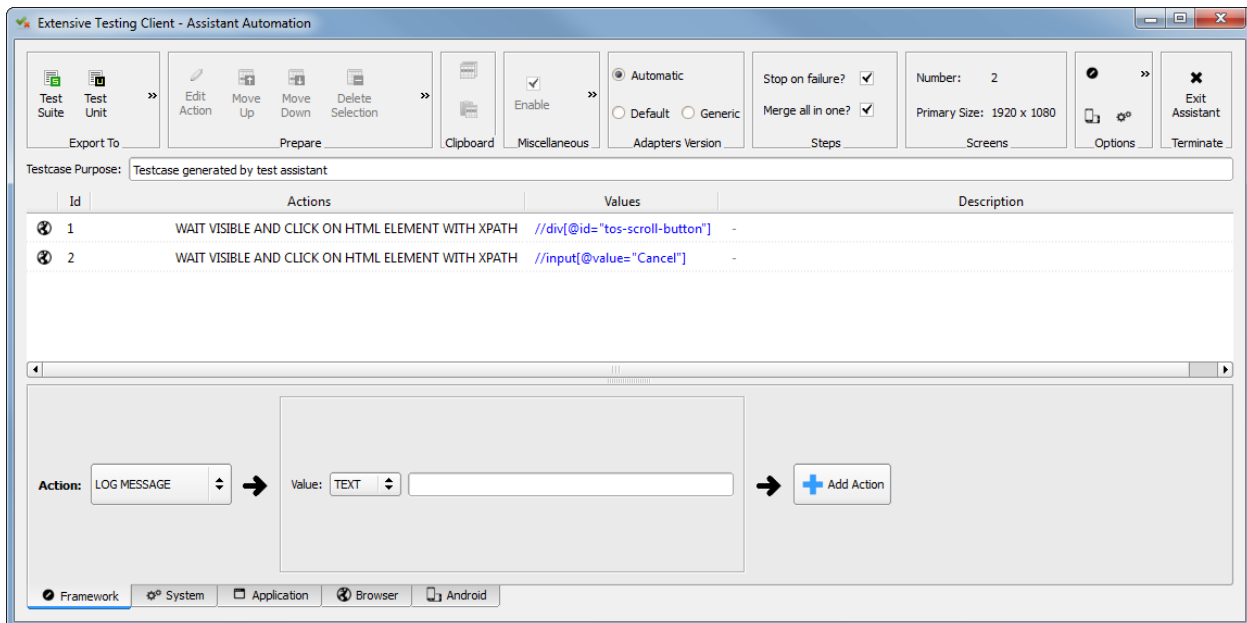
The example below shows how to create a Google Account using a random name and first name.

- Selenium
  - 01_Snippets
    - 01_000 Prepare test data
    - 01_001 Load signin page
    - 01_002 Load page signin_create account
    - 01_003 Configure page signin_create account
    - 01_004 Configure page signin_create account_privacy terms_cancel
  - 02_TestsPlan
    - 02_001 Open browser_load google
    - 02_002 Close browser
    - 02_003 Create account_privacy terms_cancel
  - 03_TestsGlobal
    - 03_001 Create google account and cancel

Example of result:

Tip: It is possible to use browser development tools to validate xpaths.
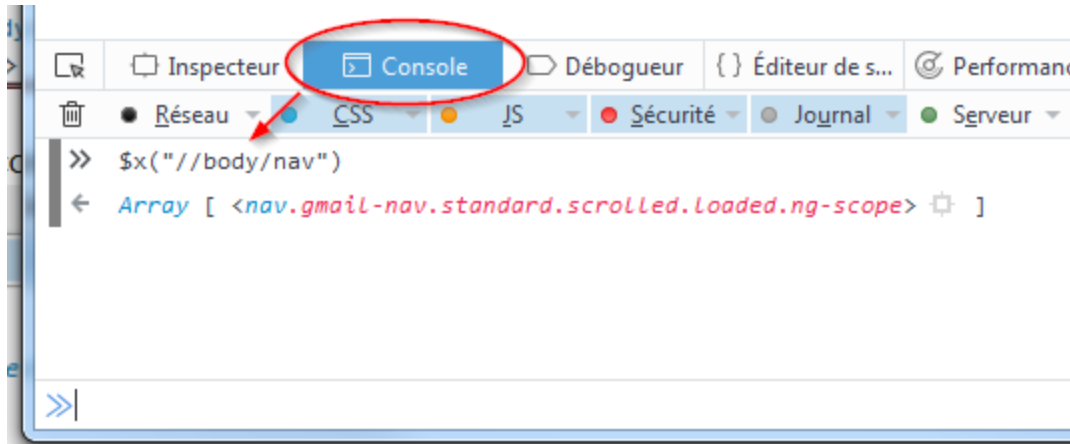
Note: The example presented below is available in full in the test samples `/Samples/Tests_Gui/Selenium/`.

Note: Selenium3 requires at least Java 8 on the client machine.

Browsers	Version Selenium	Gecko
Firefox <47	Selenium 2	Non
Firefox > 47	Selenium 3	Oui
IE	Selenium 3	N/A
Chrome	Selenium 3	N/A

8.9 Android mobile

To write the test of a mobile application, you must:

- Have an Android mobile phone connected in USB on a PC
- Deploy an adb agent on a computer with an android mobile connected to it.
- Have access to the xml description of applications from the agent

Connecting the adb agent on the android mobile requires accepting the RSA key.
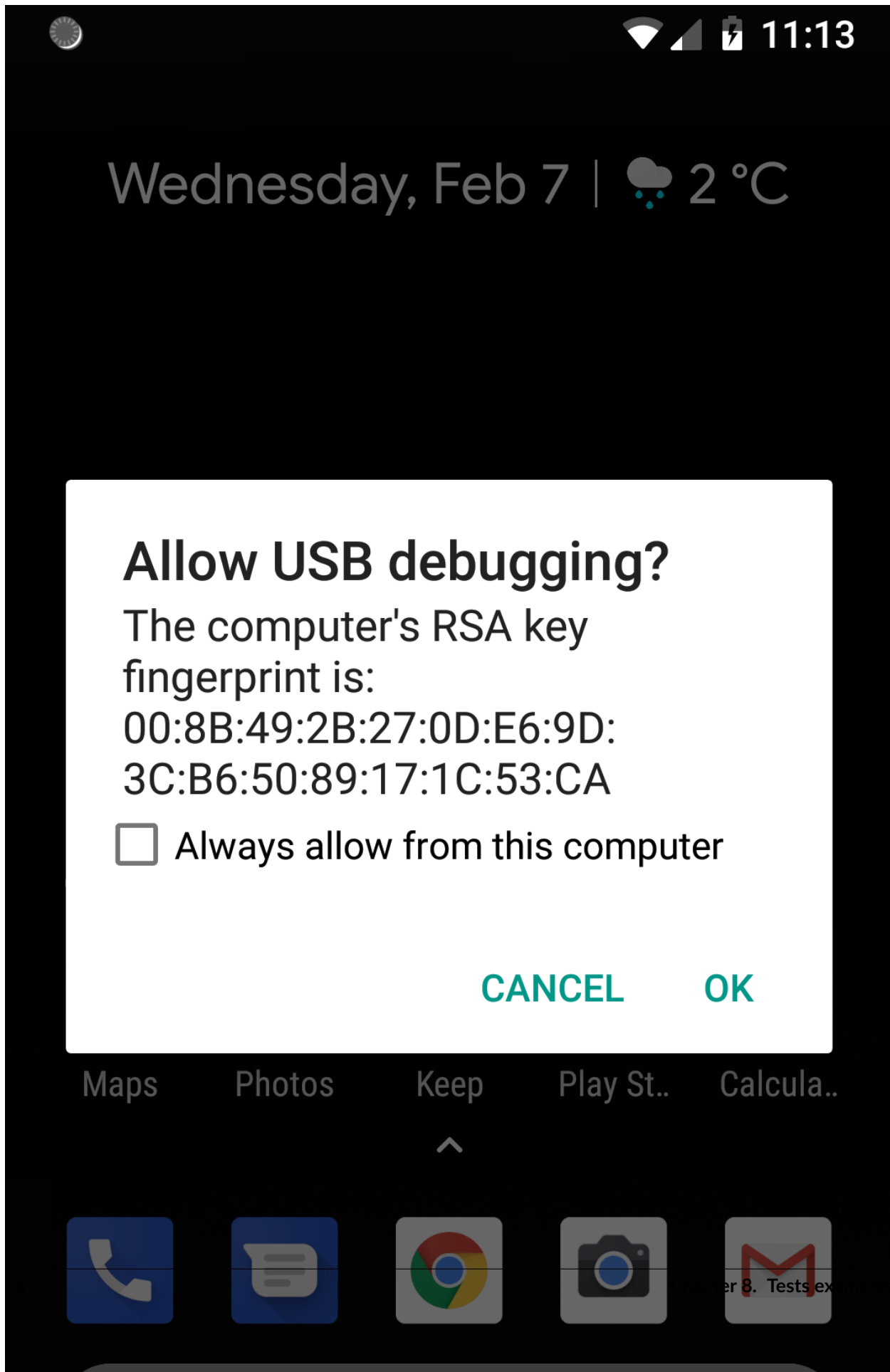
After login, the agent displays a preview of the screen on the pc, it is possible to browse the interface from the agent and have the XML elements available in the page.
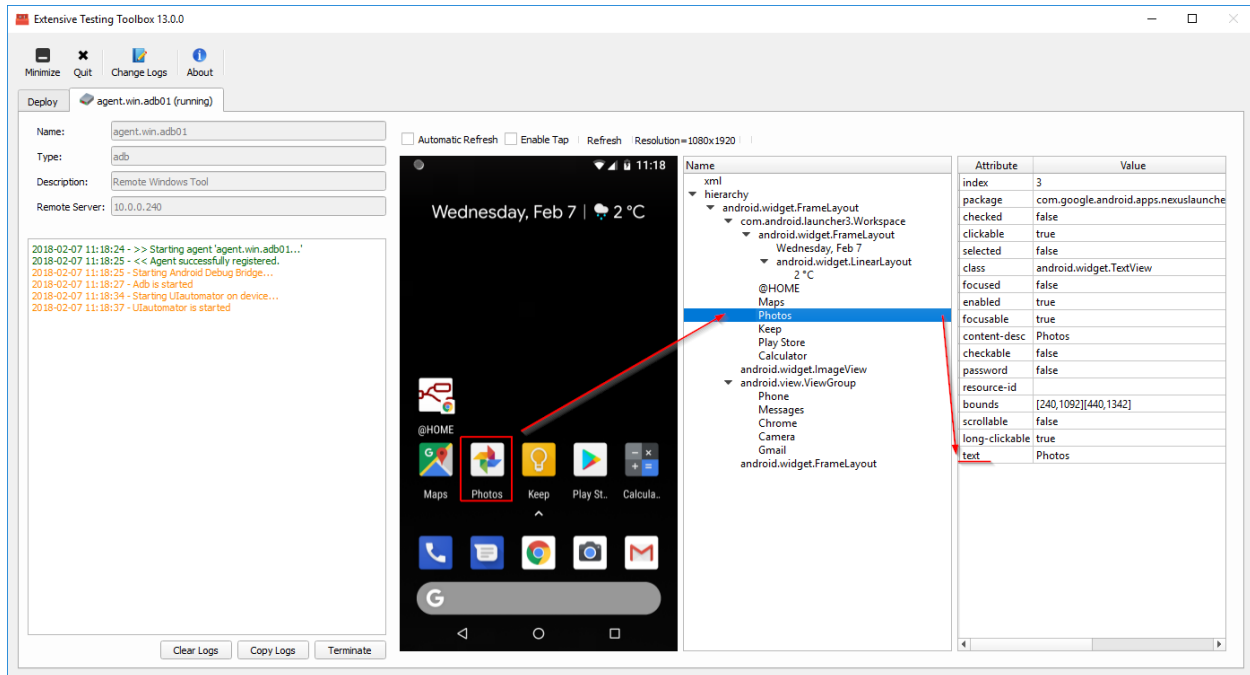
The writing of the tests is done with the assistant. It allows to describe the different stages and generate the equivalent unit test. It is essential to rely on the adb agent for have the list of available XML elements and attributes.
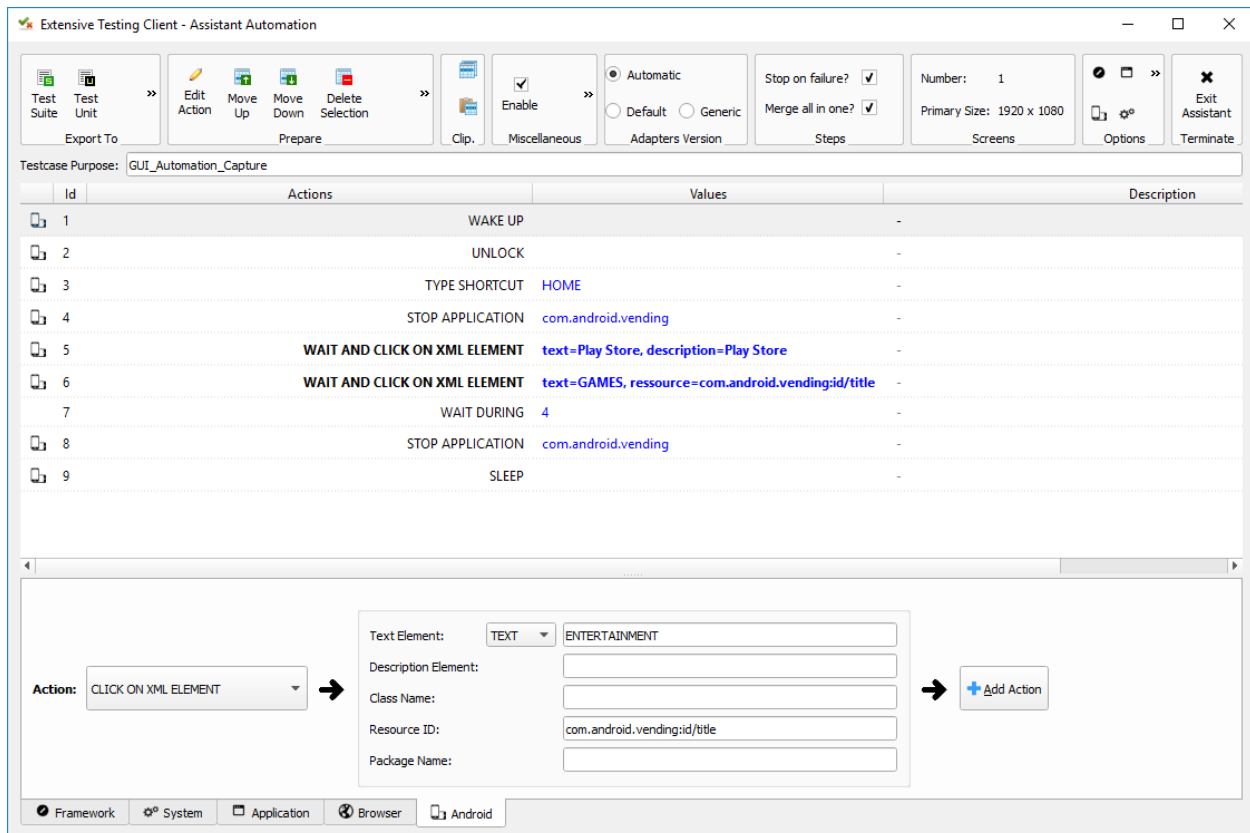
Note: The complete example is available in test samples `/Samples/Tests_Mobiles/03_PlayStore.tux`.

Important: Enabling USB *debug* mode is required on the phone.

Tests Snippets

The interest of reusable tests

- factorize the test database
- reuse the tests
- limit scripting to design scenarios

These types of tests are to be used in `test plan` mode.

9.1 Shared data between tests

9.1.1 Add in the cache a new data

Important: path of the reusable test /Snippets/Cache/01_Set_Cache.tux

This reusable test consists of saving a value in the data cache during the execution of a test.

Parameter(s) to configure:

The `DATAS` parameter contains the list of values to save with the format:

```
# my comment
[!TO:CACHE:<MA_CLE>:];my value
```

Example

```
# Save misc data
[!TO:CACHE:EXAMPLE:];hello world

# Save server information in the cache
[!TO:CACHE:SERVER_DESCRIPTION:];[!FROM:INPUT:TEST_PURPOSE:]
```

Note: It is possible to save several values with this test.

9.1.2 Display a value from the cache

Important: path of the reusable test /Snippets/Cache/02_Log_Cache.tux

This reusable test makes it possible to display the value of a key present in the cache during the execution of the test.

Parameter(s) to configure:

Parameters	Description
MESSAGES	Contains the list of parameters to log in the test

```
# display cache
[!FROM:CACHE:EXAMPLE:]

# log timeout input
[!FROM:INPUT:TIMEOUT:]
```

Note: It is possible to display multiple values at one time

9.1.3 Reset the cache

Important: path of the reusable test /Snippets/Cache/02_Reset_Cache.tux

This reusable test makes it possible to totally empty the cache. No parameters to configure.

Note: This test can be used when several scenarios are chained together in a global test.

9.1.4 Checking a value in the cache

Important: path of the reusable test /Snippets/Cache/02_Checking_Cache.tux

This reusable test makes it possible to check the value in a key present in the cache.

Parameter(s) to configure:

Parameters	Description
CHECKING	List of values to check in the cache

Operators available:

Parameters	Description
contains	Check if the value contains a string
matches	Check if the value matches the regular expression
==	Check if the value equals
! =	Check if the value is different from
>	Check if the value is greater than
<	Check if the value is less than
> =	Check if the value is greater than
<=	Check if the value is less than

```
# Check if value contains the test string
[!FROM:CACHE:EXAMPLE:] contains test
```

Note: It is possible to check multiple values at one time

9.1.5 Delete entry from the cache

Important: path of the reusable test /Snippets/Cache/02_Delete_Cache.tux

This reusable test is used to delete a entry in the cache according to the key.

Parameter(s) to configure:

Parameters	Description
MESSAGES	List of keys to delete

```
# delete the key EXEMPLE from the cache
[!FROM:CACHE:EXEMPLE:]
```

Note: It is possible to delete several keys at one time

9.2 Basics actions

9.2.1 Hold on a test

Important: path of the reusable test /Snippets/Do/01_Wait.tux

This reusable test allows you to wait for xx seconds while the test runs.

Parameter(s) to configure:

Parameters	Description
DURATION	duration in seconds

9.2.2 Stop a test

Important: path of the reusable test `/Snippets/Do/02_Terminate.tux`

This reusable test makes it possible to force the stopping of a scenario on error occurences.

Note: It is possible to customize the stop message by setting the variable `STOP_TEST_MSG`.

9.2.3 Load test environment

Important: path of the reusable test `/Snippets/Do/03_Initilize.tux`

This reusable test is used to load the test environment data into the cache (ip addresses, server access account, etc.).

An environment is described with 4 levels:

- environment
- cluster
- node
- instance

An `environment` may consist of one or more clusters.

```
{
  "PLATFORM": {
            "NOM_CLUSTER_1": [ .. ],
            "NOM_CLUSTER_2": [ .. ]
    }
}
```

A `cluster` consists of a list of nodes.

```
{
  "NOM_CLUSTER_1": [
            { "NOM_NOEUD_1": { .. },
            { "NOM_NOEUD_2": { .. }
    ]
}
```

A `node` consists of one or more instances.

```
{
  "NOM_NOEUD_1": {
            "COMMON": { ... },
            "INSTANCES": {....}
    }
}
```

An `instance` is made up of several keys / values.

```
{
  "INSTANCES": {
            "TYPE_INSTANCE_1": {
                                "NOM_INSTANCE_1": { ...},
                                "NOM_INSTANCE_2": { ...}
                       },
            "TYPE_INSTANCE_2": { ... }
      }
}
```

Parameter(s) to configure:

Parameters	Description
ENVIRONMENT	Link to a shared variable or directly contains JSON.

Example of a test environment containing an http server with an instance of type rest. After loading into the cache, the REST instance is accessible by using the NODE_HTTP_REST key. All keys in COMMON are automatically copied to each instance.

```
{
  "PLATFORM": {
    "CLUSTER": [
      { "NODE": {
                "COMMON": {
                    "HOSTNAME": "httpbin"
                },
                "INSTANCES": {
                    "HTTP": {
                        "REST": {
                            "HTTP_DEST_HOST": "httpbin.org",
                            "HTTP_DEST_PORT": 443,
                            "HTTP_DEST_SSL": true,
                            "HTTP_HOSTNAME": "httpbin.org",
                            "HTTP_AGENT_SUPPORT": false,
                            "HTTP_AGENT": null
                        }
                    }
                }
            }
        }
    ]
},
"DATASET": [    ]
}
```

The DATASET key can contain datasets.

9.3 Data Generators

9.3.1 Hash SHA

Important: path of the reusable test /Snippets/Generators/01_Gen_Sha.tux

This reusable test is used to generate a hash of a value and store it in the cache.

Parameter(s) to configure:

Parameters	Description
DATA_IN	Hash character string
CACHE_KEY	Key name
SHA	Type of hash realize (sha1, sha256, sha512)

9.3.2 Hash MD5

Important: path of the reusable test /Snippets/Generators/01_Gen_Md5.tux

This reusable test is used to generate md5 hash and store it in the cache.

Parameter(s) to configure:

Parameters	Description
DATA_IN	Hash character string
CACHE_KEY	Name of the key or the result will be saved in the cache

9.3.3 UUID

Important: path of the reusable test /Snippets/Generators/01_Gen_Uuid.tux

This reusable test is used to generate an uuid and store it in the cache.

Parameter(s) to configure:

Paramètres	Description
CACHE_KEY	Name of the key to save the result in the cache

9.3.4 BASE64

Important: path of the reusable test /Snippets/Generators/01_Gen_Base64.tux

This reusable test is used to encode or decode a string and store the result in the cache.

Parameter(s) to configure:

Parameters	Description
CACHE_KEY	Name of the key to save the result in the cache
DECODE	Set to True to encode
ENCODE	To set to True to decode
URLSAFE	Set to True if the result after encoding is to be used in an url
STR_BASE64	Character string to encode / decode

9.3.5 GZIP

Important: path of the reusable test /Snippets/Generators/01_Gen_Gzip.tux

---

This reusable test can compress or uncompress a string and store the result in the cache.

Parameter(s) to configure:

Parameters	Description
CACHE_KEY	Key name
COMPRESS	To set to True to compress
UNCOMPRESS	Set to True to decompress
STR_GZIP	Character string to compress / decompress

9.4 Networks protocols

9.4.1 SSH

Important: path of the reusable test /Snippets/Protocols/01_Send_SSH.tsx

---

This reusable test is used to send a sequence of ssh commands. It is used in conjunction with the reusable test /Snippets/Do/03_Initilize.tux to load an environment into the cache.

Parameter(s) to configure:

Parameters	Description
SERVERS	List of servers to contact
COMMANDS	List of commands to run on the remote machine
TIMEOUT_CONNECT	Max time to connect to the remote machine

The *COMMANDS* parameter is waiting for one or more blocks of 4 lines. Each block must respect the following formalism:

1. A comment explaining the action, this information is used to initialize the test step

2. The command to execute

3. The string expected on the screen, if the expected value is not found then the step will be in error. (optional line)

4. empty

Warning: Each block will be executed even if the previous one is in error.

The following example performs the following actions:

1. Send 3 pings on the remote machine whose ip is stored in the DEST_HOST cache

2. Verification of having the message on the screen indicating that the 3 packets have been sent. Then the mddev value is stored in the cache with the STATS key

---

3. The second block clears the screen by sending the clear command.

4. Finally the test is waiting to find the prompt on the screen

```
# send a ping
ping -c 3 [!CACHE:SVR:DEST_HOST:]
.*3 packets transmitted, 3 received, 0% packet loss.*mdev = [!CAPTURE:STATS:] ms.*

# clear the screen
clear
.*root@.*
```

Note: It is possible to run the test multiple times by providing a server list.

Note: By default, the test waits for a maximum of 20 seconds to find the expected string. This value can be configured with the TIMEOUT parameter.

Note: By default, the test waits 10 seconds to connect to the remote server. This value can be configured with the TIMEOUT_CONNECT parameter.

9.4.2 HTTP

Important: path of the reusable test /Snippets/Protocols/01_Send_HTTP.tsx

---

This reusable test makes it possible to send an HTTP request by checking the response received. It is used in conjunction with the reusable test /Snippets/Do/03_Initilize.tux which loads an environment into the cache.

Parameter (s) to configure to set the destination:

Parameters	Description
SERVERS	List of servers to contact
TIMEOUT_CONNECT	Max time to connect to the remote machine

Parameter (s) to configure the HTTP request to send:

1	HTTP_REQ_BODY	custom	{ "login": "[!CA [...]
2	HTTP_REQ_HEADERS	custom	Content-Type: applic [...]
3	HTTP_REQ_METHOD	str	POST
4	HTTP_REQ_URI	custom	/session/login

Parameter (s) to configure the expected HTTP response (and which will allow to consider the test as valid):

Parameters	Description
HTTP_RSP_BODY	Body of the expected answer.
HTTP_RSP_CODE	The expected HTTP code. 200 by default
HTTP_RSP_HEADERS	List of expected headers
HTTP_RSP_PHRASE	The expected HTTP sentence. OK by default
HTTP_RSP_VERSION	The expected HTTP version. HTTP / 1. [0|1] default

6	HTTP_RSP_BODY	custom	
7	HTTP_RSP_CODE	str	200
8	HTTP_RSP_HEADERS	custom	set-cookie: session_ [...]
9	HTTP_RSP_PHRASE	str	OK
10	HTTP_RSP_VERSION	str	HTTP/1.1

Note: The use of regular expressions is possible to check or save a value in the body of the answer or in the headers.

9.4.3 XML

Important: path of the reusable test /Snippets/Protocols/01_Send_XML.tsx

This snippet enable to send HTTP request with XML in body. The response can be checked too. This snippet should be used with /Snippets/Do/03_Initilize.tux to load the test environment in the cache.

Parameter(s) to configure the remote destination:

Parameters	Description
SERVERS	List of servers to test
TIMEOUT_CONNECT	Timeout to connect on the remote machine

Parameter (s) to configure the HTTP request to send:

Parameters	Description
HTTP_REQ_BODY	Request body
HTTP_REQ_HEADERS	List of headers to add
HTTP_REQ_METHOD	HTTP method (GET, POST, etc..)
HTTP_REQ_URI	URI

Parameter(s) to configure the expected HTTP response (and the test will be pass in this case):

Parameters	Description
HTTP_RSP_BODY	Xpaths to check
HTTP_RSP_CODE	HTTP code expected. 200 by default
HTTP_RSP_HEADERS	List of expected headers
HTTP_RSP_NAMESPACES	List of namespaces
HTTP_RSP_PHRASE	HTTP phrase expected. OK by default
HTTP_RSP_VERSION	HTTP version expected. HTTP/1.[0|1] by default

> **Warning:** The test will be failed if the response does not content XML.

9.4.4 JSON

> **Important:** path of the reusable test `/Snippets/Protocols/01_Send_JSON.tsx`

This snippet enable to send a HTTP request with JSON in body and to check the associated response. This snippet should be used with `/Snippets/Do/03_Initilize.tux` to load the test environment in the cache.

Parameter(s) to configure to set the remote machine:

Parameters	Description
SERVERS	List of remote machine to reach
TIMEOUT_CONNECT	Timeout of connection with the remote machine

Parameter(s) to configure the request to send:

Parameters	Description
HTTP_REQ_BODY	Request body
HTTP_REQ_HEADERS	List of header to add
HTTP_REQ_METHOD	HTTP method (GET, POST, etc..)
HTTP_REQ_URI	URI to call

Parameter(s) to configure the expected response (and the test will be pass in this case):

Parameters	Description
HTTP_RSP_BODY	List of xpath to check
HTTP_RSP_CODE	HTTP code expected. 200 by default
HTTP_RSP_HEADERS	List of header expected
HTTP_RSP_PHRASE	HTTP phrase expected. OK by default
HTTP_RSP_VERSION	HTTP version expected. HTTP/1.[0|1] by default

> **Warning:** The test will be failed if the response does not content JSON.

9.5 User Interface

9.5.1 Open application in Windows

Important: path of the reusable test `/Snippets/UI/01_Win_OpenApp.tux`

This snippet enable to open a application on a Windows or Linux machine. The parameter `AGENT_GUI` must be configured with the agent to use.

Parameter(s) to configure:

Parameters	Description
APP_PATH	Application path to open

9.5.2 Close an application in Windows

Important: path of the reusable test `/Snippets/UI/02_Win_CloseApp.tux`

This snippet enable to close a application on a Windows or Linux machine. The parameter `AGENT_GUI` must be configured with the agent to use.

Parameter(s) to configure:

Parameters	Description
APP_NAME	Name of the application to close

9.5.3 Open a web browser

Important: path of the reusable test `/Snippets/UI/03_OpenBrowser.tux`

This snippet enable to open a browser on a Windows or Linux machine. The parameter `AGENT_GUI_BROWSER` must be configured with the agent to use.
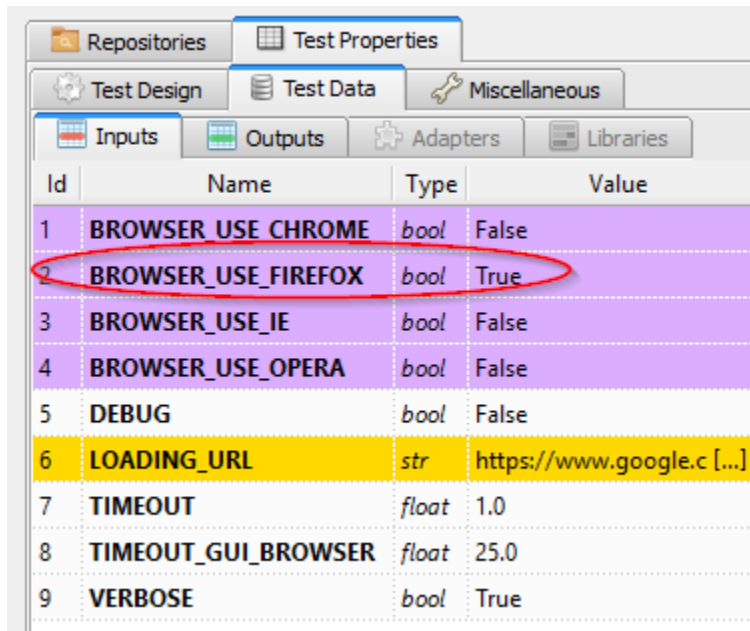
Parameter(s) to configure:

Parameters	Description
LOADING_URL	Website url to load

It's possible to select the browser to user, the following browsers are supported:

- Firefox
- Chrome

- Internet Explorer
- Opera
- Edge

Note: the url must started with `http://` or `https://`

9.5.4 Close a web browser

Important: path of the reusable test `/Snippets/UI/03_CloseBrowser.tux`

This snippet enable to close a browser on a Windows or linux machine. The parameter `AGENT_GUI_BROWSER` must be configured with the agent to use.

9.6 Checks

9.6.1 XML checks

Important: path of the reusable test `/Snippets/Verify/01_Check_XML.tux`

This snippet enable to check a XML content with xpath.

Parameter(s) to configure:

Parameters	Description
XML_STR	raw XML to inspect
XML_XPATH	xpath
XML_NAMESPACES	namespaces definitions

Example of value for the XML_STR parameter:

```
<NewDataSet>
<Table>
  <Country>France</Country>
  <City>Le Touquet</City>
</Table>
<Table>
  <Country>France</Country>
  <City>Agen</City>
</Table>
<Table>
  <Country>France</Country>
  <City>Cazaux</City>
</Table>
<Table>
  <Country>France</Country>
  <City>Bordeaux / Merignac</City>
</Table>
<Table>
  <Country>France</Country>
  <City>Bergerac</City>
</Table>
</NewDataSet>
```

Example of value for the XML_XPATH parameter.

```
(//NewDataSet/Table)[1]/City  [!CAPTURE:CITY:]
```

The value will be accessible from the cache with the CITY key.

9.6.2 JSON checks

Important: path of the reusable test /Snippets/Verify/01_Check_JSON.tux

---

This snippet enable to check JSON content with jsonpath

Parameter(s) to configure:

Parameters	Description
JSON_STR	Json to inspect
JSON_XPATH	jsonpath

Example of value for the JSON_STR parameter:

```
{
 "args": {},
```

```
"headers": {
"Connection": "close",
"Host": "httpbin.org",
"User-Agent": "ExtensiveTesting"
},
"origin": "190.117.217.129",
"url": "https://httpbin.org/get"
}
```

Example of value for the JSON_XPATH parameter.

```
headers.Connection    [!CAPTURE:CX:]
```

The value will be accessible from the cache with the CX key.

CHAPTER 10

Global variables

Global variables (or shared variables) are sued to describe test environment. Variables are accesible from a test from the pamarater of the type `global` or `list-global`.

10.1 Add/delete a variable

The adding or removing of a variable can be done from the web interface or the REST API. `JSON` must be used in variable. There are autommatically availables from tests in properties.

10.2 Describe environment test

The description of a test environment must be respect the following rules. This type of init must be used with the reusable test /Snippets/Do/03_Initialize.tux

Node declaration SAMPLE_NODE:

```
{
    "COMMON": {
        "HOSTNAME": "extensiveautomation"
    },
    "INSTANCES": {
        "SSH": {
            "ADMIN": {
                "SSH_DEST_HOST": "127.0.0.1",
                "SSH_DEST_PORT": 22,
                "SSH_DEST_LOGIN": "root",
                "SSH_DEST_PWD": "",
                "SSH_PRIVATE_KEY": null,
                "SSH_PRIVATE_KEY_PATH": null,
                "SSH_AGENT_SUPPORT": false,
                "SSH_AGENT": {
```

```
                                "type": "ssh",
                                "name": "agent.ssh01"
                        }
                }
        }
    }
}
```

Data test declaration SAMPLE_DATASET_AUTH:

```
{
            "login":        "admin",
            "password":     ""
}
```

Environment declaration SAMPLE_ENVIRONMENT:

```
{
      "PLATFORM": {
            "CLUSTER": [
                    { "NODE": "Common:SAMPLE_NODE" }
            ]
      },
      "DATASET": [
            { "AUTH": "Common:SAMPLE_DATASET_AUTH" }
      ]
}
```

10.3 Import/export variables

It's possible to export or import in mass the variables from the web interface in CSV format

---

Warning: Variables are encoded in base64.

Assisted designs

The client contains a automation assistant to create tests without knownledge in development. The assistant can be used for:

- Use the basic functions of the framework
- Execute system commands (ssh)
- Test applications with a heavy client
- Test web applications
- Run actions on an Android mobile

The test consists of a sequence of actions to perform. The wizard automatically generates a `test unit` or `test suite`. An existing test (script) can be updated from the wizard too.

To add an action in the assistant, you have to

- select the action to perform
- configure it
- save the action

The wizard natively supports the use of the cache. It is therefore possible save or retrieve values from the cache.

Value: CACHE ⬍ username
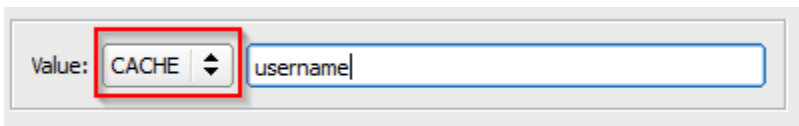
Note: Il est possible de mélanger les différents types d'actions.

Important: The wizard allows to generate tests in automatic mode but it is also possible to add its own code inside with the USERCODE action.

11.1 Framework Tabulation

The framework tab allows you to use the basic functions of the test framework.

Example of a test done with the assistant:

1. Display the message "hello" in the test
2. Ask the user during the execution his first name and save it in the cache with the firstname key
3. Display the first name in the test log
4. Check from the cache if the first name contains a specific value.
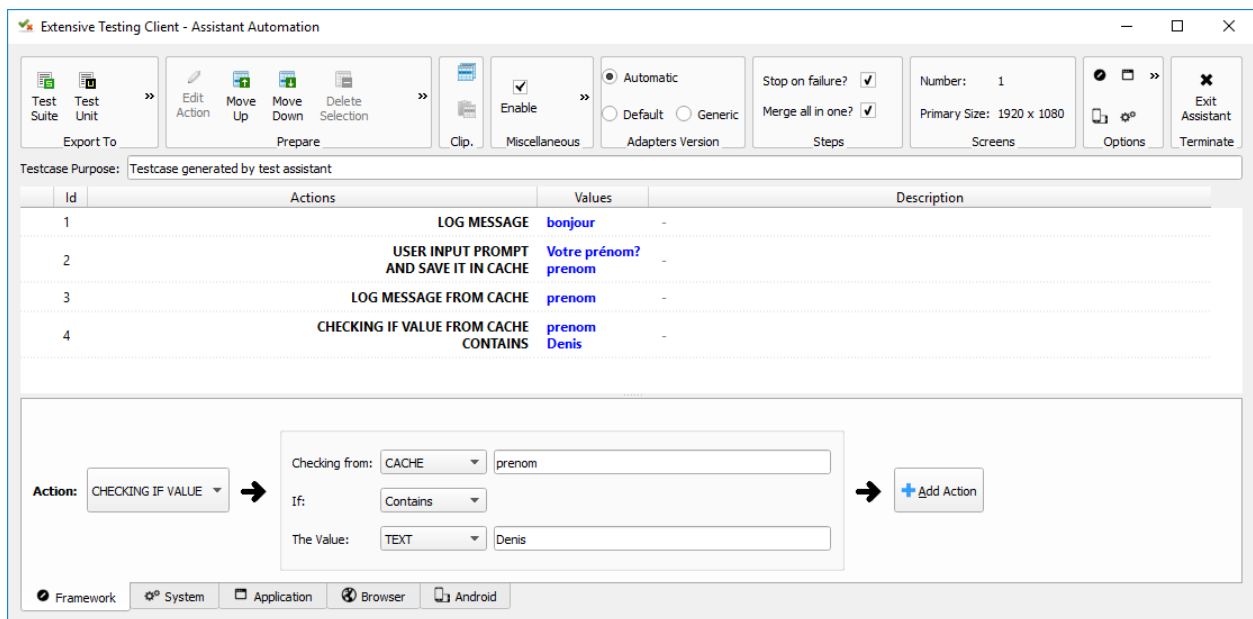
List of available actions:

Note: In red, the essential actions.

LOG MESSAGE	Displays an informational message during test execution
LOG WARNING	Display a warning message during test execution
SET VALUE	Saves a data in the cache
RESET CACHE	Blank the cache completely
USERCODE	Add custom code in the test
WAIT DURING	Wait for xx seconds
CHECK IF VALUE	Check if the value contains a specific text
ASK SOMETHING	Request a value to the user (interaction mode)

11.2 System tab

The system tab allows you to execute commands on a remote server available via SSH.

Example of a test done with the assistant:

1. Opening the ssh session on the remote machine 192.186.1.251

2. Sending the text *su -*

3. Waits to detect the text *Password:* on the screen

4. Ask the user for the root password and store it in the cache with the *pwd* key

5. Send the root password using the value stored in the cache

6. Waiting to detect on the screen the connection prompt

7. Close the SSH connection.
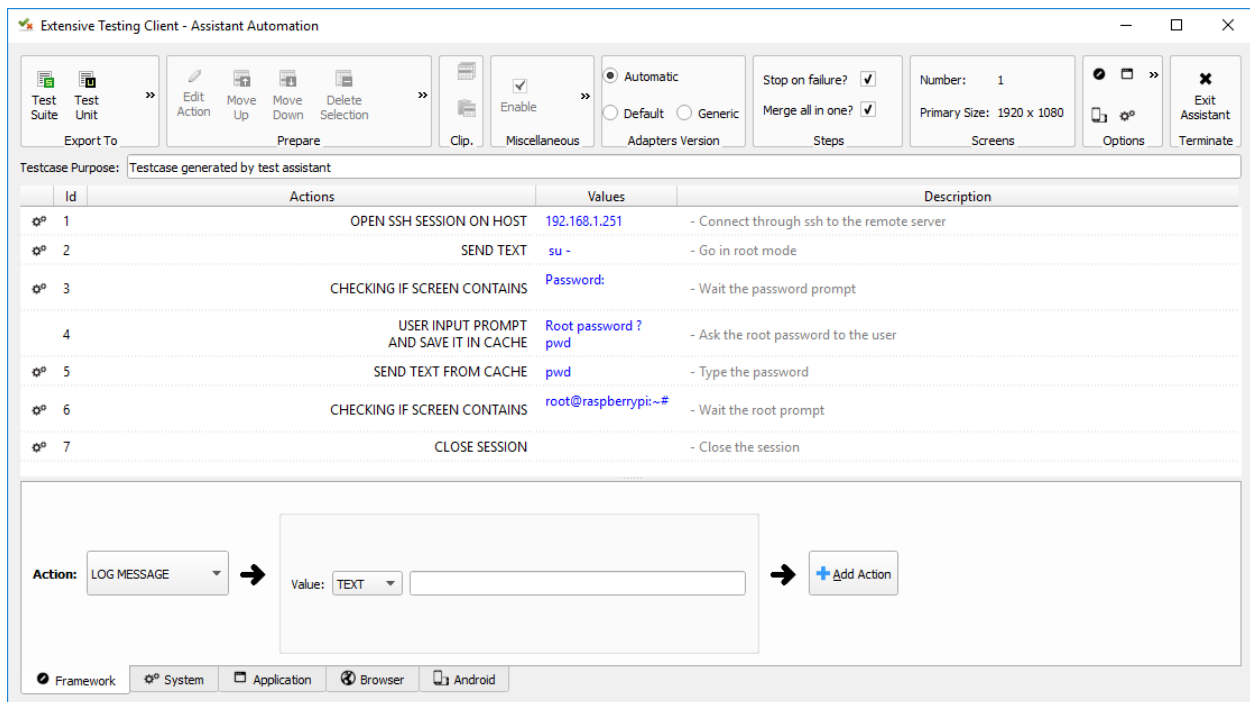
List of available actions:

Note: In red, the essential actions.

OPEN SSH SESSION	Open an SSH session
CLOSE SESSION	Close the session
CLEAR SCREEN	Blank screen
SEND TEXT	Send a string of characters
SEND SHORTCUT	Sending a keyboard shortcut (to interrupt an action)
CHECKING IF SCREEN	Check if the screen contains specific text

Note: Using the OPEN SSH SESSION action is mandatory before you can use the others available.

11.3 Tabulation application

The application tab allows you to automate rich applications by allowing:

- to simulate the keyboard
- to simulate the mouse
- search for graphic elements on the screen
- to search for text

> **Warning:** an agent sikulix-server is needed to use the actions.

Example of a test done with the assistant:

1. Send the keyboard shortcut *Win + R* to open the run window
2. Write the text *cmd*
3. Send the *Enter* keyboard shortcut to open a cmd window.
4. Waiting to detect the icon of the cmd window
5. Write the text *cls & ver* to display the version of Windows
6. Send the *Enter* keyboard shortcut to validate
7. Send the keyboard shortcut *Ctrl + A* to select the text in the window
8. Send the keyboard shortcut *Ctrl + C* to copy the selected text to the clipboard
9. Get the text from the clipboard and save it in the cache
10. Displays the text copied from the cache
11. Write the *exit* text in the cmd window
12. Send the *Enter* keyboard shortcut to close the window.
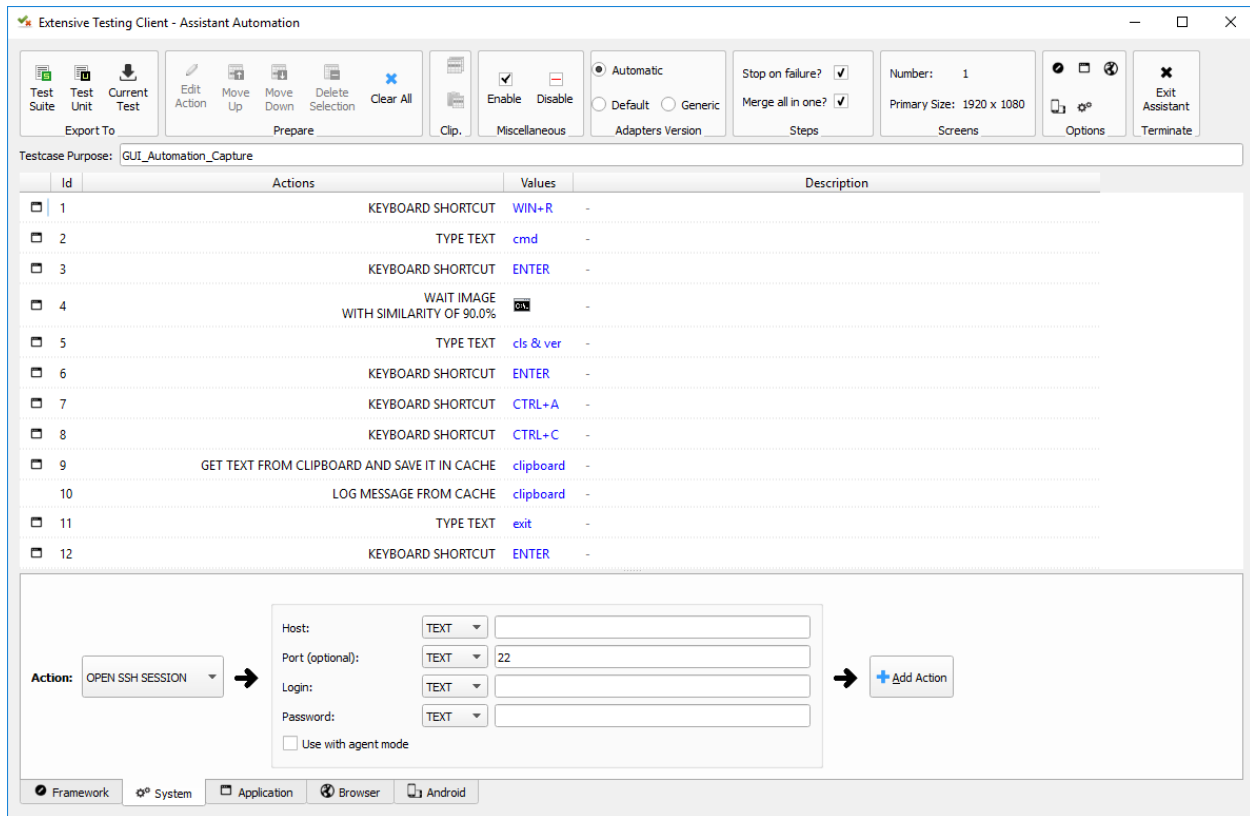
List of available actions:

Note: In red, the essential actions.

Mouse control

CLICK ON POSITION	Click on the position (x, y)
DOUBLE CLICK ON POSITION	Double click on the position (x, y)
RIGHT CLICK ON POSITION	Right click on the position (x, y)
MOUSE WHEEL DOWN	Turn the mouse wheel down
MOUSE WHEEL UP	Turn the mouse wheel up
MOVE TO POSITION	Move the cursor to the position (x, y)

Keyboard control

TYPE TEXT	Writes text
TYPE PATH	Writes text (to use for paths)
TYPE PASSWORD	Writes text (to be used to type a password)
GET TEXT FROM CLIPBOARD	Retrieves the text present in the clipboard
KEYBOARD SHORTCUT	Allows you to type a keyboard shortcut

String control

CLICK ON WORD	Search a word on the screen and click on it
DOUBLE CLICK ON WORD	Search for a word on the screen and double-click on it
RIGHT CLICK ON WORD	Search for a word on the screen and right-click on it
WAIT WORD	Search a word until it appears
WAIT AND CLICK ON WORD	Search a word until it appears and click on it

Image Control

CLICK ON IMAGE	Search an image and click on it
DOUBLE CLICK ON IMAGE	Search an image and double-click on it
RIGHT CLICK ON IMAGE	Search an image and right-click on it
WAIT IMAGE	Search an image until you see it on the screen
WAIT AND CLICK ON IMAGE	Search an image until you see it on the screen and click on it
HOVER MOUSE ON	Find an image and move the mouse cursor over it
DRAG IMAGE AND DROP TO	Find an image and drag and drop to position (x, y)

11.4 Browser Tabulation

The browser tab allows you to automate web applications by allowing:

- to control browsers (firefox, internet explorer, chrome, edge)

- to simulate the keyboard

Warning: an agent `selenium3-server` or `selenium2-server` is needed to use the actions.

Tip: To click on an HTML element, it is advisable to use systematically the `WAIT VISIBLE AND CLICK ON HTML ELEMENT` function.

Example of a test done with the assistant:

1. Get the name from the cache and send it to the HTML element found by the xpath

2. Click on the HTML element found by the xpath

3. Find the HTML element found by the xpath and click on it as soon as it is visible on the screen.
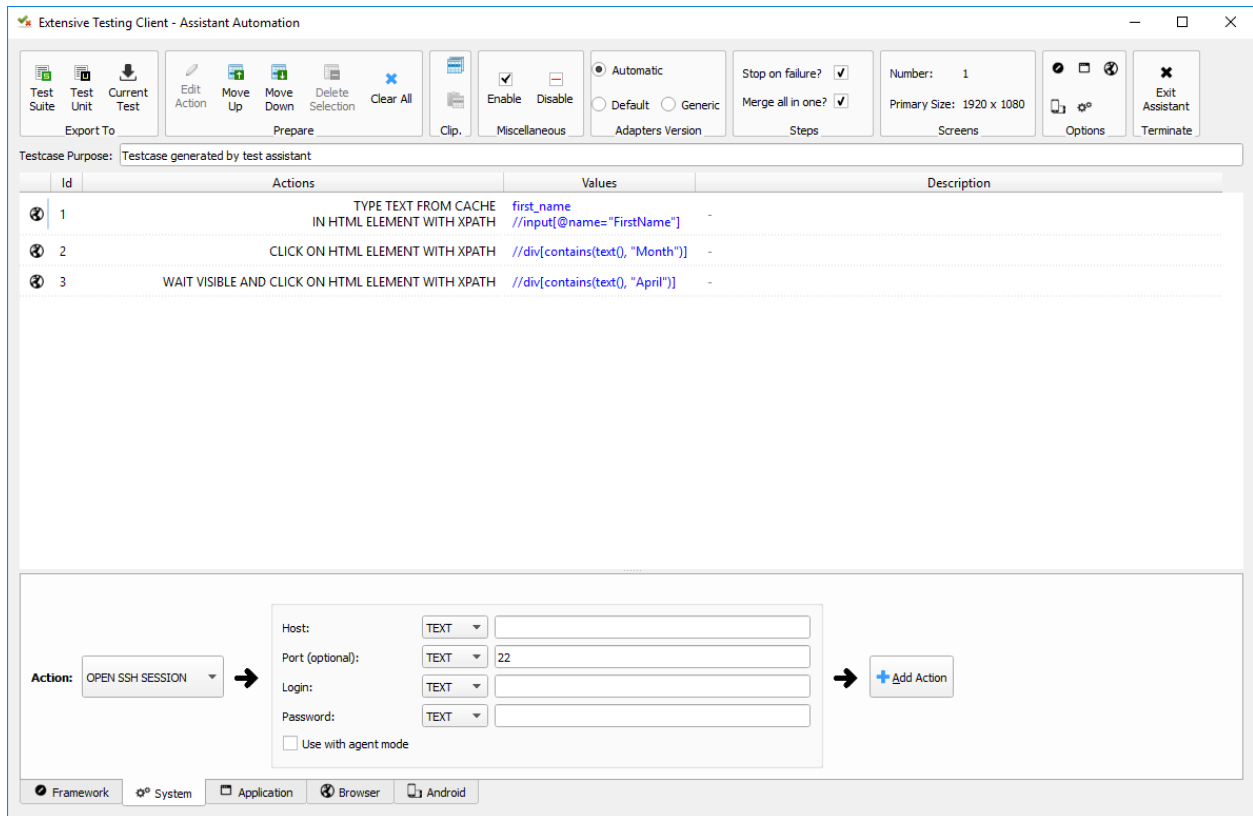
Note: It is possible to open multiple browsers in parallel on the same extension to define a new session. The name of the session is defined by the OPEN BROWSER action. Then use the SWITCH TO SESSION action to switch sessions.

Available actions:

Note: In red, the essential actions.

Browser Control

OPEN BROWSER	Open the browser and load the specified url
CLOSE BROWSER	Closes the browser
MAXIMIZE BROWSER	Enlarges the browser window

Navigation actions

REFRESH PAGE	Refresh the page
GO BACK	Backspace
GO FORWARD	Go forward
ACCEPT ALERT	Validate the javascript alert
DISMISS ALERT	Dismiss the javascript alert
CLOSE CURRENT WINDOW	Closes the current window
SWITCH TO NEXT WINDOW	Toggle on next window
SWITCH TO FRAME	Toggle on the next frame
SWITCH TO SESSION	Toggles to another selenium session
SWITCH TO WINDOW	Toggle on the next frame

javascript actions

EXECUTE JAVASCRIPT ON HTML ELEMENT	Allows you to inject javascript script on an html element

Actions on html elements

WAIT HTML ELEMENT	Wait for the appearance of a precise HTML element
WAIT AND CLICK ON HTML ELEMENT	Wait for the appearance of a precise HTML element and click on it
WAIT VISIBLE HTML ELEMENT	Wait for an HTML element to be visible to the user
WAIT NOT VISIBLE HTML ELEMENT	Wait until an HTML element is not visible to the user
WAIT VISIBLE AND CLICK ON HTML ELEMENT	Wait for an HTML element to be visible to the user and click on it
HOVER ON HTML ELEMENT	Move the mouse cursor over a specific HTML element
CLICK ON HTML ELEMENT	Click on a specific HTML element
DOUBLE CLICK ON HTML ELEMENT	Double click on a specific HTML element
CLEAR TEXT ON HTML ELEMENT	Empty the text on a specific HTML element
SELECT ITEM BY TEXT	Select item according to the text (for combolist or list)
SELECT ITEM BY VALUE	Select item according to the value attribute (for combolist or list)

Text Recovery

GET TEXT ALERT	Retrieves the text of an alert message javascript
GET TEXT FROM HTML ELEMENT	Retrieves the text an exact html element
GET PAGE TITLE	Retrieves the title of the page
GET PAGE URL	Get the URL of the page
GET PAGE SOURCE CODE	Get the source code page

Keyboard simulation

TYPE KEYBOARD SHORTCUT	Sends a keyboard shortcut to a specific HTML element
TYPE TEXT ON HTML ELEMENT	Sends text on a specific HTML element

11.5 Android Tabulation

The android tab allows you to automate mobile applications by enabling:

- to simulate the keyboard
- to simulate the use of the fingers on the screen
- to control the system and the applications

Warning: an adb agent is needed to use the actions.

Overview of the agent
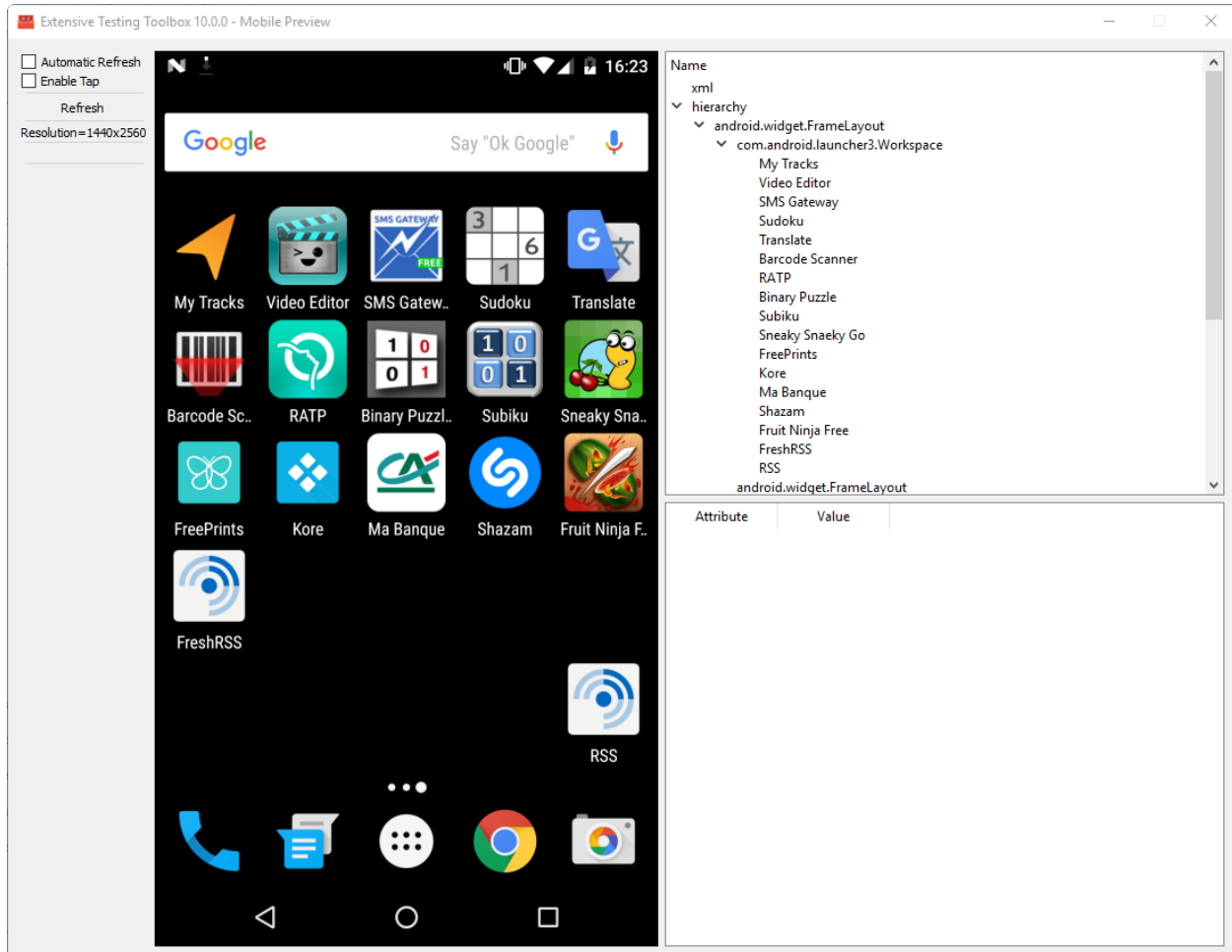
Example of a test done with the assistant:

1. Wake up the device
2. Unlock the device
3. Click on the *HOME* button
4. Stop the application
5. Click on the 'Play Store' app to open it
6. Wait for the application to open and search the *APPS & GAMES* menu
7. Click on the text *ENTERTAINMENT*
8. Click on the menu *MOVIES & TV*

---

9. Wait for 5 seconds

10. Research the image

11. Put the device to sleep.
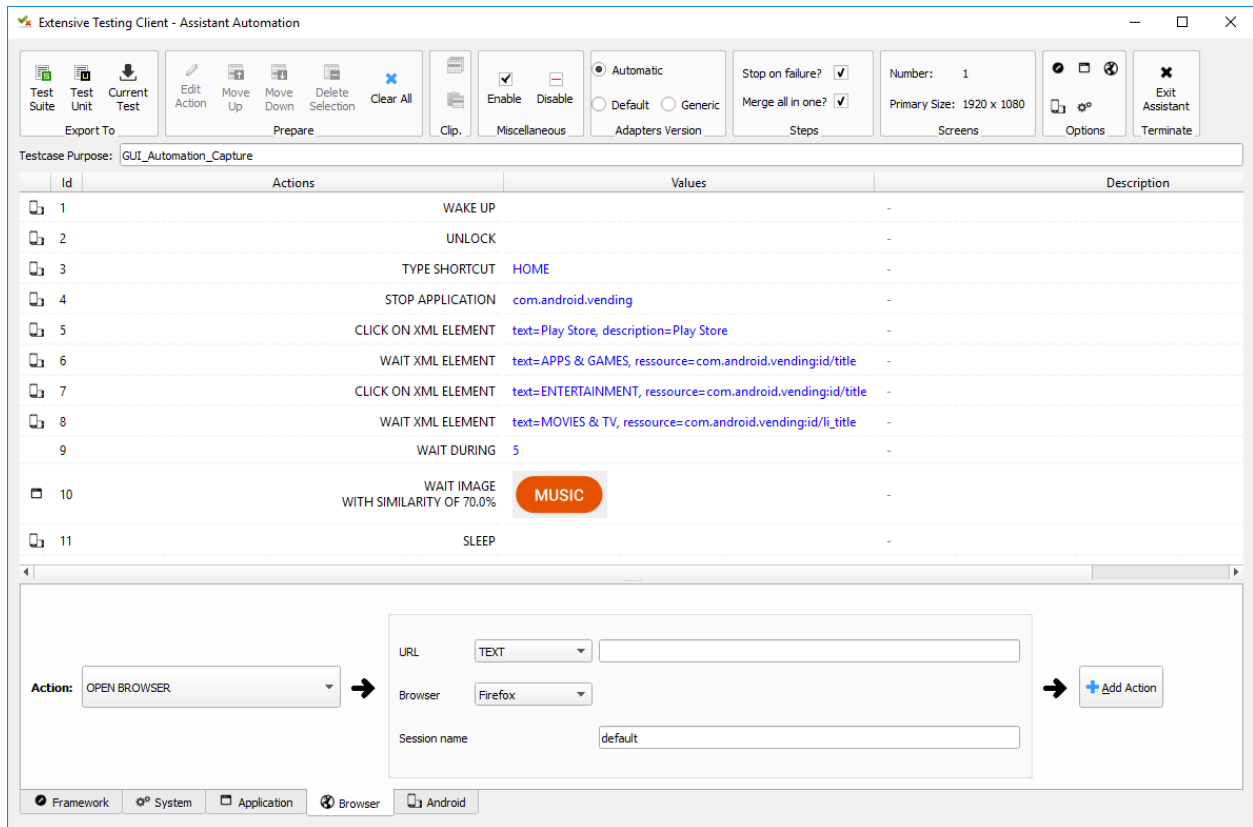
Available actions:

Note: In red, mandatory actions.

Mobile controls

WAKE UP AND UNLOCK	Wake up and unlock the device
REBOOT	Restarting the device
SLEEP	Paused

Texts

TYPE SHORTCUT	Simulates a shortcut
TYPE TEXT ON XML ELEMENT	Sends text on a specific element of the interface
GET TEXT FROM XML ELEMENT	Retrieves the text of a specific element of the interface

Contrôles des éléments XML

CLEAR XML ELEMENT	Removes text from a specific element of the interface
CLICK ON XML ELEMENT	Click on a specific element of the interface
LONG CLICK ON XML ELE-MENT	Long-term click on a specific element of the interface
WAIT AND CLICK ON XML ELEMENT	Wait for the appearance of a specific element of the interface and click on it

Tap on screen

CLICK TO POSITION	Click on the position x, y
DRAG FROM POSITION	Drag from position x1, y1 to x2, y2
SWIPE FROM POSITION	Swipe from position x1, y1 to x2, y2

Troubleshooting

12.1 Errors codes

Framework error code

Error code	Description
ERR_TE_000	Generic error during test execution
ERR_TE_001	Generic error during testcase execution
ERR_TE_500	Generic error during script execution

Steps errors

Error code	Description
ERR_STP_001	The step is already started, the function *start()* is called several time in the test.
ERR_STP_005	The test try to set the result bu the step is not started

Test properties errors

Error code	Description
ERR_PRO_004	The parameter name for *input* function does not exits

12.2 How to fix

12.2.1 Unable to generate the test design of a test

The generation of the description of a test does not works in some cases:

- bad version for adapters and libraries
- syntax error in the test

- the cache is used in the step definition

Installation

13.1 Server

13.1.1 Automatic mode

> **Warning:**
>
> **Basic configuration to respect before starting the installation:**
>
> - the network interface is correctly configured on the server
>
> - access to official repositories is available
>
> - using a Linux CentOS 7 or equivalent RedHat system

Installation of the solution can be done using the `install.sh` script in the tar.gz. If the prerequisites are respected then the installation will be done in automatic mode, that is to say that the missing packets will be recovered automatically.

Example of installation in automatic mode

```
./install.sh
Are you sure to install the product? (yes or no) yes
========================================================
=  - Installation of the ExtensiveAutomation product -  =
=                 Denis Machard                         =
=              www.extensiveautomation.org              =
========================================================
* Detecting the operating system (centos 7)          [  OK  ]
* Detecting the system architecture (x86_64)         [  OK  ]
* Detecting Perl, Python                              [  OK  ]
* Detecting primary network address (XXX.XXX.XXX.XXX)[  OK  ]
* Adding external libraries .................        [  OK  ]
```

(continues on next page)

```
* Adding external libraries .......                    [  OK  ]
* Adding interop libraries .......                     [  OK  ]
* Detecting Apache                                     [  OK  ]
* Detecting MySQL/MariaDB                              [  OK  ]
* Detecting Postfix                                    [  OK  ]
* Detecting Openssl                                    [  OK  ]
* Detecting Php                                        [  OK  ]
* Copying source files                                 [  OK  ]
* Adding startup service                               [  OK  ]
* Updating configuration files                         [  OK  ]
* Creating extensivetesting user                       [  OK  ]
* Updating folders rights                              [  OK  ]
* Updating php configuration                           [  OK  ]
* Updating httpd configuration                         [  OK  ]
* Adding virtual host                                  [  OK  ]
* Restarting httpd                                     [  OK  ]
* Restarting MySQL/MariaDB                             [  OK  ]
* Restarting postfix                                   [  OK  ]
* Adding the ExtensiveAutomation database              [  OK  ]
* Starting ExtensiveAutomation X.X.X                   [  OK  ]
======================================================================
- Installation completed successfully!
- Continue and go to the web interface (https://XXX.XXX.XXX.XXX/web/index.php)
======================================================================
```

Example to check if the server is working properly.

```
xtctl status
Extensive Testing is running
```

The server is accessible at the address indicated at the end of the installation. It is possible to use no fault accounts to connect:

- user admin

- user test

- user monitor

Note: The default accounts do not have a password.

Warning: Do not forget to change the default account passwords or disable the accounts.

13.1.2 Custom mode

Warning: This installation mode is only recommended for advanced users.

This mode is used to change the destination of the installation or to specify certain parameters (see the example).

Example of installation in custom mode (here, the destination of xtc)

```
./custom.sh
=======================================================
=  - Installation of the ExtensiveAutomation product -  =
=                 Denis Machard                 =
=             www.extensiveautomation.org              =
=======================================================
* Detecting the operating system (XXXXXXXX)           [  OK  ]
* Detecting the system architecture (XXXXXX)          [  OK  ]
* Detecting Perl, Python                              [  OK  ]
* Detecting primary network address (XX.XX.XX.XX)     [  OK  ]
* Download automatically all missing packages? [Yes]
* In which directory do you want to install the ExtensiveTesting product? [/opt/xtc/] <INSTALL_PATH>
* What is the directory that contains the init scripts? [/etc/init.d/]
* What is the external ip of your server? <IP_EXTERNE>
* What is the FQDN associated to the external ip of your server? <FQDN>
* What is the database name? [xtcXXX]
* What is the table prefix? [xtc]
* What is the ip of your mysql/mariadb server? [127.0.0.1] <IP_BASE>
* What is the login to connect to your mysql/mariadb server? [root] <LOGIN_BASE>
* What is the password of previous user to connect to your mysql/mariadb server? [] <MOTDEPASSE_BASE>
* What is the sock file of your mysql/mariadb server? [/var/lib/mysql/mysql.sock]
* Do you want to configure iptables automatically? [Yes]?
* Do you want to configure php automatically? [Yes]?
* Where is your php conf file? [/etc/php.ini]
* Do you want to configure apache automatically? [Yes]?
* What is the directory that contains the httpd conf file? [/etc/httpd/conf/]
* What is the directory that contains the httpd virtual host conf files? [/etc/httpd/conf.d/]
* What is the directory that contains the virtual host? [/var/www/]
* Do you want to configure selinux automatically? [No]?
* What is the path of the openssl binary? [/usr/bin/openssl]
```

Example to check if the server is working properly.

```
xtctl status
Extensive Testing is running
```

The server is accessible at the address indicated at the end of the installation. It is possible to use the default accounts to log in:

- user admin

- user test

- user monitor

Note: The default accounts do not have a password.

Warning: Do not forget to change the default account passwords or disable the accounts.

13.1.3 From sources

> **Warning:** This installation mode is only recommended for advanced users.

First, you need to install the system and python libraries necessary for the proper functioning of the program, the complete list is available in the contribution chapter.

After retrieving the sources from github you have to recompose the different directories

core-server	/opt/xtc/vXXX
plugins-adapters	</opt..>/SutAdapters/vXXX
plugins-libraries	</opt..>/SutLibraries/vXXX
test-interop	</opt..>/TestInterop/
test-library	</opt..>/TestExecutorLib/
web-client	</opt..>/Web

Preparation of the database

To initialize the database, execute the "add-bdd.py" scripts available in the `/Scripts/database` directory.

Starting the server

Run the `python run` scripts available at the root of the `/opt/xtc/vXXX/` directory.

13.1.4 Update

The server update is possible by running the script `./Update.sh` Old tests, adapters and users are automatically migrated.

> **Note:** The update is refused if no version of the product is detected.

13.1.5 Rollback

Backtracking is possible to earlier versions already installed on the server. Run the `rollback.sh` script with the previous version.

```
./rollback.sh X.X.X
==================================================
=  - Rollback of the ExtensiveAutomation product -  =
=              Denis Machard                =
=          www.extensiveautomation.org          =
==================================================
* Detecting the operating system                    [  OK  ]
* Detecting the system architecture                 [  OK  ]
* Stopping the ExtensiveAutomation server           [  OK  ]
* Rollbacking to ExtensiveAutomation-X.X.X          [  OK  ]
* Restarting the ExtensiveAutomation server         [  OK  ]
======================================================================
- Rollback completed successfully!
======================================================================
```

13.1.6 Uninstall

Uninstalling the product can be done using the `./Uninstall.sh` script in the installation package.

```
./uninstall.sh
====================================================
=  - Uninstall of the ExtensiveAutomation product -  =
=              Denis Machard                  =
=           www.extensiveautomation.org          =
====================================================
* Detecting the operating system                   [  OK  ]
* Detecting the system architecture                [  OK  ]
* Stopping the ExtensiveAutomation server          [  OK  ]
* Stopping httpd                                   [  OK  ]
* Removing the ExtensiveAutomation database        [  OK  ]
* Removing the ExtensiveAutomation source          [  OK  ]
* Removing the ExtensiveAutomation service         [  OK  ]
* Removing ExtensiveAutomation user                [  OK  ]
* Restoring php                                    [  OK  ]
* Removing httpd configuration                     [  OK  ]
* Restarting httpd                                 [  OK  ]
=========================================================================
- Uninstallation completed successfully!
=========================================================================
```

Note: It is possible to use the `force` mode if an error occurs during the uninstallation.

13.2 Client

13.2.1 Windows Installation

There are 2 installation modes:

- portable mode (recommended version)
- installation mode

The client can be retrieved from the website https://www.extensiveautomation.org or from the test server.
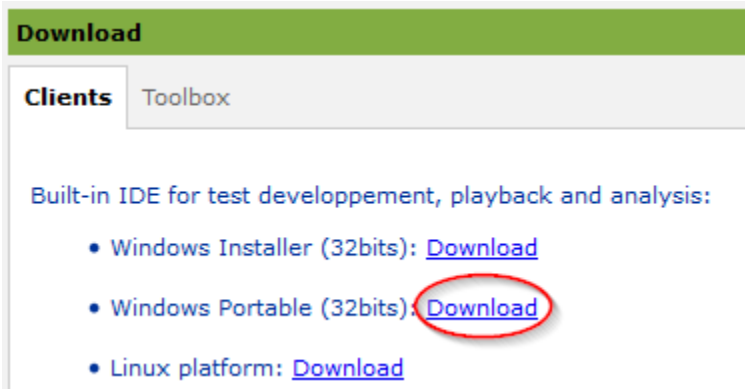
Then you have to unzip it and run the *ExtensiveAutomationClient.exe* file

Files	24/08/2016 13:56
imageformats	24/08/2016 13:56
Logs	24/08/2016 13:56
Plugins	24/08/2016 13:56
ResultLogs	24/08/2016 13:56
Tmp	24/08/2016 13:56
Update	24/08/2016 13:56
ExtensiveTestingClient.exe	24/08/2016 13:56
ExtensiveTestingClient.ico	08/07/2016 19:20
HISTORY	08/07/2016 19:20

13.2.2 Linux Installation

There is no pre-compiled version for Linux. It is necessary to recover the sources since github, to install the missing packets and to execute the following file

```
python Main.py
```

13.2.3 Update

The update of the client is possible in automatic mode (if present on the server) or manual. From the client it is possible to verify the presence of an update.

A newer minor of this client is available.
You may click on download button and install the update.

Version 10.2.0

Download   Cancel
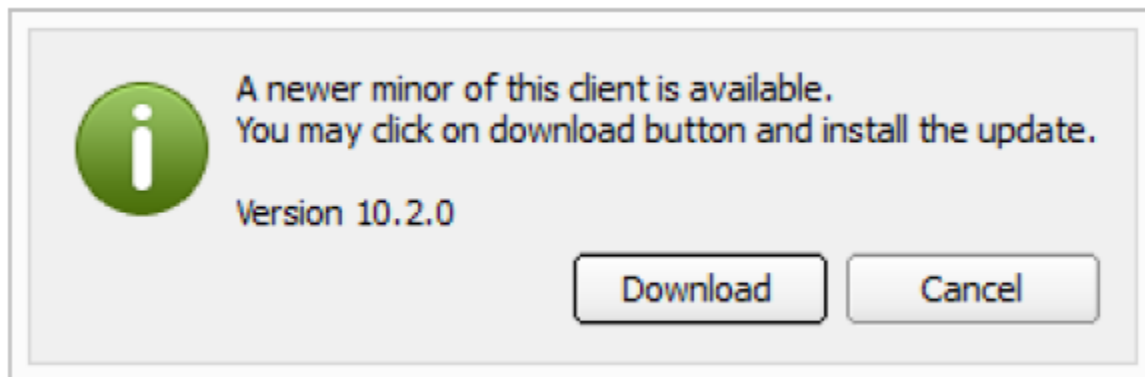
Note: If the proposed version is a major release then the update is mandatory.

13.3 Toolbox

13.3.1 Windows Installation

There are 2 installation modes:

- portable mode (recommended version)

- installation mode

The toolbox can be retrieved from the website https://www.extensiveautomation.org or from the test server.
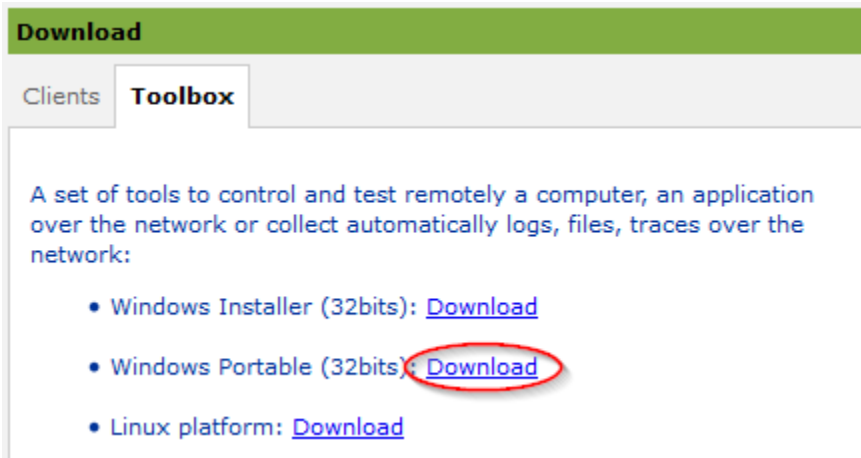
Then you have to decompress it and run the ExtensiveAutomationToolbox.exe file

13.3.2 Linux Installation

The toolbox can be retrieved from the website https://www.extensiveautomation.org or from the test server.
2 scripts are available to start an agent or probe.

- ./toolagent

- ./toolprobe

```
./toolagent
Command line tool launcher

Usage: ./toolagent [test-server-ip] [test-server-port] [ssl-support] [ftp|sikulix|socket|dummy|
database|selenium|gateway-sms|command|soapui|file|adb|ssh] [tool-name]
[tool-description] [[proxy-ip] [proxy-port]]

* Server parameters
[test-server-ip]: your test server ip or hostname. This option is mandatory.
[test-server-port]: your test server port. This option is mandatory.
[ssl-support=True/False]: ssl support. This option is mandatory.

* Tools parameters
[Values expected: ftp|sikulix|socket|dummy|database|selenium|gateway-sms|
```

(continues on next page)

```
command|soapui|file|adb|ssh]: tool type to start. This option is mandatory.
[tool-name]: The tool name. This option is mandatory.
[tool-description]: The tool description. This option is mandatory.

* Proxy parameters
[proxy-ip]: proxy address. This option is optional.
[proxy-port]: proxy port. This option is optional.
```

```
./toolprobe
Command line tool launcher

Usage: ./toolprobe [test-server-ip] [test-server-port] [ssl-support] [dummy|textual|network|
file] [tool-name] [tool-description] [[proxy-ip] [proxy-port]]

* Server parameters
[test-server-ip]: your test server ip or hostname. This option is mandatory.
[test-server-port]: your test server port. This option is mandatory.
[ssl-support=True/False]: ssl support. This option is mandatory.

* Tools parameters
[Values expected: dummy|textual|network|file]: tool type to start. This option is mandatory.
[tool-name]: The tool name. This option is mandatory.
[tool-description]: The tool description. This option is mandatory.

* Proxy parameters
[proxy-ip]: proxy address. This option is optional.
[proxy-port]: proxy port. This option is optional.
```

13.3.3 Update

The update of the toolbox is to be done manually. You have to get the package from the website or from the test server.

The update requires:

- delete the current version
- add the new version and reconfigure the agents or probes to restart.

Note: The automatic update is not supported yet.

CHAPTER 14

Administration

14.1 Start/Stop of the server

The server can be controled with the following command `xtctl`. This command enables to

- start or stop the server
- check the status
- deploy a new graphical client or toolbox
- display the version

Use the following command to start the server `xtctl start`.

```
# xtctl start
Checking database                                [  OK  ]
Saving current adapters                          [  OK  ]
Saving current libraries                         [  OK  ]
Starting Extensive Automation                    [  OK  ]
```

Use the following command to stop the server `xtctl stop`.

```
# xtctl stop
Saving current adapters                          [  OK  ]
Saving current libraries                         [  OK  ]
Stopping Extensive Automation                    [  OK  ]
```

Tip:

More details in logs about the start or stop procedure.

```
# tailf /opt/xtc/current/Var/Log/output.log
2014-12-06 11:00:54,092 - INFO - Extensive Automation successfully started (in 14 sec.)
...
```

```
2014-12-06 10:58:51,810 - INFO - Stopping server
2014-12-06 10:58:51,911 - INFO - Extensive Automation successfully stopped!
```

14.2 Server status's

xtctl enable to check the status of the server, 3 states exists:

- `starting`: the server is starting
- `running`: the server is running properly
- `stopped`: the server is stopped

Tip: Don't forget to check the status of `httpd` and `mysql` services.

14.3 New packages deployment

The solutions make available all additionnals package needed for users:

- The graphical client
- The toolbox
- Plugins

When a new client is available, it's possible to put it in the server and notify all users of this new package.

Packages must be uploaded in the following folder `<INSTALL_PATH>/current/Packages/`

Client	Portable version and installation
ClientPlugins	Plugins
Toolbox	Portable version and installation
ToolboxPlugins	Plugins

Packages are automatically available from the web interface. It's possible to execute the command `xtctl` `deploy` in the server to make it available to all.

```
./xtctl deploy
Deploying clients.(ExtensiveAutomationClient_X.X.X_Setup.exe)
Deploying tools.(ExtensiveAutomationToolbox_X.X.X_Setup.exe)
Deploying portable clients... (No client)
Deploying portable tools... (No client)
```

14.4 Server settings

The file `settings.ini` contains all parameters to configure the server. Parameters are separated in several sections:

- Boot

- Notifications

- Client_Channel

- Agent_Channel

- Probe_Channel

- WebServices

- TaskManager

- Network

- Paths

- Bin

- Server

- Web

- Bind

- Misc

- MySql

- Trace

- Backups

- Default

- Csv_Test_Results:

- Tests_Framework

- Events_Colors

- Supervision

- Users_Session

14.5 Automatic backups

The solution make a backup of all tests, adapters and libraries every days. Backups are stored in the folder /opt/xtc/current/Var/Backups.

The interval of backup can be configured from the section Backups in the file settings.ini.

```
[Backups]
; tests repository
; 0=disable 1=enable
tests=1
; backup zip name
tests-name=tests-automatic-backup
; backup weekly on sunday at 23:40:00
tests-at=6|23,40,00
```

Scheduler type:

- 7: weekly

- 6: daily

- 5: hourly

14.6 Crontab scripts

`cron.backup-tables`: this script allows to save the tables of the solution

`cron.cleanup-backups`: this script allows you to delete backups older than 14 days. The number of days is configurable.

`cron.cleanup-testsresult`: this script allows you to delete results older than 30 days. The number of days is configurable.

14.7 Security banner

It is possible to configure a security banner on the web interface of the server and on the portable client login window.

For this you have to configure the file `BANNER` present in

- in the `/opt/xtc/current/Web/` web directory for the server
- the connection of the execution file for the graphical client.

Projects

The solution is multi-project. It is therefore possible to organize the tests by projects and grant access rights for users.
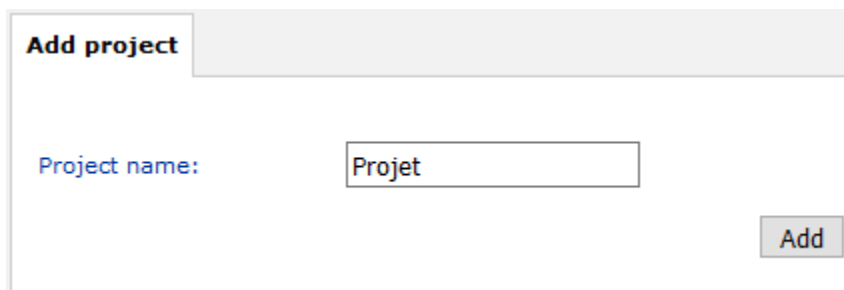
Note: The `Common` project exists by default, it is accessible by all users and can not be deleted.

15.1 Add a project

Only an administrator can add a new project. Creating a project requires specifying its name and can be done via the web interface or the API

15.2 Delete a project

Only an administrator can remove a project. This action can be done throught the web interface or the web api.
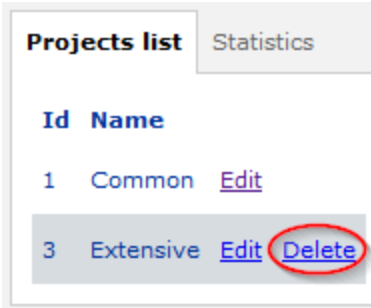
Note: If the project is associated with a user, deletion is not allowed.

15.3 Link a project to a user

A user can access to several projects. From the profile of a user, you can select all the projects authorized. It's possible to define the default one.

Users

The solution is multi-user, 3 users exists by default:

- Admin
- Tester
- Monitor

Note: Don't forget to disable default account in a production environment.

16.1 Add user

Only an administrator can add a new user. The creation of a user requires at least the following parameters and can be done via the web interface or the API

- username
- password

Note: Email is used by the solution to send test reports and results.

Note: It is possible to configure multiple email addresses for a user by separating them with ;

16.2 Delete a user

Only an administrator can remove a user. This action can be done throught the web interface or the web api.

CHAPTER 17

Adapters and libraries

The version of the adapters (or libraries) to use by default for a test can be configured. It is also possible to use multiple versions of adapters (or libraries) in parallel on a test.

It is therefore advisable to define:

- a default version to use
- but also a generic version

Separation in default or generic version allows to evolve your adapters (or libraries) independently open source adapters (or libraries)

Troubleshooting

18.1 Logs

18.1.1 Server

The server logs are stored on `/opt/xtc/current/Var/Logs/`. The logs are set in `INFO` mode by default. The DEBUG level can be activated from the `settings.ini` file.

Note: It is possible to change the level of logs by doing an `xtcl reload`

18.1.2 Client

The client logs are available in `<Program Files>` `/Extensive Testing Client/Logs/` The logs are set in `INFO` mode by default.

The DEBUG level can be activated from the client preferences.
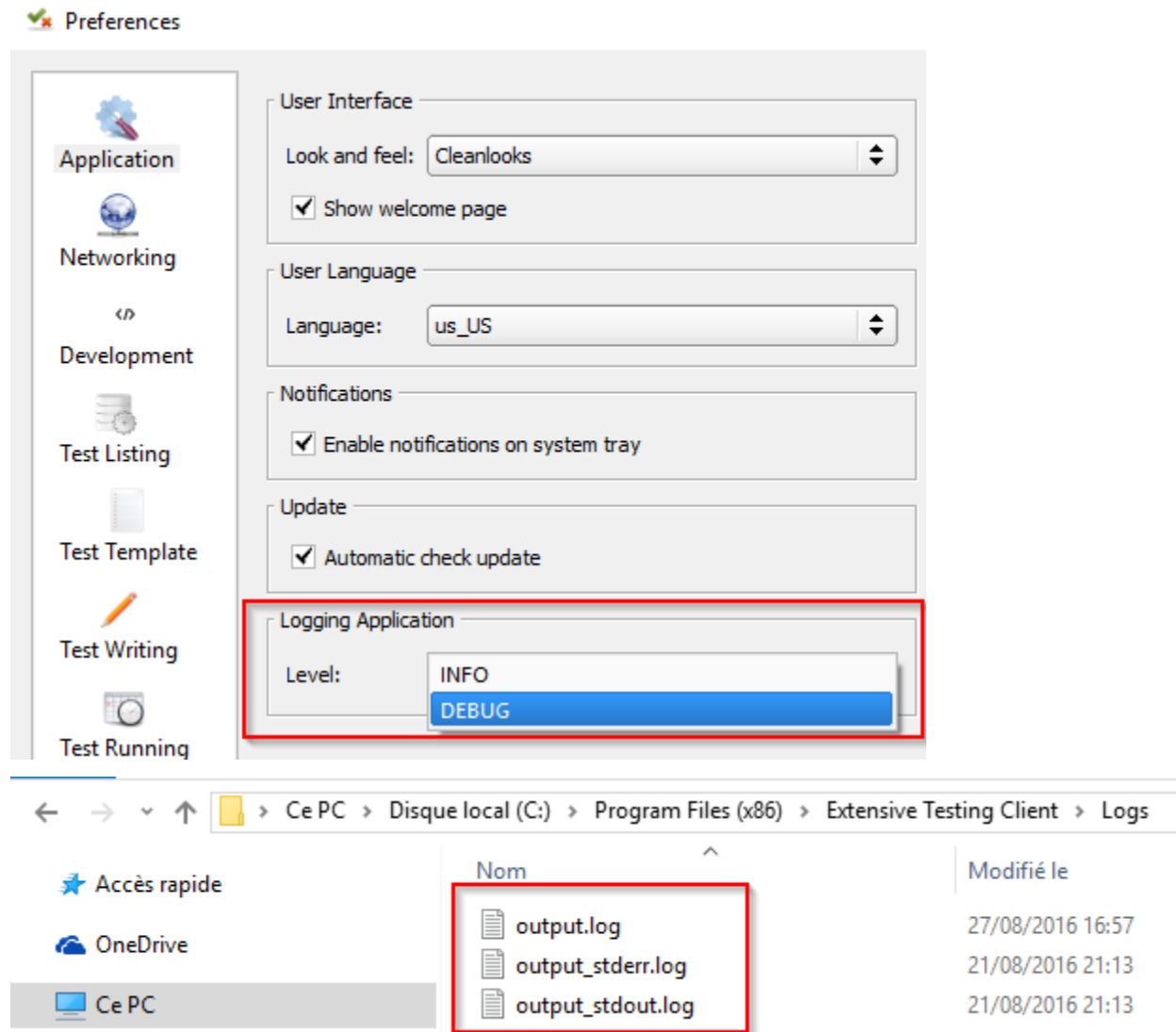
18.1.3 Toolbox

The logs in the toolbox are available in `<Program Files>/Extensive Testing Toolbox/Logs/` The logs are set in `INFO` mode by default.

The DEBUG level can be activated from the `settings.ini` file.
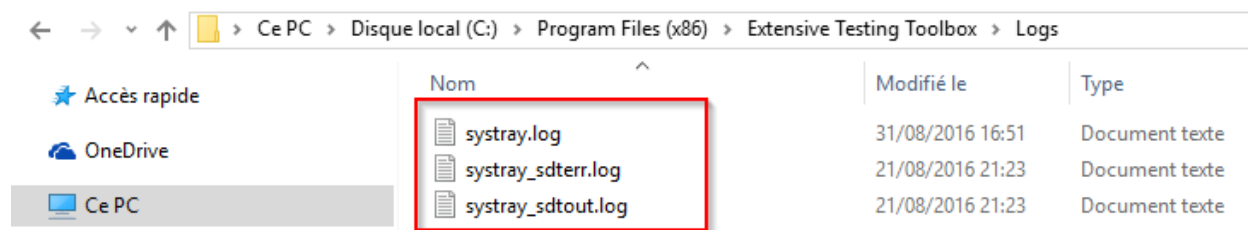
Note: A restart of the toolbox is necessary to take into account the change

18.2 Frequently Asked Questions

How to change the listening port (tcp / 443) of the server? ~~~~~~ ~~~~~

Edit the `/etc/httpd/conf.d/extensivetesting.conf` file and change the listening port of the virtual host 443. Do not forget to modify the `/etc/httpd/conf/httpd.conf` file to add the new listening port.

Note: An apache restart is needed.

18.2.1 How to change the connection port of the client?

The destination port can be changed from the client preferences. Or directly from the `settings.ini` file.
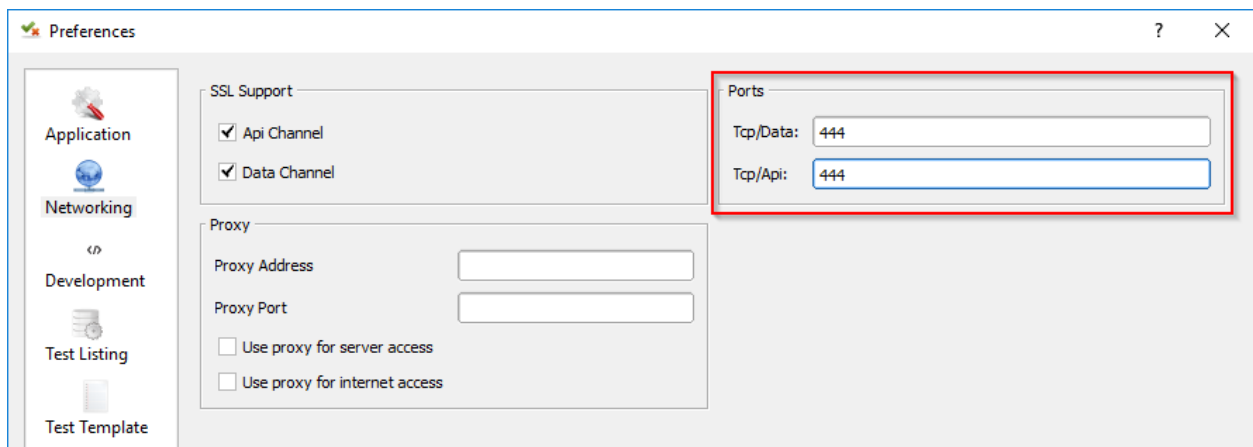
18.2.2 View the server version?

```
./xtctl version
Server version: 18.0.0
```

What to do if the server installation does not work? ~~~~~~ ~~~~~~

The server installation process is logged in an `install.log` file in the directory after extracting the package. It is necessary to look for the error messages present in the file.

What if my connection to the server does not work? ~~~~~~ ~~~~~~

If the connection from the client to the server does not work, an analysis is necessary.

The first SLR to have is to connect to the server in SSH and execute the `xtctl status` command to check if the server is running.

1. If the server is running then check:

   - network connectivity in the client and the server
   - a firewall blocking the https flow (443)

---

2. If the network connectivity is good and the server is working (or not), check the logs. The file is available in the /opt/xtc/current/Var/Logs/output.log directory. You must look for messages of type ERROR

How to fix the error "hping3 is not installed"? ~~~~~~

This error occurs while running a test when the Pinger adapter is used. Indeed requires to have the hping3 system library installed on the server.

You have to retrieve the sources from https://github.com/antirez/hping and compile them:

```
cd hping-master
yum install libpcap-devel-1.5.3-9.el7.x86_64
ln -s /usr/include/pcap/bpf.h /usr/include/net/bpf.h
./configure
make
make install
```

How to install the server in a specific directory? ~~~~~~

By default, the server installs in the directory /opt/xtc/, it is possible to change this directory at installation time by changing the INSTALL key in the default.cfg file

```
INSTALL=/opt/xtc/
```

Server installation remains stuck on adding external libraries ~~~~~~

Before starting the server installation, check that the yum service is not already running. If so, the installation script will remain blocked until yum is available. This issue Usually happens when the server is installed in graphical mode.

In the logs, we can observe the following error:

```
Existing lock /var/run/yum.pid: another copy is running as pid 3293.
Another app is currently holding the yum lock; waiting for it to exit...
  The other application is: PackageKit
    Memory :  26 M RSS (429 MB VSZ)
    Started: Tue Nov  1 11:09:25 2016 - 00:42 ago
    State  : Sleeping, pid: 3293
```

To solve this problem, you must stop the program that already uses yum.

18.2.3 Can not navigate in the web interface

If you can connect to the web interface but can not navigate the menus. The cookie generated by the server may be expired, it must be verified that the server is on time.

Tests definitions

The solution is based on different types of tests for:

- enable the construction of advanced tests
- decrease the use of script
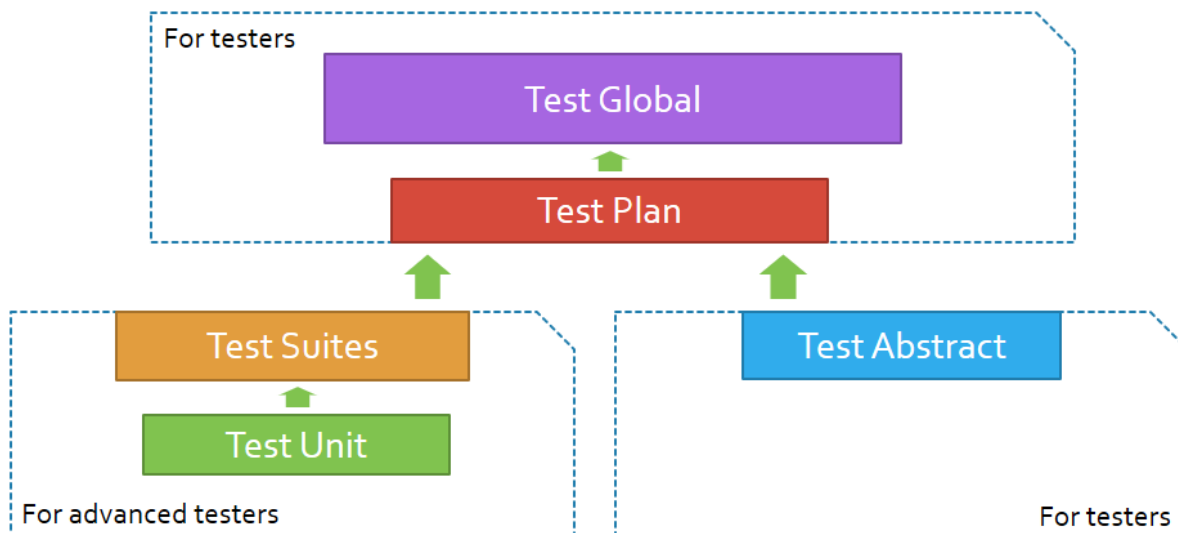
19.1 Test Abstract

The `test abstract` (tax) allows you to write a test case with several steps. This format is oriented graphic modeling so does not require any knowledge in development.

A Test Abstract
Testcase modelisation

19.2 Test Unit

The `test unit` (tux) allows you to write a test case with several steps. This format is oriented development.

U Test Unit
Testcase scripting

19.3 Test Suite

The `test suite` (tsx) allows you to write several test cases with several steps. This format is oriented development.

S Test Suite
Multiple testcases scripting

19.4 Test Plan

The `plan test` (tpx) allows you to write test cases. The design is realized by nesting the tests *abstract*, *unit* and *suite* This test format requires no knowledge in development.

P Test Plan
Procedural test conception

19.5 Test Global

The `global test` (tgx) allows you to write test campaigns. The preparation of the campaigns is carried out by importing the tests *plans*.

G Test Global
End to end test conception

Note: It is also possible to import other types of tests.

The fundamentals

The test framework provides a framework for standardizing the creation of test cases.

The main features are:

- the support of test cases with steps
- Support extensions to communicate with the system to test or control
- automatic generation of test reports.

20.1 Test case

The creation of a test case in the solution is standardized.

A test case is divided into 4 sections:

- `description`: description of the different stages of the test
- `prepare`: preparation of adapters and libraries for communicating with the system to be tested or piloted
- `definition`: test flow
- `cleanup`: cleaning phase

The result of a test case is automatically calculated by the framework when the test is completed according to the different stages defined.

There are 3 possible results:

- `PASS`: all the steps of the tests have been successfully executed
- `FAILED`: at least one step is in error after execution
- `UNDEFINED`: at least one step of the test has not been executed

Note: The `cleanup` section is always called, even if there is an error.

20.2 Test steps

A test case is divided into sub-steps.

A step is defined by:

- a summary of the action to be carried out
- the detailed description of the action to be carried out
- the description of the expected result to validate the step.

The definition of the test steps must be done in the `description` section:

```
self.step1 = self.addStep(
                                expected="Logged in",
                                description="Login throught the rest api",
                                summary="Login throught the rest api",
                                enabled=True
                        )
```

The result of a step is to be specified in the `definition` section

Example to set the result to PASS or FAILED

```
self.step1.setPassed(actual="step executed as expected")

self.step1.setFailed(actual="error to run the step")
```

Warning: Do not forget to start a step with the function `start` otherwise it is not possible to put the result.

Note: Do not forget to specify the result of a step, otherwise it will be considered as UNDEFINED.

Important: A step set to FAILED can not become PASS thereafter in a test.

20.3 Cancellation of a test

It is possible to force the execution of a test case by using the `stop` function in the `description` section of your test.

```
Test(self).interrupt(err="aborted by tester")
```

Using the `stop` feature will stop the test and automatically call the `cleanup` section of the test case. In this case, the `aborted` argument is set to True by the framework to indicate the cancellation of the test.

```
def definition(self):
    Test(self).interrupt("bad response received")

def cleanup(self, aborted):
    if aborted: self.step1.setFailed(actual="%s" % aborted)
```

20.4 Adding trace

The framework provides some functions to add messages during the execution of a test.

The following levels are available:

- Example to display a message of type `info`

```
Trace(self).info(txt="hello world")
```

- Example to display a `warning` message

```
Trace(self).warning(txt="hello world")
```

- Example to display an `error` message

```
Trace(self).error(txt="hello world")
```

Note: If an error message is displayed then the result will automatically be set to FAILED.

Note: Messages appear automatically in the basic report.

20.5 Data

20.5.1 Public

A public space is available on the test server. This space makes it possible to provide files that are necessary during the execution of a test.

- Public
  - addFolder
  - appendFile
  - getFile
  - getPath
  - saveFile
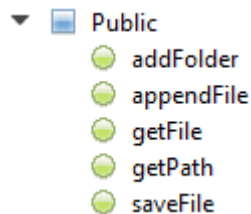
The files are stored in the `/opt/xtc/current/Var/Public/` directory on the server.

Warning: This space is common to all projects configured on the server.

20.5.2 Private

Private vault only exists while running a test. It can save logs generated or recovered during the execution of the test. These logs are automatically made available to the user in a zip file when the test is completed. They can be retrieved from the client or from the server API.
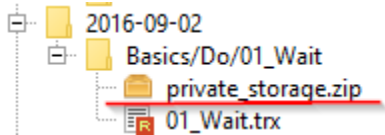
The logs are organized by directory:

- TC-TESTCASE directory - # <id_tc>: contains the logs generated by the test case

- ADP directory - # <id_id>: contains the logs generated by the different adapters used during the test
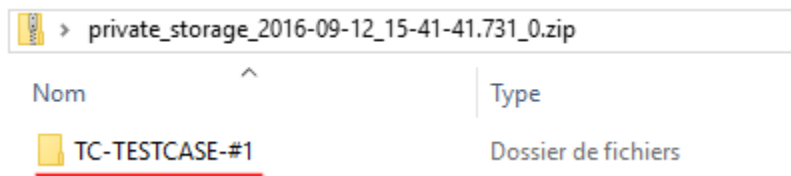
Example to save the text *hello world* in a' my_logs' file from the test case

```
Private(self).saveFile(destname="my_logs", data="hello world")
```

Example to add text to an already existing log file

```
Private(self).appendFile(destname="my_logs", data="hello world2")
```

Note:

It is also possible to save files from an adapter. They will be automatically stored in a directory with the name of the adapter.
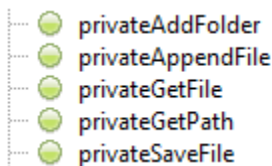
20.5.3 Cache

The test framework allows caching of data in the key/value form. This function may be necessary to share data between tests when writing a scenario for example.
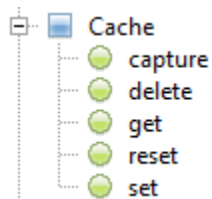
Example to save a value in the cache

```
Cache().set(name="my_data", data="hello")
```

Read a value from the cache

```
my_data= Cache().get(name="my_data")
Trace(self).warning(my_data)
```

Example to capture a data with a regular expression and with record in the cache

```
my_data="March, 25 2017 07:38:58 AM"

Cache().capture(data=my_data, regexp=".* (?P<TIME>\d{2}:\d{2}:\d{2}) .*")

Trace(self).info( txt=Cache().get(name="TIME") )
```


It is also possible to rely on a `custom` parameter to supply the regular expression.

```
.*session_id=[!CAPTURE:SESSIONID:];expires.*
```

or in greedy mode

```
.*session_id=[!CAPTURE:SESSIONID:.*?];.*
```

Important: The cache exists only during the execution of a test.

20.6 Put on hold

This function allows you to pause while running a test.

Example of holding for 10 seconds:

```
Time(self).wait(timeout=10)
```

Standby example until the current date and time match the specified date:

```
Time(self).waitUntil(dt='2016-09-12 02:00:00', fmt='%Y-%m-%d %H:%M:%S', delta=0)
```

20.7 Interaction with the tester

The framework makes it possible to write semi-automatic tests, ie in interaction mode. This function can be interesting for a test in interactive mode (eg configuration of a device)

Example asking the name of the person:

```
user_rsp = Interact(self).interact(ask="Your name?", timeout=30.0, default=None)
```

From the client, the `Interact` tab automatically appears to answer the question asked during the execution of the test. This window is available from the analysis window.
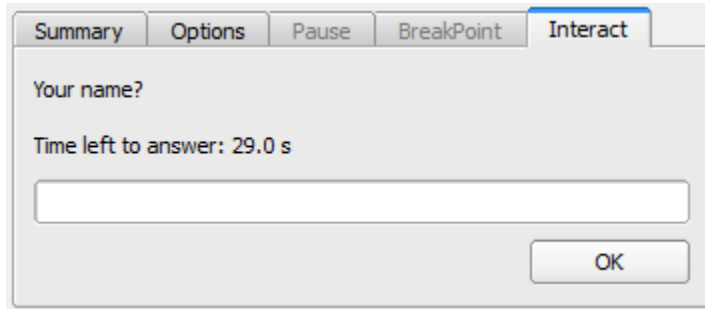
Note: If no response is provided within the interval, it is possible to provide a default value with the `default` argument.

20.8 Parameters of a test

20.8.1 Inputs

Input parameters are used to add variables to a test. They are configurable from the client.

There are several types of parameters:

The variables are accessible from a test with the `input (...)` function

```
input('DEBUG')
```

The custom parameter

The `custom` type is used to construct parameters that use other parameters or the cache. It is therefore possible to use keywords that will be interpreted by the test framework at the time of execution.

List of available keywords:

Keywords	Description
[! INPUT::]	Retrieves the value of a parameter present in the test
[! CACHE::]	Retrieves a value present in the cache

Note: The name of a parameter is unique and must be capitalized.

20.8.2 The agents

Test Properties
Prepare the test.

		Test Design		Test Data		Miscellaneous
	Agents		Probes			

Id	Name	Type	Value	Description
1	**AGENT_SOCKET**	socket	agent.win.sock01	

Inputs		Outputs		Adapters		Libraries

Id	Name	Type	Value	Description
5	**SUPPORT_AGENT**	*bool*	False	
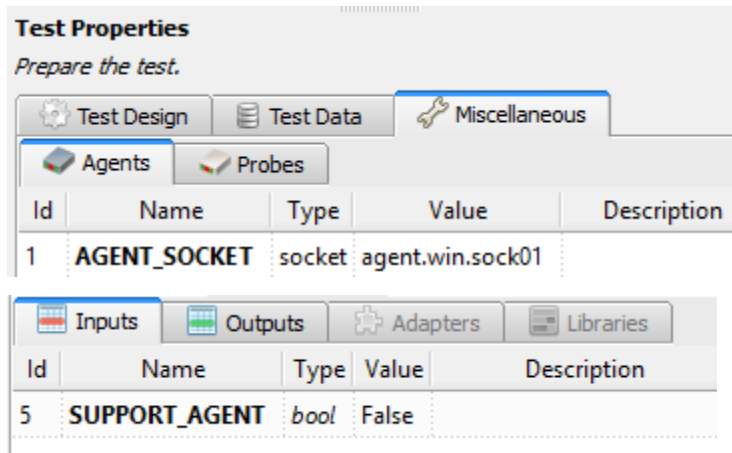
The list of agents can be accessed from a test using the () key mode.

```python
self.ADP_REST= SutAdapters.REST.Client(
                                parent=self,
                                destinationIp=input('HOST'),
                                destinationPort=input('PORT'),
                                debug=input('DEBUG'),
                                sslSupport=input('USE_SSL'),
                                agentSupport=input('SUPPORT_AGENT'),
                                agent=agent('AGENT_SOCKET')
                                )
```

20.8.3 The probes

Test Properties
Prepare the test.

	Test Design		Test Data		Miscellaneous
	Agents		Probes		

Id	Use	Name	Type	Arguments
1	☐	**probe01**	*default*	
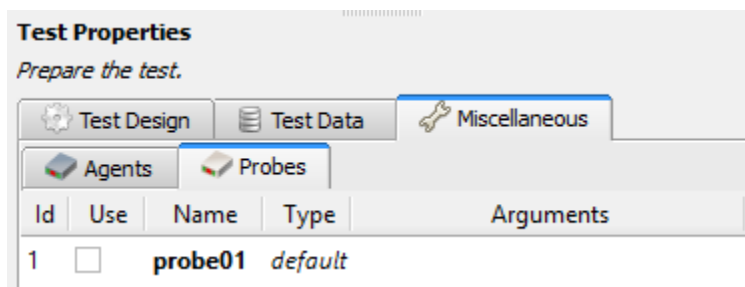
20.8.4 Import / export settings

The test parameters can be exported to a dedicated `testconfig` (tcx) file type. It is therefore possible to prepare the parameters without having the test.
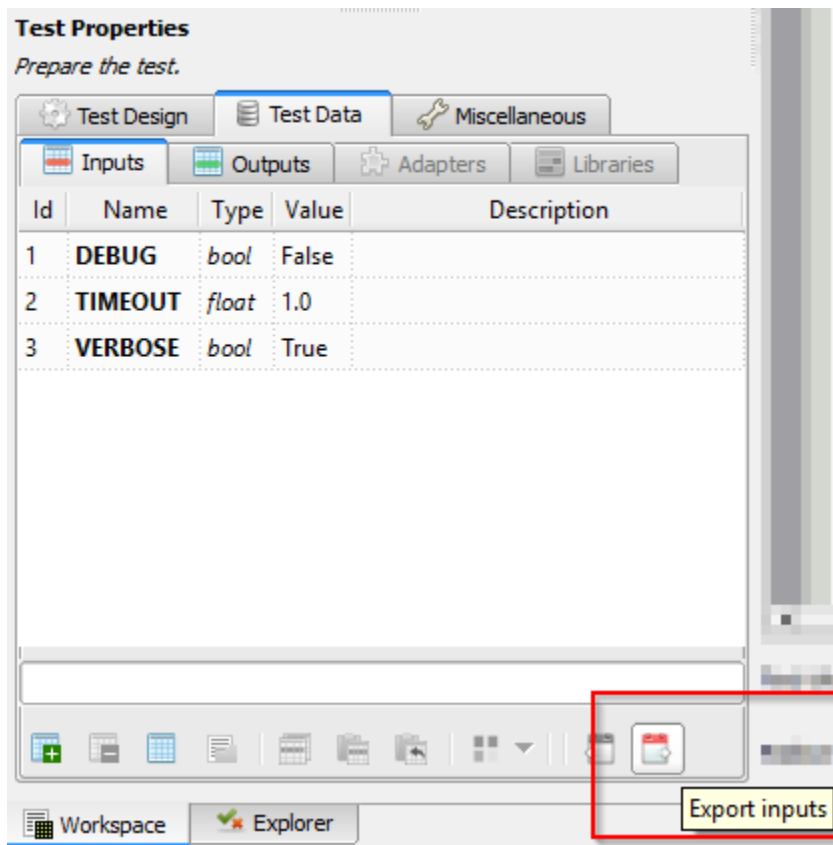
It is possible to import a configuration file into a test. The import will overwrite all the parameters if the name is the same.

CHAPTER 21

The tracability
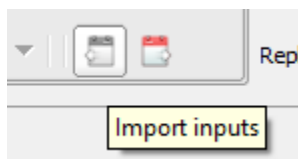
21.1 The events

The execution of a test is divided into events, all of these events are stored and can be viewed afterwards. An event can represent:

- an action performed by the test framework
- an action performed by the test
- a data item received by the system to be tested or checked.
- data to send to the system to test or control.

Event execution makes it possible to have robust tests thanks to the definition of observation intervals. The approach is to write the tests with the following formalism:

- I perform an action in my test.
- During a given interval, I look and compare all the events received with an expected.
- **I decide on the next action**
    - after receiving the event I was waiting for
    - or when the observation interval is over.

During the execution of a test, the framework captures all the events generated by the tested or piloted system. The events are then converted and stored in a message called `template`.

A `template` is split into one or more `layers`. A `layer` is defined by a set of key/value. The value of a layer can be another layer too.
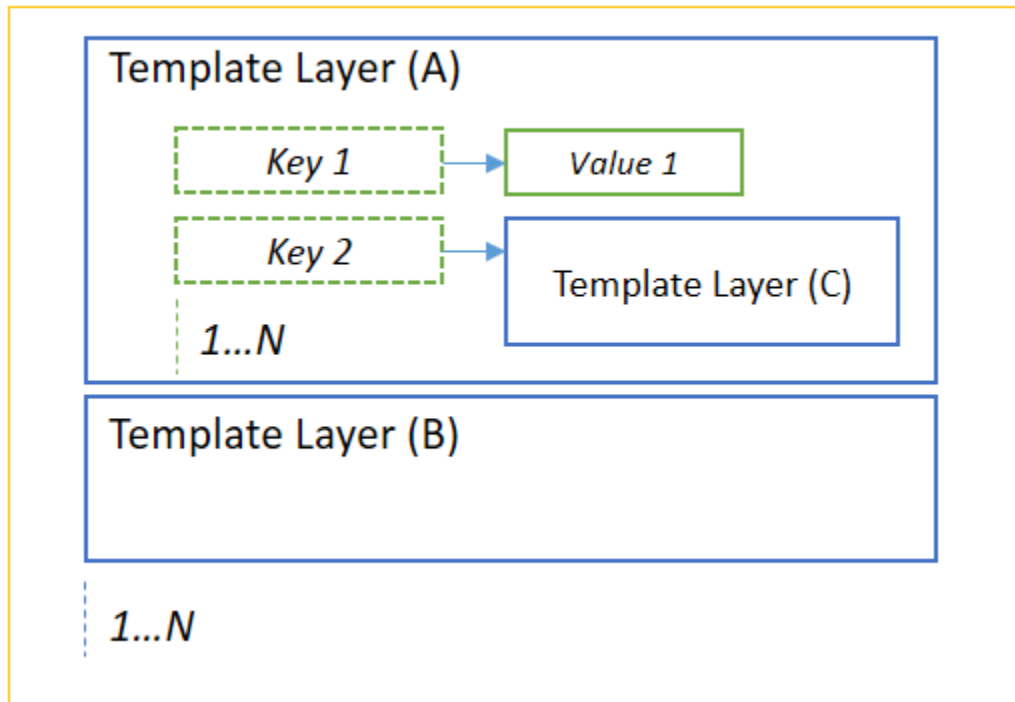
21.1.1 Creating a template

Creating a template can be done using the `TestTemplates` test framework.

```
tpl = TestTemplates.TemplateMessage ()
layer = TestTemplates.TemplateLayer (name = 'response')
layer.addKey (name = 'code', data = '200')
layer.addKey (name = 'msg', data = 'hello world')
tpl.addLayer (layer = layer)
```

This template indicates that the event should contain:

- a layer called *response* and containing the key' code' and *msg*

- the key code must be strictly equal to the value 500

- the msg key must be strictly equal to the text hello world.

Example of an expected message visible from the graphical client:

Key	Value
any()	
notcontains('source2')	any()
endswith('source-ip')	127.0.0.1
contains('TCP')	
destination-port	greaterthan(70.0)
contains('source')	60246
tcp-event	startswith('conn')
HTTP	
contains('request')	endswith('HTTP 1.1')
version	contains('1.1')
endswith('PP')	
contains('raw')	xxxxxxxxx

21.1.2 The operators

Operators are available to facilitate comparison of the models received.

Sample template using comparison operators:

```
tpl = TestTemplates.TemplateMessage()
layer = TestTemplates.TemplateLayer(name='response')
layer.addKey(name='code', data=TestOperators.LowerThan(x=500)))
layer.addKey(name='msg', data=TestOperators.Contains(x="hello"))
tpl.addLayer(layer=layer)
```

This template indicates that the event should contain:

- a layer called *response* and containing the key' code' and *msg*

- the code key must be less than 500

- the msg key must contain the text hello.

21.1.3 Visualization

The client can graphically display the comparison made by the framework.

Definition of the color code:

Green	Perfect match between the value received and expected
Red	The value received does not correspond to the expected value
Yellow	The expected value has not been verified

21.2 Test reports

After each run of a test, the framework automatically generates the associated test reports.

There are 2 type reports:

- An advanced report
- A basic report (accessible by default from the graphical client)

The reports are accessible from the client, the web interface or from the API.

Note: Reports can be exported in html, csv, xml and pdf format.

21.2.1 Advanced report

The advanced report displays information such as:

- the execution time of each test case
- the complete description of the test steps.
- performance statistics.
- the test parameters.

| Advanced Report | Basic Report | Xml Verdict | Text Verdict | Preview | Comments |

| To TXT | To HTML | To PDF | To Printer |
Exports

Test report auto-generated by Extensive Testing - Copyright (c) 2010-2017 - Denis Machard

📊 General Test Description

Global Result	FAILED				
Duration	10.859 seconds				

		PASS	FAIL	UNDEF	TOTAL
	TESTCASE:	3 (75%)	1 (25%)	0	4
	TESTABSTRACT:	0	0	0	0
Statistics	TESTUNIT:	3 (100%)	0	0	3
	TESTSUITE:	0	1 (100%)	0	1
	TESTPLAN:	1 (50%)	1 (50%)	0	2
	TESTGLOBAL:	0	1 (100%)	0	1

Terminated At	21/01/2018 19:27:11
Run At	21/01/2018 19:27:01
Run By	admin
Test Name	0001_Rest_Api
Test Path	//Self Testing/REST API/
Project Name	Common
Adapters	v1110
Libraries	v800

🔲 Test Global - 0001_Rest_Api

Summary	Just a basic sample.
Started At	21/01/2018 19:27:01
Duration	10.858 seconds

🔲 Test Plan - PREPARE TEST
🔲 Test Plan - LOGIN TO THE REST API

It is possible to display variables in the test report by prefixing the variables:

- SUT_ Variables describing the version of the system to be tested or piloted
- DATA_ Variables describing specific data
- USER_ User variables

This feature can be useful for increasing the level of traceability in reports.

Test Suite: Noname2

Summary	Just a basic sample.
Started At	07/01/2017 06:19:49
Duration	0.127 seconds
SUT Inputs	SUT_TEST (float) = 1.0

TestCase: TESTCASE_01

21.2.2 Basic report

The basic report summarizes the result of all test cases and reports.

Color code:

Green	The test case is valid
Red	The test case is in error
Orange	The result of the test case is not determined
Gray	The test case was not executed

Tip: You must click on the test cases to display the steps.

Note: The messages displayed by the test with the `Trace (self) .info ()` function are available in the report by clicking on the `[logs details]` link.

Errors are also displayed by clicking on the `[errors details]` link.

21.3 The logs

The framework allows you to save logs while running a test and make them available quickly to the uses. All additional logs are zipped and accessible from the client or the API.
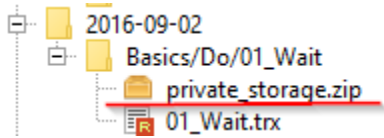
Note: For more details, read the chapter *The fundamentals >> Data*.

Interoperability

22.1 Adapters

The adapters allow communication with the system to be tested or piloted. The solution embeds several default adap

- network protocol support
- application level protocol support
- communication with databases
- systems interaction
- interaction with graphic interfaces
- telecom protocol support

Adapters have two modes of use:

- a direct mode: communication is done directly from the test server to the system to be controlled.
- an agent mode: the communication with the system to be controlled is done through an agent communicating with the test server.

Note: The Verbose mode is enabled by default on all adapters. This mode can be disabled to reduce the number of events during a test.

Note: The Debug mode is not enabled by default. It can be activated in case of problem.

Note: Examples are available in test samples /Samples/Tests_Adapters

List of adapters available by default:

22.1.1 Network Protocols

Adapters	Agents	Descriptions
ARP	socket	Sniffer to send and receive ARP packets
ICMP	socket	Sniffer to send and receive ICMP packets
Ethernet	socket	Sniffer for sending and receiving Ethernet frames
IP	socket	Sniffer for sending and receiving IPv4 packets
Pinger	not supported	Machine life tests via ICMP, TCP or URL
UDP / TCP	socket	Sniffer and UDP client and TCP
NTP	socket	Client to request an NTP server
DNS	not supported	Resolver Customer
SNMP	socket	Receiving SNMPv2 Alarms

22.1.2 Network Protocols Applications

Adapters	Agents	Descriptions
HTTP	socket	Server and client with TLS support and proxy
SOAP	socket	Client with TLS support and proxy
REST	socket	Client with TLS support and proxy
WebSocket	socket	Websocket client
SoapUI	soapui	Client to run SoapUI campaigns

22.1.3 System commands

Adapters	Agents	Descriptions
Dig		Customer dig
Curl		Customer curl
Nmap		Nmap client
Ncat		Customer ncat
Openssl		Openssl client

22.1.4 User Interfaces

Adapters	Agents	Descriptions
Adb	adb	Integration with the Android Gateway
Selenium	selenium2-server or selenium3-server	Integration with the Selenium project
Sikuli	sikulix-server	Integration with the SikuliX project

22.1.5 Data base

Adapters	Agents	Descriptions
Microsoft SQL	database	Communication with a base of type Microsoft SQL
MySQL	database	Communication with a MySQL/MariaDB database
PostgreSQL	database	Communication with a PostgreSQL database

22.1.6 System controls

Adapters	Agents	Descriptions
SSH / SFTP	ssh	SSH console
TELNET	socket	Customer to send and receive text
FTP	ftp	Customer with TLS support
System File	file	Allows interaction with Linux or Windows system files
System Win / Unix	command	Lets you control Linux and Windows systems (wmic)
Cisco Catalyst	ssh	Configuration Client, based on the Telnet adapter

22.1.7 Telecom Protocols

Adapters	Agents	Descriptions
SMS Gateway	gateway-sms	Receive or send SMS using an Android smartphone
SIP	socket	SIP Phone
RTP	socket	Module for sending and receiving audio and video streams

22.2 Bookstores

A library makes it possible to quickly make available functions for

- support data encryption methods
- support existing compression formats
- support authentication functions
- manipulate the different format of date, time and units
- support codecs (XML, JSON, etc . . .)
- support data hash functions

A library does not communicate directly with the system to be tested or piloted. It is used:

- directly from the tests
- from the adapters.

Tip: If several adapters need the same functions, it is advisable to factor them in a library.

List of libraries available by default:

22.2.1 Encryption

AES	Encryption or decryption support
Blowfish	Encryption or decryption support
OpenSSL	Execute SSL command
RC4	Encryption or decryption support
XOR	Encryption or decryption support
RSA	RSA Key Generator

Note: An example is available in test samples `/Samples/Tests_Libraries/02_Ciphers`

22.2.2 Codecs

Base64	Encode or decode in base64 format
Excel	Excel file reading
G711A	Encode or decode the audio codec
G711U	Encode or decode the audio codec
JSON	Encode or decode text in JSON format
XML	Encode or decode text in XML format

Note: An example is available in test samples `/Samples/Tests_Libraries/03_Codecs`

22.2.3 Compression

GZIP	Compression or decompression in GZIP format

Note: An example is available in test samples `/Samples/Tests_Libraries/09_Compression`

22.2.4 Hashing

Checksum	Checksum Generator
HMAC	Creating a hash md5, sha1 and sha256
MD5	Creating a md5 hash
SHA	Creating a hash sha1, sha256 and sha512
CRC32	Checksum Generator

Note: An example is available in test samples `/Samples/Tests_Libraries/05_Hashing`

22.2.5 Identifiant

SessionID	Session Builder ID
UUIDS	UUID Generator (Universally Unique IDentifier)

Note: An example is available in test samples `/Samples/Tests_Libraries/07_Identifiers`

22.2.6 Média

ChartsJS	Visible graph generator in test reports
DialTones	Tone generator
Image	Manipulation of images
Noise	Noise generator
SDP	Decodes or encodes SDP messages
WavContainer	Creating audio file type WAV
Waves	Simple wave generator

Note: An example is available in test samples /Samples/Tests_Libraries/04_Media

22.2.7 Date

Today	Retrieves today's date

Note: An example is available in test samples /Samples/Tests_Libraries/11_Date

22.2.8 Security

Basic	Decode or encode the authorization
Digest	Decode or encode the authorization
Hmac	Decode or encode the authorization
Oauth	Decode or encode the authorization
Wsse	Decode or encode the authorization
Certificate	Decodes certificates in a readable format
JWT	Decode or encode tokens

Note: An example is available in test samples /Samples/Tests_Libraries/01_Security

22.2.9 Time

Timestamp	Generate a timestamp or convert to a readable value

Note: An example is available in test samples /Samples/Tests_Libraries/06_Time

22.2.10 Units

Bytes	Convert fromtes to readable

Note: An example is available in test samples `/Samples/Tests_Libraries/08_Units`

22.3 Third party tools

The product comes at the base with a number of plugins to interface with other existing tools (defect tracking, test management, etc.).

These plugins can be used directly from a test.

List of supported tools:

Git	Clone / commit file on remote repository
Jira	Ticket creation
HP ALM QC	Test run, ticket creation. Version 12 minimum
ExtensiveAutomation	Test execution, variable creation
Jenkins	Running tests before or after a build
VSphere	VM creation or supression on VMware

Note:

The solution has a REST API, it can be driven also by these tools.

- Jenkins Plugin: https://wiki.jenkins.io/display/JENKINS/ExtensiveTesting+Plugin

---

22.3.1 HP ALM

This plugin allows you to export test results in the HP ALM tool. It can be used from an etst to export results without user intervention.

Example of use:

::

> HP ALM ——> Call REST API ——> AND ^ | | v | Execution of the requested test | v + <———
> Push the result ———+

Note: An example is available in the test samples `/Samples/Tests_Interop/02_HP_QC`

22.3.2 Jenkins

This plugin allows to launch a build from the Extensive solution.

Note: An example is available in test samples /Samples/Tests_Interop/06_Jenkins

22.3.3 VSphere

This plugin allows you to control a VMware virtual environment. It can be used for:

- create virtual machines automatically
- remove machines

Note: An example is available in test samples /Samples/Tests_Interop/05_VSphere

22.3.4 ExtensiveTesting

This plugin makes it possible to make a link between several environment (dev, integration, qualification) by allowing to run tests from one environment to another.

Note: An example is available in test samples /Samples/Tests_Interop/03_ExtensiveTesting

22.3.5 Jira

This plugin makes it possible to create tickets following the execution of a test in the tool Jira.

Note: An example is available in test samples /Samples/Tests_Interop/01_Jira

22.3.6 Git

This plugin allows you to recover or push files from a source repository. It can be used as a prerequisite for a test.

Note: An example is available in test samples /Samples/Tests_Interop/04_Git

22.4 Agents

Agents are available from the toolbox. They are to be used together with the adapters

- to communicate with the system to test or control when it is not accessible live by the test server (ex: a web page)
- run a test on several different environments.

Note: The dummy agent is to be used as a basis for developing a new agent.

22.4.1 Network Protocols

socket	Lets you start TCP / UDP sockets
ftp	Connect to an FTP server(s)
database	Queries databases (MySQL, Microsoft SQL and PostgreSQL)
ssh	Connect to machines via SSH or SFTP

22.4.2 Systems

command	Execute system commands on Windows or Linux
file	Allows you to recover files on Windows or Linux systems

22.4.3 Third party tools

sikulix-server	Interactions with heavy applications
selenium3-server	Allows you to control the latest generation web browsers
selenium2-server	Allows you to control web browsers
soapui	Allows you to run SoapUI tests
adb	Allows you to control Android smartphones
gateway-sms	Send or receive SMS

Note: Using the `Selenium3-Server` agent requires at least `Java 8` on the machine.

22.5 Probes

The probes are available in the toolbox. The main goal is to recover automatically logs (network trace, files) during the execution of a test.

textual	Allows follow-up of log files on Windows or Linux (tailf)
network	Take network traces, probe based on tcpdump on linux, or tshark on Windows
file	Recovery of configuration files on Windows or Linux

The use of a probe in a test is to be defined in the properties.

Note: The `dummy` agent is to be used as a basis for developing a new agent.

CHAPTER 23

Test engine

23.1 The scheduler

23.1.1 Programming of performances

The scheduler present in the server makes it possible to program the execution of the tests in several ways.

- Run the test once in `x_seconds` or `date_time`
- Run the test several times at *date_time*.
- Run the test at each `start time` at `finish time`
- Run the test every hour at a specific `hour`
- Run the test every day at a precise `hour`
- Run the test once a week the `day of the week` at a specific hour

Note: A recursive task will be automatically restarted by the server after a reboot.

23.1.2 Task Management

The following actions are available to manage tasks scheduled by users:

- cancel one or more tasks
- stop one or more tasks
- reprogram one or more tasks
- view the history of the performances.

All of its actions can be done from the heavy client or from the API.

23.2 Parallelized executions

It is possible to run multiple tests in parallel using the `Grouped` function This function is available from the heavy client or from the API.

There are 2 options of executions:

- run tests one after the other (no link)

- or parallel execution
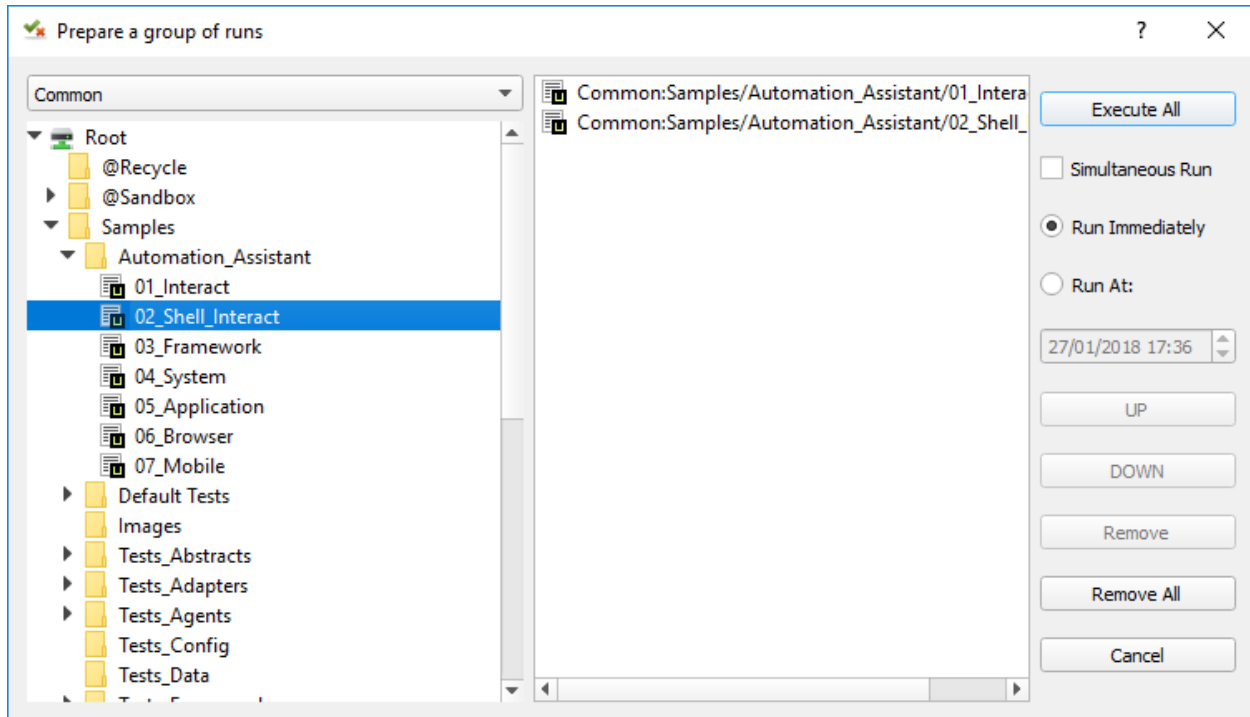
Note: From the API, use the function `/rest/tests/schedule/group`.

```
{
 "test": [
         "Common:/Samples/Tests_Unit/02_A.tux",
         "Common:/Samples/Tests_Unit/03_B.tux"
       ],
 "postpone-at": [],
 "parallel-mode": False,
 "postpone-mode": False
}
```

Important: There is no guarantee that the tests will start at the same time with this mode of execution.

23.3 Synchronized executions

23.3.1 Sharing adapters

The " shared mode" feature allows you to reuse the same adapter in several test cases. This mode is to be used in a scenario (test plan) or a test suite with several test cases.

Here is an example of possible use:

- the scenario tests an application

- in the background, the scenario also checks the logs generated by the application

- It is therefore possible to influence the result of the test based on what is found in the logs.

To enable shared mode, set the " shared" parameter to True and give the adapter a name:

```
self.ADP_EXAMPLE = SutAdapters.Dummy.Adapter(
                                    parent=self,
                                    debug=False,
                                    name="MY_ADAPTER",
                                    shared=True
                                 )
```

Note: It is important to give a name to its adapter because it makes finding it easier. If no name is given, the framework configures the adapter with a random name.

After initialization of the adapter it is possible to recover an adapter from another test case by searching for it by name.

```
self.ADP_EXAMPLE = self.findAdapter(name="MY_ADAPTER")
if self.ADP_EXAMPLE is None: Test(self).interrupt("unable to find the adapter")
```

23.3.2 Sharing data

Since the cache is unique when a test (no matter the type) is performed, it is possible to exchange data between several test cases.

A first test can record data in the cache and a 2nd test can retrieve the value stored by the 1st test.

23.3.3 Synchronization

Synchronized execution of several test cases is possible using a testplan. This scenario should contain:

- an observer test case
- one or more test cases running actions in the background

The observer test must be used to make the connection between the different adapters.

Important: The use of adapters in shared mode is mandatory.

Note: An example is available in the /Samples/Tests_Non_Sequential test samples.

23.4 Distributed executions

The solution allows for distributed executions using distributed agents across the networks.

Advanced examples

24.1 SSH adapter

The SSH adapter allows you to connect to remote servers using the SSH protocol.

The configuration of the adapter consists of indicating at least:

- the ip address of the remote server
- the remote server port (default 22)
- the user account

The adapter supports the following features:

- authentication by username and password
- key exchange authentication

Example of configuring the adapter in the prepared section of the test.

```
self.ADP_SSH = SutAdapters.SSH.Client(
                                    parent=self,
                                    login=input('LOGIN'),
                                    password=input('PWD'),
                                    destIp=input('DEST_IP'),
                                    destPort=input('DEST_PORT'),
                                    debug=input('DEBUG'),
                                    agentSupport=input('SUPPORT_AGENT')
                            )
```

Example to connect, to authenticate on a remote server and to disconnect:

```
connected = self.ADP_SSH.doConnect(
                                timeout=input('TIMEOUT'),
                                prompt='~]#'
                        )
```

```
if not connected: self.abort("ssh connect failed")
self.info("SSH connection OK" )

disconnected = self.ADP.doDisconnect(timeout=input('TIMEOUT'))
if not disconnected: self.abort("disconnect failed")
self.info("SSH disconnection OK" )
```

Example to send a command on a remote machine:

```
rsp = self.ADP_SSH. doSendCommand(
                                command='date',
                                timeout=input('TIMEOUT'),
                                expectedData=None,
                                prompt='~]#'
                    )
if rsp is None: self.abort("run command failed")
self.warning( rsp )
```

> **Warning:** SSH replies can be split into several events (this depends on the network). We must be careful when waiting for a specific response, the use of a buffer may be necessary in this case.

Note: Examples are available in the `/Samples/Tests_Adapters/05_SSH.tsx` sample.

24.2 HTTP adapter

The `HTTP` adapter is used to send requests and inspect associated responses to a web server.

The configuration of the adapter consists of indicating at least:

- the ip address of the remote server
- the remote server port (default 80)

The adapter supports the following features:

- encryption `tls` of the communication
- the use of `socks4`, `5` proxy and http
- `digest` or `basic` authentication
- reassembly of responses `chunked`

Example of configuring the adapter in the `prepared` section of the test.

```
self.ADP_HTTP = SutAdapters.HTTP.Client(
                                parent=self,
                                debug=input('TRACE'),
                                destinationIp=input('DST_IP'),
                                destinationPort=input('DST_PORT'),
                                sslSupport = input('SSL_SUPPORT'),
                                agent=agent('AGENT_SOCKET'),
```

```
                                            agentSupport=input('SUPPORT_AGENT')
                        )
```

Example to send a GET type query and a response with the 200 code.

```
rsp = self.ADP_HTTP.GET(
                        uri="/",
                        host=input('HOST'),
                        timeout=input('TIMEOUT'),
                        codeExpected=200
                    )
if rsp is None:
  self.step1.setFailed(actual="bad response received")
else:
  self.step1.setPassed(actual="http response OK")
```

Example to send a GET type query and wait for a response that meets the following criteria:

- the version must end with 1.1

- the code must not contain the value 200

- the sentence must not contain the text *Testing*

- the body of the answer must contain the text *google*

- the response must contain a header containing the text *server,* regardless of the value

```
headersExpected = { TestOperators.Contains(needle='server'): TestOperators.Any() }

rsp = self.ADP_HTTP.GET(
                        uri="/",
                        host=input('HOST'),
                        timeout=input('TIMEOUT'),
                        versionExpected=TestOperators.Endswith(needle='1.1') ,
                        codeExpected=TestOperators.NotContains(needle='200') ,
                        phraseExpected=TestOperators.NotContains(needle='Testing') ,
                        bodyExpected=TestOperators.Contains(needle='google') )
                        headersExpected=headersExpected
                        )
if rsp is None:
  self.step1.setFailed(actual="bad response received")
else:
  self.step1.setPassed(actual="http response OK")
```

24.3 Telnet adapter

The Telnet adapter is used to connect to machines with a telnet interface.

The configuration of the adapter consists of indicating at least:

- the ip address of the remote server

- the remote server port (default 23)

Example of configuring the adapter in the prepared section of the test.

```
self.ADP_TELNET = SutAdapters.Telnet.Client(
                                   parent=self,
                                   destIp=input('TELNET_IP'),
                                   destPort=input('TELNET_PORT'),
                                   debug=input('DEBUG'),
                                   agentSupport=input('SUPPORT_AGENT')
                                   )
```

Example to connect or disconnect from the remote server

```
self.ADP_TELNET.connect()
connected = self.ADP_TELNET.isConnected( timeout=input('TIMEOUT') )
if not connected: Test(self).interrupt( 'unable to connect' )

self.ADP_TELNET.disconnect()
disconnected = self.ADP_TELNET.isDisconnected( timeout=input('TIMEOUT') )
if not disconnected: Test(self).interrupt( 'unable to disconnect' )
```

Example showing how to wait for the receipt of a particular text.

```
rsp = self.ADP_TELNET.hasReceivedData(
                                   timeout=input('TIMEOUT'),
                                   dataExpected=TestOperators.Contains(needle='Password:') )
                                   )
if rsp is None: Test(self).interrupt( 'Password prompt not found' )
```

Example to send data to the remote server

```
tpl = self.ADP_TELNET.sendData(dataRaw="exemple")
```

search for a particular text. To guard against this problem, we must add an intermediary buffer, there is a complete example with the `Catalyst` adapter.

Note: An example is available in the test samples `/Samples/Tests_Adapters/12_Telnet.tsx`.

24.4 MySQL adapter

The `MySQL` adapter allows you to connect to a remote database.

The configuration of the adapter consists of indicating at least:

- the ip address of the remote server

- the remote server port (by default 3306)

- the user name

- the associated password

Example of configuring the adapter in the `prepared` section of the test.

```
self.ADP_MYSQL = SutAdapters.Database.MySQL(
                                   parent=self,
                                   host=input('HOST_DST'),
                                   user=input('MYSQL_LOGIN'),
```

```
                                password=input('MYSQL_PWD'),
                                debug=input('DEBUG'),
                                verbose=input('VERBOSE'),
                                agent=agent('AGENT_DB'),
                                agentSupport=input('SUPPORT_AGENT')
                                )
```

Example to connect or disconnect from the remote server:

```
self.ADP_MYSQL.connect(dbName=input('MYSQL_DB'), timeout=input('TIMEOUT'))

self.ADP_MYSQL.disconnect()
```

Example to execute an SQL query in the database:

```
query = 'SELECT id FROM `%s-users` WHERE login="admin"' % input('TABLE_PREFIX')
self.ADP_MYSQL.query(query=query)
rsp = self.ADP_MYSQL.hasReceivedRow(timeout=input('TIMEOUT'))
```

Note: An example is available in the /Samples/Tests_Adapters/15_Database.tsx test samples.

24.5 SNMP adapter

The SNMP adapter allows you to receive SNMP v1 or v2 alarms.

The configuration of the adapter consists of indicating at least:

- the listening address
- the listening port

Example of configuring the adapter in the prepared section of the test.

```
self.ADP_SNMP = SutAdapters.SNMP.TrapReceiver(
                                        parent=self,
                                        bindIp=get('SRC_IP'),
                                        bindPort=get('SRC_PORT'),
                                        debug=get('DEBUG'),
                                        agent=agent('AGENT_SOCKET'),
                                        agentSupport=input('SUPPORT_AGENT')
                                        )
```

Example to start listening to the server

```
self.ADP_SNMP.startListening()
listening = self.ADP_SNMP.udp().isListening( timeout=get('TIMEOUT') )
if not listening: Test(self).interrupt( 'UDP not listening' )
```

Example to wait for the reception of an alarm:

```
trap = self.UDP_ADP.hasReceivedTrap(
                                timeout=input('TIMEOUT'),
                                version=SutAdapters.SNMP.TRAP_V1,
```

```
                                         community=None,
                                         agentAddr=None,
                                         enterprise=None,
                                         genericTrap=None,
                                         specificTrap="17",
                                         uptime=None,
                                         requestId=None,
                                         errorStatus=None,
                                         errorIndex=None
                                    )
if trap is None:  Test(self).interrupt("trap expected not received")
```

Note: An example is available in the /Samples/Tests_Adapters/18_SNMP.tsx test samples.

24.6 FTP adapter (s)

The FTP adapter allows you to connect to remote servers and supports the following functions:

- TLS connection

- Download or recover files or directories

- Add / delete and rename files or directories

- List the contents of a directory

- Detect the appearance of a file or directory with the support of regular expressions.

The configuration of the adapter consists of indicating at least:

- the ip address of the remote server

- the username to login

- the password

Example of configuring the adapter in the prepared section of the test.

```
self.ADP_FTP = SutAdapters.FTP.Client(
                                         parent=self,
                                         debug=input('DEBUG'),
                                         destinationIp=input('FTP_HOST'),
                                         user=input('FTP_USER'),
                                         password=input('FTP_PWD') ,
                                         agentSupport=input('SUPPORT_AGENT')
                                         )
```

Example to connect or disconnect from the FTP server:

```
self.ADP_FTP.connect(passiveMode=True)
if self.ADP_FTP.isConnected(timeout=input('TIMEOUT')) is None:
    Test(self).interrupt("unable to connect")

self.ADP_FTP.login()
if self.ADP_FTP.isLogged(timeout=input('TIMEOUT')) is None:
```

```
    Test(self).interrupt("unable to login")
Trace(self).info("SFTP connection OK" )
```

```
self.ADP_FTP.disconnect()
if self.ADP_FTP.isDisconnected(timeout=input('TIMEOUT')) is not None:
    Test(self).interrupt("disconnect failed")
Trace(self).info("FTP disconnection OK" )
```

Example to list the contents of a directory:

```
self.ADP_FTP.listingFolder()
if self.ADP_FTP.hasFolderListing(timeout=input('TIMEOUT')) is not None:
    Trace(self).error("unable to get listing folder")
```

Example to detect a file in a directory with a regular expression:

```
self.ADP_FTP.waitForFile(
                        path='/var/log/',
                        filename='^messages-.*$',
                        timeout=input('TIMEOUT')
                    )


found = self.ADP_FTP.hasDetectedFile(
                            path=None,
                            filename=None,
                            timeout=input('TIMEOUT')
                        )
if found is None: Trace(self).error("file not found")
```

Note: An example is available in the test samples /Samples/Tests_Adapters/21_Ftp.tsx.

24.7 SFTP adapter

The SFTP adapter allows you to connect to servers with an SSH interface. The following features are supported:

- Download or recover files or directories
- Add / delete and rename files or directories
- List the contents of a directory
- Detect the appearance of a file or directory with the support of regular expressions.

The configuration of the adapter consists of indicating at least:

- the ip address of the remote server
- the username to login
- the password

Example of configuring the adapter in the prepared section of the test.

```
self.ADP_SFTP = SutAdapters.SFTP.Client(
                                    parent=self,
                                    login=input('LOGIN'),
                                    password=input('PWD'),
                                    destIp=input('DEST_IP'),
                                    destPort=input('DEST_PORT'),
                                    debug=input('DEBUG'),
                                    agentSupport=input('SUPPORT_AGENT')
                                )
```

Example to connect and disconnect from the server:

```
connected = self.ADP_SFTP.doConnect(timeout=input('TIMEOUT'))
if not connected: Test(self).interrupt("sftp connect failed")
self.info("SFTP connection OK" )

disconnected = self.ADP_SFTP.doDisconnect(timeout=input('TIMEOUT'))
if not disconnected: Test(self).interrupt("disconnect failed")
self.info("SFTP disconnection OK" )
```

Example to list the contents of a directory:

```
self.ADP_SFTP.listingFolder(
                        path="/var/log/",
                        extended=False
                        )


rsp = self.ADP_SFTP.hasFolderListing(timeout=input('TIMEOUT'))
if rsp is None: Trace(self).error("unable to get listing folder")
self.warning( rsp.get("SFTP", "result") )
```

Example to detect a file in a directory with a regular expression:

```
self.ADP_SFTP.waitForFile(
                        path='/var/log/',
                        filename='^messages-.*$',
                        timeout=input('TIMEOUT')
                    )


found = self.ADP_SFTP.hasDetectedFile(
                                path=None,
                                filename=None,
                                timeout=input('TIMEOUT')
                            )
if found is None: Trace(self).error("file not found")
```

Note: An example is available in the test samples /Samples/Tests_Adapters/22_Sftp.tsx.

24.8 ChartJS librairies

The `ChartJs` adapter, based on the javascript library of the same name, allows you to generate graphics that can be integrated into an html page. The main interest of this library is to be able to integrate graphs in the

test report.

Example configuration of the library in the prepared section of the test.

```
self.LIB_CHART = SutLibraries.Media.ChartJS(parent=self, name=None, debug=False)
```

Example to generate a bar chart and integrate it into the report

```
# génération de données
labelsAxes = ["Red", "Blue", "Yellow", "Green", "Purple", "Orange"]
dataA = [12, 19, 3, 5, 2, 3]
dataB = [22, 49, 3, 5, 23, 3]
legendDatas = ["tets", "test"]
backgroundColor = '#4BC0C0'
borderColor = '#36A2EB'

# génération du grahique
myChart = self.LIB_CHART.barChart(
                              labelsAxes=labelsAxes,
                              datas=[dataA, dataB],
                              legendDatas=legendDatas,
                              width=400,
                              height=300,
                              backgroundColors=[borderColor, backgroundColor],
                              borderColors=[borderColor, backgroundColor],
                              chartTitle="test"
                          )

# ajout du graphique dans le résultat de l'étape
self.step1.setPassed(actual="chart", chart=myChart)
```

The chart is automatically inserted into the advanced report.
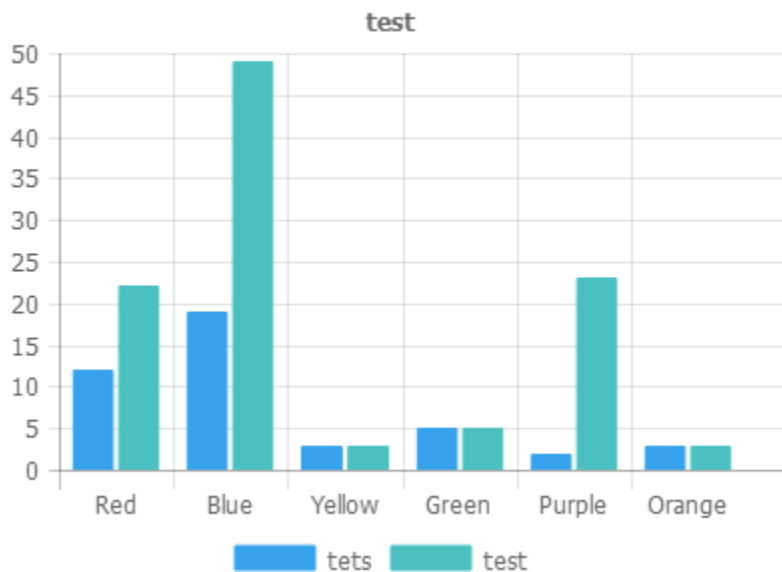
24.9 "Text" parameter

The `text` parameter is used to construct values calling other variables.

For example, consider a test containing the following 2 variables:

- DEST_IP with the value 192.168.1.1
- DEST_PORT with the value 8080

2	**DEST_IP**	*str*	192.168.1.1
3	**DEST_PORT**	*int*	8080
4	**DEST_URL**	*custom*	

The `text` type will allow us to build a 3rd variable

- DEST_URL with the value

Test Config > Custom Values

```
1 https://[!INPUT:DEST_IP:]:[!INPUT:DEST_PORT:]/welcome
```

The keyword `[! INPUT: <VARIABLE_NAME:]` allows calling another incoming variable. The framework will replace at the time of execution of the test the various keywords with the associated value. We will obtain the value https://192.168.1.1:8080/welcome for the variable DEST_URL.

To go further, it is also possible to add a value available from the cache. Assuming that the value "welcome? User = hello" is in the cache and accessible via the key "url_params". It is possible to integrate in the parameter as below

Test Config > Custom Values

```
1 https://[!INPUT:DEST_IP:]:[!INPUT:DEST_PORT:]/welcome?[!CACHE:url_params:]
```

Example of result after execution:

SECTION	TESTCASE	*Preparing*
SECTION	TESTCASE	*Starting*
STEP-STARTED	TESTCASE [Step_1]	step sample
INFO	TESTCASE	https://192.168.1.1:8080/welcome?user=hello
STEP-PASSED	TESTCASE [Step_1]	success
SECTION	TESTCASE	*Cleaning*
INFO	TESTCASE	END

24.10 "Json" parameter

todo

24.11 "alias" parameter

The alias parameter can be used to define a new name for an already existing parameter. This mechanism can be used in plan test to avoid overloading all parameters with the same name.

Example of use

1. Before execution

```
Scenario (TIMEOUT_A(int)=2 seconds)
 ---> Test 1 (TIMEOUT_A(int)=10 seconds)
 ---> Test 2 (TIMEOUT_A(int)=30 seconds)
 ---> Test 3 (TIMEOUT_A(int)=20 seconds)
```

2. After running the test

```
Scenario (TIMEOUT_A(int)=2 seconds)
  ---> Test 1 (TIMEOUT_A(int)=2 seconds)
  ---> Test 2 (TIMEOUT_A(int)=2 seconds)
  ---> Test 3 (TIMEOUT_A(int)=2 seconds)
```

When executing the above scenario, test 1, 2 and 3 are automatically set to 2 seconds for the TIMEOUT_A parameter. This is the behavior provided by the test framework.

How to do if you want the test 2 to keep the value 30 seconds against the test 1 and 2 inherit the value of the scenario?

You have to use an alias parameter, they are not overloaded by the framework.

1. Before execution

```
Scenario (TIMEOUT_A(int)=2 seconds et TIMEOUT_B(int)=30 seconds)
 ---> Test 1 (TIMEOUT_A(int)=10 seconds)
 ---> Test 2 (TIMEOUT_A(alias)=TIMEOUT_B et TIMEOUT_B(int) = 0 seconds)
 ---> Test 3 (TIMEOUT_A(int)=20 seconds)
```

2. After running the test

```
Scenario (TIMEOUT_A(int)=2 seconds et TIMEOUT_B(int)=30 seconds)
 ---> Test 1 (TIMEOUT_A(int)=2 seconds)
 ---> Test 2 (TIMEOUT_A(alias)=TIMEOUT_B et TIMEOUT_B(int)= 30 seconds)
 ---> Test 3 (TIMEOUT_A(int)=2 seconds)
```

24.12 "global" parameter

The `global` parameters are added from the web interface or from the REST API. They are global and accessible by all the tests in the same project. The expected value for this parameter is of `JSON` type.

A selection window in the graphical client allows you to select the parameter to be used in the test.
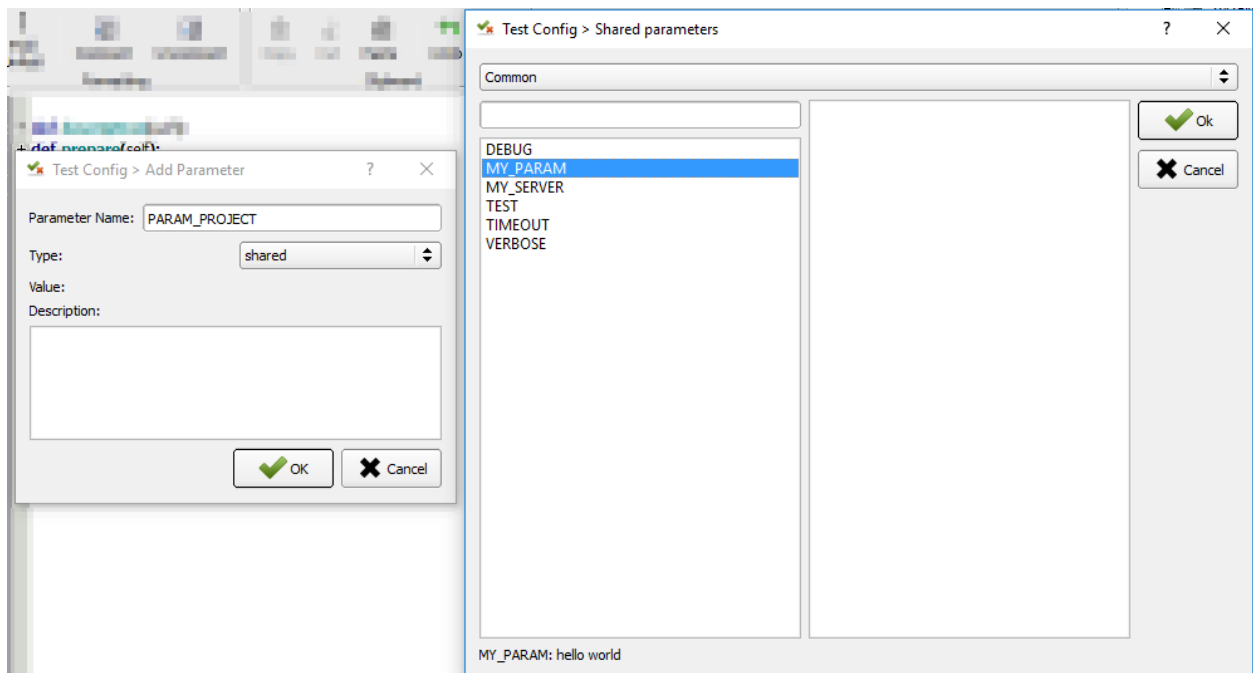
In the example below, the `MY_SERVER` test parameter contains the value of the `IP` key present in the variable global `MY_SERVER` which is itself present in the `Common` project.

Tip: To have a test parameter that contains a list of elements, use the `list-global` type.

24.13 "Dataset" parameter

The `dataset` parameter is used to import `tdx` files. A `dataset` file is just a text file, it can be created from the graphical client and saved to the remote test repository.

Sample content of a dataset file with the csv format

```
a;1;administrator
b;2;tester
```

This file can be used in a test that is important in the settings.

Example to read the variable:

```
for d in input('DATA').splitlines():
    Trace(self).info( d )
```

24.14 Using a probe

To use a probe, you need 2 things:

- Deploy the toolbox and start the desired probe.
- Declare the probe in the test.

To select the probe in the test, it must be activated and configured in the test (tab `Miscellaneous> Probes`)
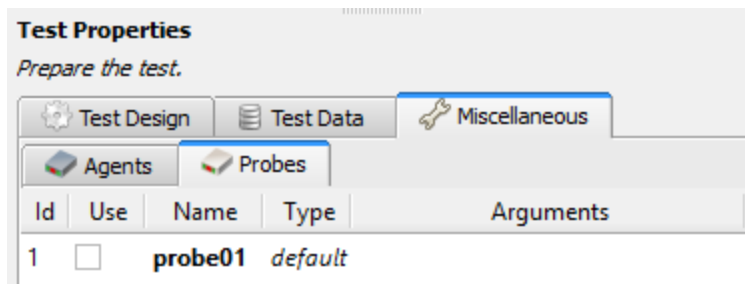
When a probe is activated on a test, running the test automatically initializes the probe.

After execution, all the files collected by the probe are downloaded to the server and accessible from the graphical client.

Note: It is possible to use multiple probes in one test.

24.15 Using an agent

To use an agent, you need two things:

- Deploy the toolbox and select the desired agent.
- Declare the agent in the test
- Configure the adapter to use the agent.

Agents are to be declared from the client in the tab `Miscellaneous> Agents`

Enabling agent mode on adapters is done with the `agentSupport` and `agent` arguments.

```
agentSupport=input('SUPPORT_AGENT'),
agent=agent('AGENT_SOCKET')
```

```
self.ADP_REST= SutAdapters.REST.Client(
                        parent=self,
                        destinationIp=input('HOST'),
                        destinationPort=input('PORT'),
                        debug=input('DEBUG'),
                        sslSupport=input('USE_SSL'),
                        agentSupport=input('SUPPORT_AGENT'),
                        agent=agent('AGENT_SOCKET')
                        )
```

In the analysis window, it is possible to see the agent used for each event:

Arrival Time: 2016-09-09 18:13:48.945

Key	Value
AGENT	
type	selenium
name	agent.win.selenium01
GUI	

Note: It is advisable to put in test parameter the use of the agent mode.

Inputs		Outputs		Adapters		Libraries

Id	Name	Type	Value	Description
5	SUPPORT_AGENT	bool	False	

Requirements

25.1 Server

For now the server can only be run on a Linux environment. It can be run on a virtual or physical environment.

Features	Minimum	Recommended
OS	CentOS 7.x	
Python	2.7	
Arch	x86_64	
Memory	4GB	8 GB
CPU	2 core	4 core
Disk	~10GB	~50GB
Network	100Mb/s	1 Gbit/s
User Number	2	5

Note: Python 3.x support is under development.

Note: Centos 6.6 can be used but with python 2.7 minimum

Note: The server can be used on other Linux system, but the integration must be done manually.

25.2 Customer

The client can be run on a Windows or Linux environment.

Features	Minimum	Recommended
OS	Windows 7+ Linux CentOS 6.5+	Windows 10+ CentOS7+ or Ubuntu 17+
Arch	x86_64	
Memory	4GB	8GB
Disk	~1GB	~2GB

Note: The 32-bit architecture is no longer supported since version 17.0.0. However it is still possible to compile the sources on a 32bits environment.

Important: The plugins for the client are only available for the Windows environment.

25.3 Toolbox

The toolbox can be run on a Windows or Linux environment.

Features	Minimum	Recommended
OS	Windows 7+ Linux CentOS 6.5+	Windows 10+ CentOS7+ or Ubuntu 17+
Arch	x86_64	
Memory	4GB	8GB
Disk	~1GB	

Note: The 32-bit architecture is no longer supported since version 17.0.0. However it is still possible to compile the sources on a 32bits environment.

Important: The plugins for the client are only available for the Windows environment.

Architecture

The solution is based on a client/server mode. The tests and adapters are centralized in a server that allows to quickly provide the same test environment to all users.

The solution consists of several components:

- A scheduler
- A graphical client
- Agents
- Probes
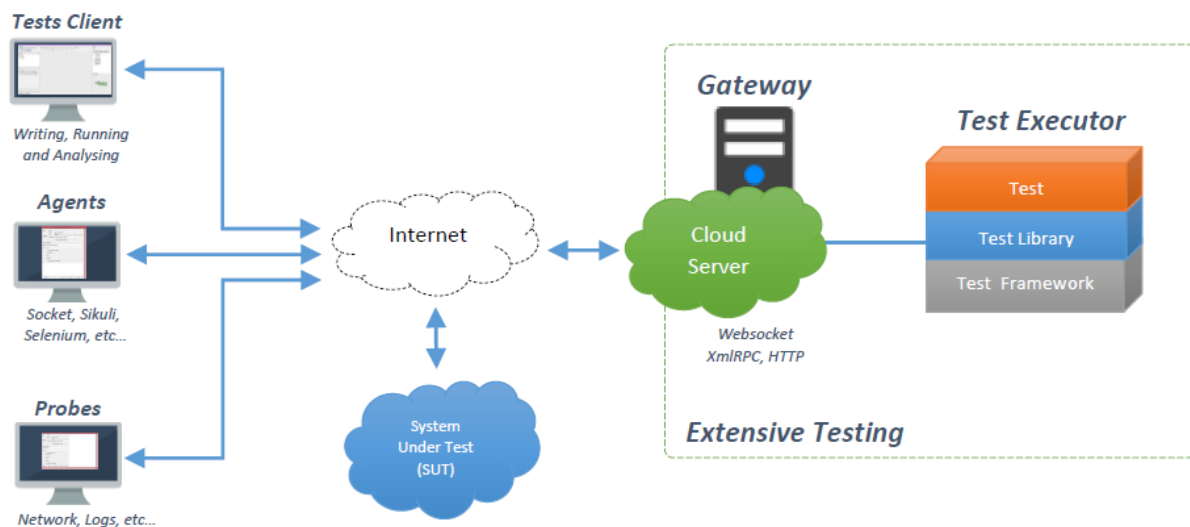
26.1 Server

The server consists of:

- a reverse proxy (apache)
- a scheduler
- a REST API server
- the test framework
- adapters and libraries
- tool extensions
- a web interface

26.2 Graphic Client

The graphical client uses a single tcp/443 (https) stream between the client and the server. The stream is bidirectional and the client can:

- make calls to the server's REST API
- receive events from the server via WebSockets.

26.3 Agents

An agent is compulsorily controlled by an adapter via the test server intermediary. It allows to deport the communication point with the system to test or control. Agents use a single tcp/443 (https) stream to communicate with the server.

26.4 Probes

The probes make it possible to recover during the execution of a test the network traces, configuration files or logs. Communication with the server only takes place on port tcp/443 (https).

Specifications

27.1 Version cycle

The set of software packages for the solution follows the following rules for naming versions.

The version is divided into 3 digits (A.B.C)

- **A: the 1st digit indicates the major version. Incrementation of this figure implies**
    - the addition of major features (with potentially a loss of compatibility with the previous version)
    - adding minor features
    - bug fix
- **B: The 2nd digit indicates a minor version. Incrementing this number indicates**
    - adding minor features
    - bug fix
- **C: the 3rd digit indicates a maintenance version. Incrementing this number indicates**
    - bug fix

27.2 Server tree

All files handled by the server are stored in the `/opt/xtc/current/` directory.

```
ServerEngine/
ServerControls/
ServerInterfaces/
ServerRepositories/
Libs/
```

```
Scripts/
Packages/
TestCreatorLib/
TestExecutorLib/
TestInterop/
SutAdapters/
SutLibraries/
Var/
  Tests/
  TestsResults/
  Logs/
  Backups/
Web/
```

The tests are stored in the `/opt/xtc/current/Var/Tests/` directory, they are organized by project ID.

27.3 Data model

A database is used by the server to store:

- the users of the solution

- the list of projects

- test data (project variables)

- statistics

- the history of executions

Tables	Description
xtc-agents	Not used
xtc-agents-stats	Not used
xtc-probes	Not used
xtc-probes-stats	Not used
xtc-config	Server Configuration
xtc-projects	List of projects
xtc-relations-projects	Relationship between projects and users
xtc-users	List of users
xtc-users-stats	Connection Statistics
xtc-test-environment	List of variables in JSON format
xtc-tasks-history	History of tasks running on the server
xtc-scripts-stats	Statistics on tests run
XTC-testabstracts-stats	Statistics on tests run
xtc-testcases-stats	Statistics on tests run
xtc-testsuites-stats	Statistics on tests run
xtc-testunits-stats	Statistics on tests run
xtc-testplans-stats	Statistics on tests run
xtc-testglobals-stats	Statistics on tests run
xtc-writing-stats	Statistics on the duration of writing tests

27.4 Passwords management

No password (in plain text) is stored in the database. Using a hash is however used. The hash of the password is stored in the *xtc-users* table.

The algorithm used:

```
MariaDB [xtc112]> select id, login, password from 'xtc-users';
+----+-----------+----------------------------------+
| id | login     | password                         |
+----+-----------+----------------------------------+
|  1 | system    | 146aeec808258c699abf36fb9cb81697833a946f |
|  2 | admin     | c9188bd805ea6c0018469b5bf106fb38c3cb9f88 |
|  3 | leader    | c9188bd805ea6c0018469b5bf106fb38c3cb9f88 |
|  4 | developer | c9188bd805ea6c0018469b5bf106fb38c3cb9f88 |
|  5 | tester    | c9188bd805ea6c0018469b5bf106fb38c3cb9f88 |
|  6 | automaton | c9188bd805ea6c0018469b5bf106fb38c3cb9f88 |
+----+-----------+----------------------------------+
```

27.5 File format

The tests are in XML format. There are several test formats:

- Test Abstract Xml

- Xml Test Unit

- Xml Test Suite

- Xml Test Plan

- Global Xml Test

Common XML Structure

```
<?xml version="1.0" encoding="utf-8" ?>
<file>
    <properties>
        <descriptions>...</descriptions>
        <inputs-parameters>...</inputs-parameters>
        <outputs-parameters>...</ outputs -parameters>
    </properties>
</file>
```

Test Abstract Xml

```
<?xml version="1.0" encoding="utf-8" ?>
<file>
    <properties>...</properties>
    <teststeps>
        <steps>
            <step>
                <id>1</id>
                <description>
                    <type>string</type>
                    <value>step description</value>
                </description>
                <summary>
                    <type>string</type>
```

```xml
                    <value>step sample</value>
                </summary>
                <expected>
                    <type>string</type>
                    <value>result expected</value>
                </expected>
            </step>
        </steps>
    </teststeps>
    <testadapters><adapters /></testadapters>
    <testlibraries><libraries /></testlibraries>
    <testactions>
        <actions>
            <action>
                <item-id>1</item-id>
                <item-text>Start</item-text>
                <item-type>2</item-type>
                <item-data />
                <pos-y>1750.0</pos-y>
                <pos-x>2000.0</pos-x>
            </action>
        </actions>
    </testactions>
    <testaborted><aborted /></testaborted>
    <testdefinition><![CDATA[pass]]></testdefinition>
    <testdevelopment>1448190709.095677</testdevelopment>
</file>
```

Test Unit Xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<file>
    <properties>....</properties>
    <testdefinition><![CDATA[pass]]></testdefinition>
    <testdevelopment>1448190694.813723</testdevelopment>
</file>
```

Test Suite Xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<file>
    <properties>...</properties>
    <testdefinition><![CDATA[pass]]></testdefinition>
    <testexecution><![CDATA[pass]]></testexecution>
    <testdevelopment>1448190717.236711</testdevelopment>
</file>
```

Test Plan Xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<file>
    <properties>...</properties>
    <testplan id="0">
        <testfile>
            <id>1</id>
            <color />
```

```
            <file>Common:Defaults/testunit.tux</file>
            <enable>2</enable>
            <extension>tux</extension>
            <alias />
            <type>remote</type>
            <parent>0</parent>
            <properties>....</properties>
            <description />
        </testfile>
    </testplan>
    <testdevelopment>1448190725.096519</testdevelopment>
</file>
```

Test Global Xml

```
<?xml version="1.0" encoding="utf-8" ?>
<file>
    <properties>...</properties>
    <testplan id="0">
        <testfile>
            <id>1</id>
            <color />
            <file>Common:Defaults/testplan.tpx</file>
            <enable>2</enable>
            <extension>tpx</extension>
            <alias />
            <type>remote</type>
            <parent>0</parent>
            <properties>...</properties>
            <description />
        </testfile>
    </testplan>
    <testdevelopment>1448190733.690697</testdevelopment>
</file>
```

27.6 Storage of test results

The test results are stored on the server in the `/opt/xtc/current/Var/TestsResult` directory.

The results are stored:

- by the id of the test projects

- by the date of the day of execution of the test

- and finally by the date and time of the tests.

Organization of the results:

```
Répertoire: <project_id>
   - Répertoire: <yyyy-mm-dd>
      - Répertoire: <yyyy-mm-dd_hh:mm:ss.testid.testname.username>
         - Fichier: TESTPATH
         - Fichier: test.out
         - Fichier: test.ini
         - Fichier: <testname>_<replayid>.hdr
```

```
    - Fichier: <testname>_<replayid>_<result>_<nbcomments>.trv
    - Fichier: <testname>_<replayid>.tbrp
    - Fichier: <testname>_<replayid>.tdsx
    - Fichier: <testname>_<replayid>.trd
    - Fichier: <testname>_<replayid>.trp
    - Fichier: <testname>_<replayid>.trpx
    - Fichier: <testname>_<replayid>.trv
    - Fichier: <testname>_<replayid>.trvx
```

Description of files:

- `TESTPATH` contains the full path for the test result

- `test.out` contains the internal logs of the test, to be used to debug the test framework

- `test.ini` contains test-specific parameters

- `<testname>_<replayid>.hdr` represents the header of the test result

- `<testname>_<replayid>_<result>_<nbcomments>.trv` contains all the events generated during the execution of the tests

- `<testname>_<replayid>.tbrp` contains the basic report in html format

- `<testname>_<replayid>.trp` contains the full report in html

- `<testname>_<replayid>.trv` contains the results report in csv format

27.7 Control Agents

The control of the agents since a test is carried out through:

- the adapters

- and the server

The communication takes place with the exchange of some specific messages:

- `init`: allows to initialize an agent

- `notify`: send a message to the agent without waiting for a response

- `reset`: allows to reset the agent

- `error`: allows the agent to send an error to the adapter

- `data`: allows the agent to send data to the adapter

Direction of available communications:

- Agent -> server -> adapter -> test

- Test -> adapter -> server -> agent

	Agent	
	Function	Callback
Send an error message	def sendError * request * data	
Send a "notify" message	def sendNotify * request * data	
Send a "data" message	def sendData * request * data	
Receiving an "init" message		def onAgentInit * customer * tid * request
Receiving a "reset" message		def onAgentNotify * customer * tid * request
Receiving a "notify" message		def onAgentReset * customer * tid * request

	Adapter	
	Function	Callback
Receiving an error message		def receivedErrorFromAgent * data
Receiving a "notify" message		def receivedNotifyFromAgent * data
Receiving a "data" message		def receivedDataFromAgent * data
Send an "init" message	def initAgent * data	
Send a "reset" message	def resetAgent	
Send a "notify" message	def sendNotifyToAgent * data	

27.8 The server logs

The server logs are located in the `/opt/xtc/current/Var/logs/` directory.

access_rp.log	apache logs for reverse access
access_ssl_rp.log	apache logs for reverse ssl access
access_web.log	apache logs for web interface access
error_rp.log	apache error logs for reverse access
error_ssl_rp.log	apache error logs for reverse ssl access
error_web.log	apache errors log for web interface access
output.log	server logs

Contributions

28.1 Solution development

28.1.1 Graphic client

Environment x64 win py3.6 qt5

Tip: Recommended environment.

To prepare its development environment, it is necessary to recover and install the following software:

- Python 3.6.3 64bits
- Git-2.15.0-64-bit.exe
- TortoiseGit-2.5.0.0-64bit.msi
- InnoSetup 5.5.9 - http://www.jrsoftware.org/isdl.php

Add additional Python packages with the `pip` command

```
> py -m pip install pyinstaller pylint
> py -m pip install pyqt5
> py -m pip install qscintilla
```

And retrieve the sources of the client since the deposit on github.

```
python Main.py
```

Warning: Windows XP is not supported in this mode.

---

x64 environment win py3.4 qt4

> **Warning:** This development environment is no longer recommended.

To prepare its development environment, it is necessary to recover and install the following software:

- Python 3.4.4 64bits
- PyQt 4.11.4
- Git-2.15.0-64-bit.exe
- TortoiseGit-2.5.0.0-64bit.msi
- InnoSetup 5.5.9 - http://www.jrsoftware.org/isdl.php

Install additional Python packages with the `pip` command

```
> C:\Windows\system32>py -3.4 -m pip install py2exe Cx_Freeze pyinstaller pylint
```

> **Warning:** A update must be done in the py2exe library. Edit the *C:Python34Libsite-packagespy2exeicons.py* file Look for the *if iconheader.idCount* line and change the value 10 to 14.

Environment x64 centos py2.7 qt4

Preparation of the development environment on a Linux CentOS 6 or 7 system.

```
yum install epel-release PyQt4 python-test
yum install PyQt4-webkit qscintilla-python
yum install python-pip
yum install PyQt4-devel
```

```
pip install dpkt
pip install cx_freeze
```

Download the client's sources from the deposit on github.

```
cd Scripts/qt4/
bash MakeResources.sh
Building files resources...
bash MakeTranslations.sh
Building translations resources...
cd ../..
```

```
python Main.py
```

x64 environment ubuntu py3.5 qt5

Preparing your development environment on a Ubuntu 17.04 Linux system

```
sudo apt-get y install python3-pyqt5
sudo apt-get y install python3-pyqt5.qsci
sudo apt-get y install python3-pyqt5.qtwebengine
sudo apt-get y install pyqt5-dev-tools
```

```
sudo pip install dpkt
```

Retrieve the client's sources from the github repository.

```
cd Scripts/qt5/
chmod +x MakeResources.sh MakeTranslations.sh
bash MakeResources.sh
Building files resources...
bash MakeTranslations.sh
Building translations resources...
cd ../..
```

```
python3 Main.py
```

28.1.2 Toolbox

Environment x64 win py3.6 qt5 (recommended)

To prepare its development environment, it is necessary to recover and install the following software:

- Python 3.6.3 64bits

- Git-2.15.0-64-bit.exe

- TortoiseGit-2.5.0.0-64bit.msi

- InnoSetup 5.5.9 - http://www.jrsoftware.org/isdl.php

```
> py -m pip install pyinstaller pylint
> py -m pip install pyqt5
> py -m pip install qscintilla
```

Install the libraries used by the different agents:

```
> py -3.6 -m pip install Cx_Freeze py2exe pyinstaller pylint
> py -3.6 -m pip install requests PyMySQL psycopg2 paramiko
> py -3.6 -m pip install pymssql-2.1.3-cp36-cp36m-win_amd64.whl
```

Install the dedicated selenium library for the solution:

```
> c:\Python36\python.exe setup.py install
```

x64 environment win py3.4 qt4

To prepare its development environment, it is necessary to retrieve and install the following packages:

- Python 3.4.4 64bits

- PyQt 4.11.4

- Git-2.15.0-64-bit.exe

- TortoiseGit-2.5.0.0-64bit.msi

- InnoSetup 5.5.9 - http://www.jrsoftware.org/isdl.php

Install the libraries used by the different agents:

```
> py -3.4 -m pip install Cx_Freeze py2exe pylint
    > py -3.4 -m pip install requests PyMySQL psycopg2 pymssql paramiko
```

Install the dedicated selenium library for the solution:

```
> c:\Python34\python.exe setup.py install
```

Environment x64 centos py3.5 qt5

Preparation of its development environment on a Linux CentOS 6 or 7 system.

Install the Qt5 library (binding python)

```
sudo apt-get y install python3-pyqt5
sudo apt-get y install pyqt5-dev-tools
cd Scripts/qt5/
chmod +x MakeResources.sh MakeTranslations.sh
bash MakeResources.sh
Building files resources...
bash MakeTranslations.sh
Building translations resources...
cd ../..
```

Install additional libraries

```
sudo apt install python3-pip
pip3 install pyinstaller py2exe pylint
pip3 install paramiko requests
pip3 install PyMySQL psycopg2
pip3 install pymssql
unzip selenium-3.7.0-extensivetesting.zip
cd selenium-3.7.0/
sudo python3 setup.py install
```

Retrieve the client's sources from the github repository.

Running the toolbox in graphical mode

```
python3 Systray.py
```

Environment x64 centos py2.7 qt4

Preparation of its development environment on a Linux CentOS 6 or 7 system.

Install additional libraries

```
yum install python-test
yum install python-pip
pip install pyinstaller py2exe pylint
pip install paramiko requests
pip install PyMySQL psycopg2
pip install pymssql
unzip selenium-3.7.0-extensivetesting.zip
cd selenium-3.7.0/
python setup.py install
```

Install the Qt4 library (binding python)

```
yum install epel-release PyQt4
yum install PyQt4-devel
cd Scripts/qt4/
chmod +x MakeResources.sh MakeTranslations.sh
bash MakeResources.sh
Building files resources...
bash MakeTranslations.sh
Building translations resources...
cd ../..
```

Retrieve the client's sources from the github repository.

Running the toolbox in graphical mode

```
python Systray.py
```

28.1.3 Server

Environment x64 centos py2.7

Preparation of its development environment on a Linux CentOS 6.5 and above.

Installing system packages

```
vim
net-snmp-utils
unzip
zip
gmp
wget
curl
ntp
nmap
bind-utils
postfix
dos2unix
openssl
openssl-devel
tcpdump
mlocate
mariadb-server
mariadb
mariadb-devel
httpd
mod_ssl
php
php-mysql
php-gd
php-pear
python-lxml
MySQL-python
policycoreutils-python
python-setuptools
python-ldap
gcc
python-devel
Cython
```

(continues on next page)

```
java
git
libffi-devel
libpng-devel
libjpeg-devel
zlib-devel
freetype-devel
lcms-devel
tk-devel
tkinter
postgresql
postgresql-libs
postgresql-devel
```

Installing python libraries

```
six
appdirs
pyparsing
packaging
setuptools
httplib2
uuidlib
pycrypto
pyasn
ply
pysmi
pysnmp
freetds
setuptools_git
pymssql
ecdsa
pil
selenium
suds
requests
ntlm
kerberos
postgresql
xlrd
etxmlfile
jdcal
openxl
libpqxx
scandir
pycnic
xlwt
isodate
xml2dict
setuptools_scm
pytest
wcwidth
pyte
pysphere
pychef
idna
```

```
enum34
ipaddress
pycparser
cffi
orderddict
ntlm_auth
requests_ntlm
py_ntlm3
pywinrm
asn1crypto
cryptography
paramiko
jsonpath
wrapt
pbr
pytz
pyjenkins
snmap2
gitdb2
pygit
```

28.2 Plugins development

28.2.1 Adapter

Adding an adapter is done using the graphical client. You have to go to the `Modules Listing> Adapters` repository and right-click on the tree to add an adapter.
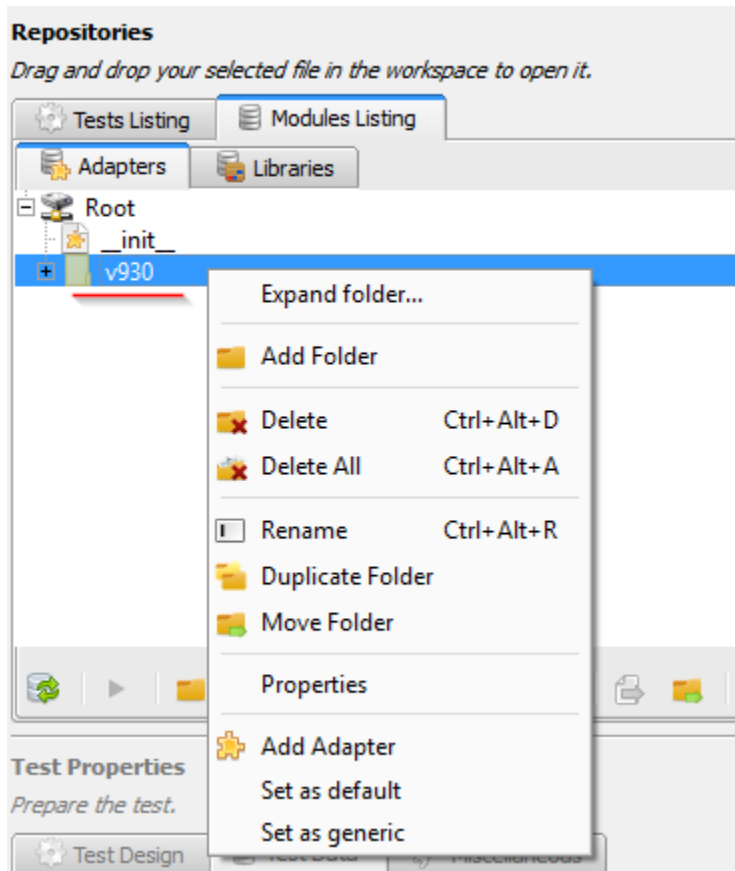
To make the adapter available for testing, you need to edit the __init __. Py file and add the lines following:

```
import Example
__HELPER__.append("Example")
```

To make the adapter appear in the documentation accessible from the graphical client, it is necessary to use the decorator @doc_public in front of the functions that one wishes to document.

```
class Example(TestAdapterLib.Adapter):
  @doc_public
      def __init__(self, parent)

  @doc_public
  def connect(self, timeout=5.0):
```
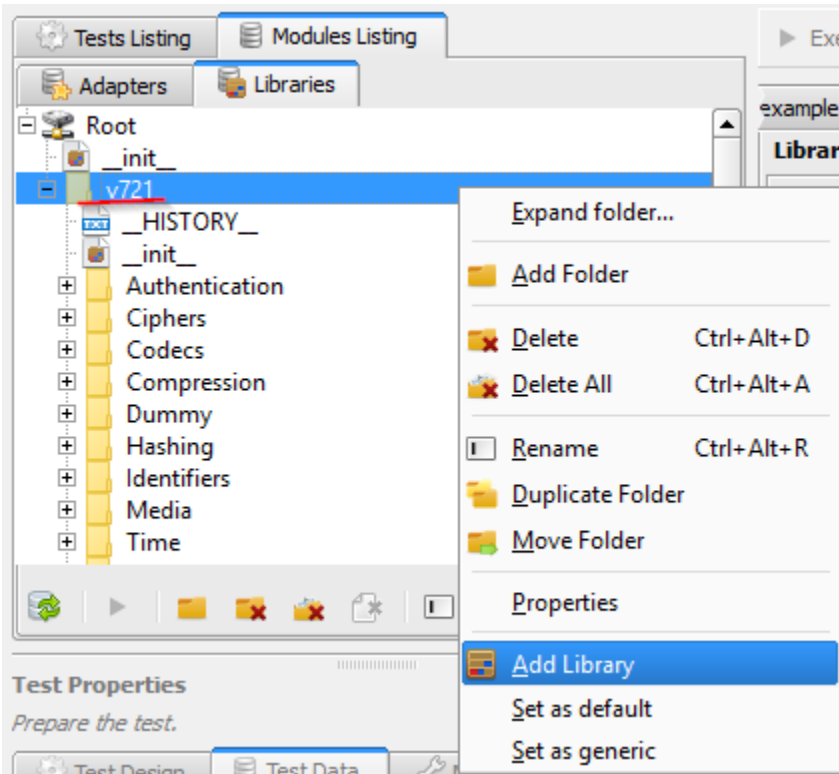
Tip: The Dummy adapter is to be used as a basis for development.

28.2.2 Libraries

Adding a library is done using the graphical client. You have to go to the *Modules Listing > Libraries* repository and right-click on the tree to add a library.

To make the library available for testing, you need to edit the `__init __.` Py file and add the lines following:

```
import Example
__HELPER__.append("Example")
```

To make the library appear in the documentation accessible from the graphical client, it is necessary to use the decorator `@doc_public` in front of the functions that one wishes to document.

```
class Example(TestLibraryLib.Library):
  @doc_public
    def __init__(self, parent)

  @doc_public
  def connect(self, timeout=5.0):
```

Tip: The `Dummy` library is to be used as a basis for development.

28.2.3 SDK Toolbox

Linux environment

Tip: It is recommended to use the `dummy` plugin as a basis for developing your agent or probe.

Using as a basis the agent or probe `dummy`, then:

- update the variable `__TYPE__` to indicate the name of the agent or the probe

---

- change the name of the class with the name of your agent or probe.
- update the `__init__` file to import your agent or probe.

Windows environment

The SDK for plugin creation is retrieved from github. It is possible to copy the plugin `Dummy` and use it as a base.

The type and name of the plugin is to be configured in the *config.json* file

```
{
  "plugin": {
          "name": "MyExample",
          "version": "1.0.0"
          }
}
```

The author is defined in the `MyPlugin.py` file.

```
# name of the main developer
__AUTHOR__ = 'Denis Machard'
# email of the main developer
__EMAIL__ = 'd.machard@gmail.com'
```

Building the plugin in binary is done by calling the `MakeExe3.bat` script.

28.2.4 Customer SDK

The client supports adding plugins. Creating a plugin requires:

- to use the SDK
- to define its type

List of possible plugins types:

Type	Description
basic	Plugin to add a shortcut on the home page
recorder-app	Export/import data in the design wizard
recorder-web	Export/import data in the design wizard
recorder-framework	Export/import data in the design wizard
recorder-android	Export/import data in the design wizard
recorder-system	Export/import data in the design wizard
remote-tests	Export/import of data in remote tests
test-results	Exporting test results and reports

The SDK for plugin creation is retrieved from github. It is possible to copy the plugin `Dummy` and use it as a basis for development.

The type and name of the plugin is to be configured in the " config.json" file

```
{
  "plugin": {
          "name": "MyExample",
          "type": "recorder-app",
          "version": "1.0.0"
```

```
            }
}
```

The author is defined in the `MyPlugin.py` file.

```python
# name of the main developer
__AUTHOR__ = 'Denis Machard'
# email of the main developer
__EMAIL__ = 'd.machard@gmail.com'
```

Building the plugin in binary is done by calling the `MakeExe3.bat` script.

The exchange of data between the plugin and the client is done with messages of `JSON` type.

1. Send data to the customer:

```python
self.core().sendMessage( cmd='import', data = {"my message": "hello"} )
```

2. Receiving data from the client:

```python
class MainPage(QWidget):
    def insertData(self, data):
```

To facilitate troubleshooting, it is possible to add traces from the plugin.

1. Add traces to the dedicated graphics window:

```python
self.core().debug().addLogWarning("my warning message")
self.core().debug().addLogError( "my error message")
self.core().debug().addLogSuccess("my success message" )
```

2. Add traces to log files:

```python
Logger.instance().debug("my debug message")
Logger.instance().error("my error message")
Logger.instance().info("my info message")
```

Tip: It is possible to run the plugin without the client by activating the debug mode.

28.3 Documentation

The documentation is stored on github in the *repository <https://github.com/ExtensiveAutomation/extensiveautomation.readth* _. It is possible to contribute by applying for participation in the deposit.

The documentation is generated by the *readthedocs <https://readthedocs.org/>* _ service.

API

29.1 Authentication

Authentication in the API REST can be done with 2 methods:

- By using the function login to obtain a session cookie
- with basic auth

29.1.1 Basic

The basic auth must be used with the api key available through the web interface. It's possible to generate a new key in the server, to do that you can use the script /opt/xtc/current/Scripts/generate-apikey.py.

```
./generate-apikey.py --user=admin
API Key ID: admin
API Key Secret: c025e7a501f144a2e1b40f9f3a91c10a47c8b1d3
API Key: YWRtaW46YzAyNWU3YTUwMWYxNDRhMmUxYjQwZjlmM2E5MWMxMGE0N2M4YjFkMw==
```

After that, add the header Authorization in your HTTP request.

```
Authorization: Basic <my_api_key>
```

Note: With the basic auth, it's not necessary to call the login function.

29.1.2 Session cookie

The authentication by cookie can be done by calling the function login to generate a cookie for the session. This cookie must be present on all next http request in the header Cookie.

```
Cookie: session_id=NjQyOTVmOWNlMDgyNGQ2MjlkNzAzNDdjNTQ3ODU5MmU5M
```

29.2 Usage example

The REST api is available throught the tcp port 443 (https) with the uri /rest.

The following example show how to execute a test with the basic auth.

```
POST /rest/tests/schedule HTTP/1.1
[...]
Authorization: Basic YWRtaW46N2UwMDExY2I3Y2ZhMGQ1MjM4NGQ1YWYyM2QyODBiMjUyM2EzMTA3ZA==
Content-Type: application/json; charset=utf-8
[...]

{
  "project-id": 1,
  "test-extension": "tux",
  "test-name": "01_Wait",
  "test-path": "/Snippets/Do/",
  "test-inputs": [ {"name": "DURATION", "type": "int", "value": 5} ]
}
```

Received response:

```
{
  "cmd": "/tests/schedule",
  "message": "background"
  "test-id": "a3f19398-b463-41e8-9e43-af86aac44a59",
  "task-id": 17,
  "tab-id": 0
  "test-name": "01_Wait"
}
```

29.3 Ressources

Description of most important functions:

Authentication

/rest/session/login	Details
/rest/session/logout	Details

Execute a test

/rest/tests/schedule	Details
/rest/tests/schedule/tpg	Details

Read the result of a test executed

/rest/results/reports	Details
/rest/results/status	Details
/rest/results/verdict	Details