



RemoteLaunch® | Automated Testing

Integration & User Guide

Inflectra Corporation

Date: April 30th, 2018



Contents

Introduction	1
1. RemoteLaunch Guide	2
2. HP UFT / QTP	18
3. SmarteScript	26
4. TestComplete	31
5. Selenium	40
6. Squish	52
7. Command-Line	62
8. LoadRunner	72
9. SoapUI	79
10. FitNesse	88
11. NeoLoad	94
12. TestPartner	101
13. BadBoy	106
14. JMeter	114
15. Ranorex	122
16. Rational Functional Tester	131
17. TestingAnywhere	138
18. Worksoft Certify	144
19. ZAPTEST	151

Introduction

SpiraTest® provides an integrated, holistic Quality Assurance (QA) management solution that manages requirements, tests and incidents in one environment, with complete traceability from inception to completion.

SpiraTeam® is an integrated Application Lifecycle Management (ALM) system that manages your project's requirements, releases, test cases, issues and tasks in one unified environment. SpiraTeam® contains all of the features provided by SpiraTest® - our highly acclaimed quality assurance system and SpiraPlan® - our agile-enabled project management solution.

This guide outlines how to use the RemoteLaunch® add-on for SpiraTest® or SpiraTeam® to remotely schedule and launch automated tests using a variety of commercial and open-source test automation engines.

For information regarding how to use SpiraTest itself, please refer to the *SpiraTest User Manual*.

The first section outlines how to use the RemoteLaunch® application, and the subsequent sections describe each of the plugins that support a different test automation tool.

1. RemoteLaunch Guide

There are actually two separate versions of RemoteLaunch® that are available from Inflectra:

1. The Microsoft Windows® compatible **Spira RemoteLaunch®** application that provides a graphic user interface application for executing automated tests on remote computers using various plugins for different testing technologies and have the results be sent to the configured SpiraTest/SpiraTeam server.
2. The cross-platform **Spira RemoteLaunchX™** Java application that provides a lightweight console application that can execute simple command line scripts on the target computer and send the results back to the configured SpiraTest/SpiraTeam server. This application can be used in **Microsoft Windows®, Linux or Apple MacOS X®** computers provided that they have the Java 1.7 (or later) runtime installed.

The first part of this section will describe how to use the Windows-only RemoteLaunch® GUI application and the second part will describe how to use the cross-platform RemoteLaunchX™ console application.

1.1. Installing RemoteLaunch

It is required that you install the program before copying or installing any test extensions for the program. Testing applications, like Selenium and QuickTest Pro can be installed with no regards to the client application – if they are not installed by the time a test requiring them needs to be executed, the test extension will simply report an error or block for the specified test set.

There are no options to the installer except for installation path. If you do not use the default installation path (typically `C:\Program Files\Inflectra\Spira RemoteLaunch\`), then make a note of where the installation path is, because it will be needed to install test extensions later.

1.1.1 Installing a Test Extension

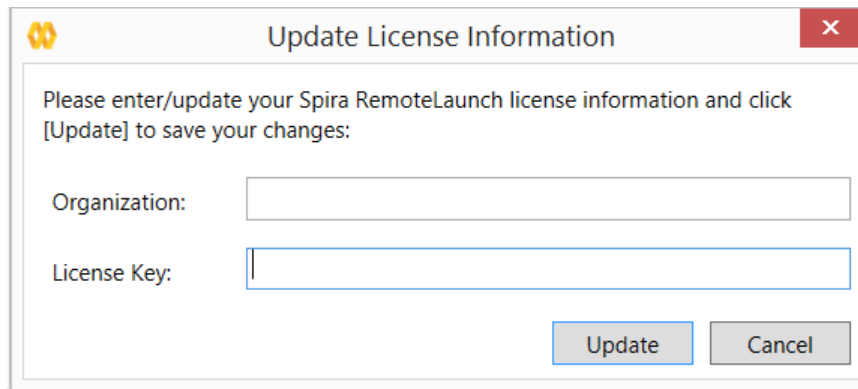
A test extension is a single or a set of DLLs that the program will read upon startup and provides a link in which testing applications (like TestComplete and Squish) to report test information and status back to SpiraTeam.

When you download a test extension, the ZIP file should contain at least one DLL file. Unless otherwise specified by a `readme.txt` file included in the compressed file, copy the DLL file to the `\extension` directory located within Spira RemoteLaunch installation directory. (If no such folder exists, you must create it.)

If an extension is removed or added, the program must be restarted for the any changes to take effect. The program will only load up to the first number of extensions that the license allows. Additional extensions will not be loaded or used during testing.

1.1.2 Registration

Spira RemoteLaunch has its own License key needed for using the program. You cannot use your existing SpiraTest/Plan/Team key in Spira RemoteLaunch. Upon the first launch of the program, you will be asked to update your license information:



Update License Information

Please enter/update your Spira RemoteLaunch license information and click [Update] to save your changes:

Organization:

License Key:

Update Cancel

Enter in your organization name and license key in the email that was sent when you purchased the license, or as listed on your customer information page at <http://inflectra.com>.

Trial licenses are good until the 28th day of the listed month. The next time the program is run after the 28th of the month, you will be prompted to re-enter a new permanent license key, or the program will be unusable.

The license key can be updated at any time by going to the Tray Menu and select Help -> About. Once the About screen opens up, click the Update button in the license details section to update or change license information.

1.2. Using RemoteLaunch

1.2.1 Basic Unattended Operation

When run, the program will start minimized to the system tray and will start its polling of the server. Polling will occur every 'x' minutes (60 by default) for any automated test sets that are scheduled to be run. When time comes for a test to be launched, it will start the test extension. The installed test extension will then perform the test and report results back to SpiraTeam. At the end of the test, the program will go back and resume scanning for tests that need to be executed.

No user input is ever needed from the application itself. However, testing applications may pop up dialogs needing user input. For existing Inflectra testing extensions, effort was put in to avoid as much user-interaction as possible, but in some cases it is unavoidable.

1.2.2 Client Configuration

By right clicking on the system tray icon and selecting "Configuration", the application's window will open to the configuration panel. The panel has the following options:

- SpiraTeam Server Configuration:

- **Server URL:** This is the URL of the SpiraTeam installation. Be sure to not put /Login.aspx or any other page in the string, this should be just the root URL of the application's install.
- **Login Username:** This is the SpiraTeam login id of the user that you want the tests reported as. Note that while the application is polling and updating test results, if the user is logged into a web browser session, they will get kicked out.
- **Login Password:** The password to the Username above.
- **Test:** Clicking this will test the login to make sure the application can connect to the server properly.

- Server Polling:

- **Automation Host Token:** This field is required, and uniquely identifies the local testing machine. Any scheduled tests assigned to the Automation Host on SpiraTeam will get polled for this machine. Except in special circumstances, this ID should be unique among all testing machines.



Important: This field must match the string that is entered into the Automation Host Details screen in the **Token:** field, or scheduled tests will not be recognized.

- **Automatically Run Overdue Tests:** When this is checked, any tests that are pulled from the SpiraTest server that has a scheduled date in the past will be marked as Overdue. Normally, overdue tests will not be executed. With this check, they will be executed as soon as the poll is finished.
- **Continue polling server after a connect error:** When this is checked, if RemoteLaunch receives an error connecting to the SpiraTest server, it will continue polling in the future. If this is unchecked, RemoteLaunch will switch to the error status upon encountering a

connection error. It is important to check this option if your SpiraTest server will be periodically unavailable for server maintenance.

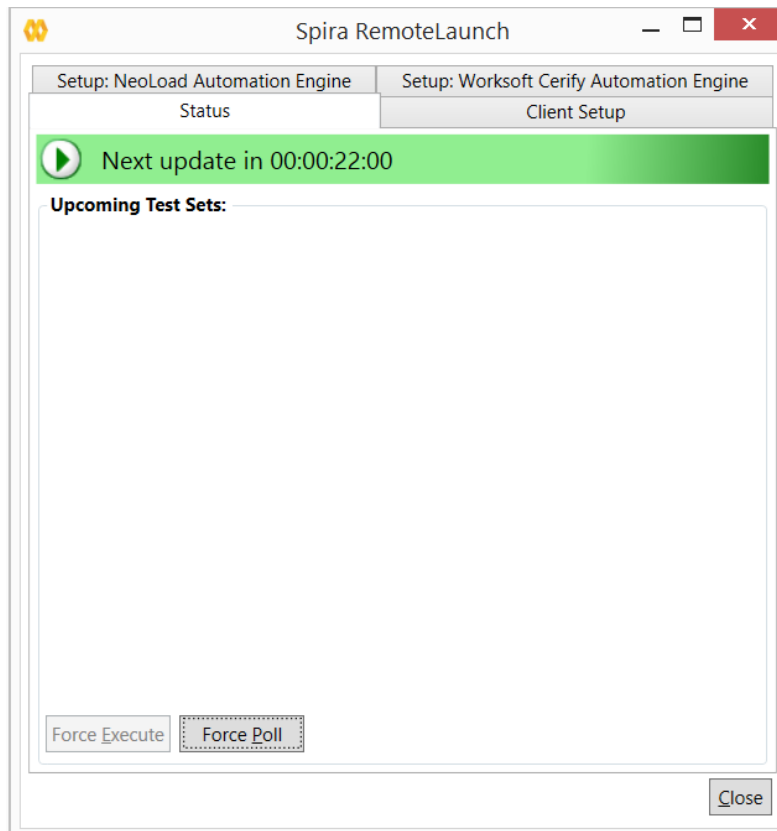
- **Polling Frequency:** How often in minutes the application will poll the SpiraTeam server for updates to the automation host's schedule. The default is 60 (1 hour), and should be fine for most installations. Note that tests will still be executed on their scheduled time, this is simply how often the program will talk to the SpiraTeam server to detect schedule changes. Updating the polling frequency will reset the currently running timers.
- **Polling Read Ahead:** How far ahead in minutes the program should read the schedule for the Automation host. Tests that are scheduled farther in advance will not show up as a pending test on the status screen.

1.2.3 Extension Configuration

If an extension has custom configuration options, they will appear as separate tabs located after the **Client Setup** tab. The contents of each tab will vary depending on the extension. View the extension's documentation for options given in those extensions.

1.2.4 Status Screen

The status screen is usually hidden, but can be brought up for display by double-clicking on the system tray icon. The top of the screen shows the current status, whether it's running a test or waiting to poll the server for an update. It will also show any errors present on the application, like a registration error or configuration issue. Under the status bar is a list of any pending or executing tests that are scheduled for this testing machine. The list will get cleared at every poll, so tests that have executed since the previous poll will still be on the list, and will show their execution status:



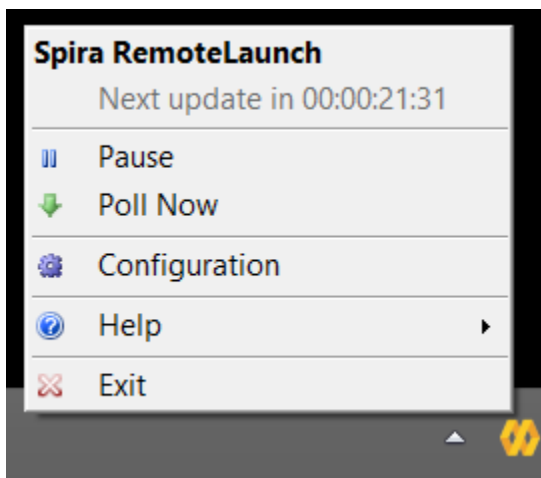
- **Green Arrow:** A green arrow indicates that the test is still running, or RemoteLaunch is waiting for a reply from the testing engine / test application.
- **Blue Checkbox:** A blue checkbox indicates that the test is completed, regardless of status of the individual test steps in the scheduled test set.
- **Red Error:** A red error indicator indicates that the test extension or the testing application ran into an issue (outside of test results). In this case, any further tests that require the extension will be marked as blocked, as the issue needs to be corrected within the extension settings or testing application.
- **No Indication:** No indication means that the test is currently awaiting for its scheduled date to start. Note that only one test will be launched at a time, so that if two tests are scheduled at the same time, the one with the lower TestSet ID will be executed first, then as soon as it's finished, the second scheduled test will be run.

By highlighting a test that has not been executed yet, you can click the Force Execute button. This will cause the selected test to have its scheduled date to the current time, causing it to be immediately executed (or, if another test is already running, next in line for execution).

At any time the Force Poll button can be clicked, causing RemoteLaunch to initiate an immediate poll of the SpiraTeam server to check for pending runs. The timers for the next server poll will be reset when the button is clicked.

1.2.5 Tray Icon Menu

Instead of operating from the application window, all functions exist on the tray icon menu as well, as well as some additional commands:



- **Pause / Resume:** The Pause/Resume option pauses or resumes the timers for polling and executing tests. If a test or server poll is already in progress, it will not cancel these. However, after they are finished, no further polls or tests will be run.
- **Poll Now:** This will force a server poll for upcoming tests, and reset the poll timer.
- **Configuration:** Opens the main window to the Configuration page.
- **Help -> About:** Opens the About window, which displays the current license information and any loaded extensions.
- **Help -> View Help:** Opens this PDF file in a browser.

- **Exit:** Will completely exit the program. Doing this will cancel any tests currently running and shut down the program. Any tests that were waiting to be executed will not execute until the program is restarted and the polling is resumed.

You can double-click the try icon to bring up the main window on the Status page.

1.3 Test Execution and Reporting




All test handling is performed by the extension that the automated tests are configured for. Test Sets that have multiple Test Cases, the Test Cases will all be executed in order, sequentially. (No parallel executing.)

At the start of execution for a Test Set, the test set will be updated in SpiraTeam as “In Progress”. As tests are performed, the Test Cases will be updated with their status. The Test Set on the status screen will be marked with the executing icon.

Once the Test Set is completed, the status of the Test Set will be changed to “Completed”, and will be marked on the status screen with a completed icon.

In case of an uncaught exception that is thrown by the testing extension, the Test Set will be marked “Blocked”, and the Test Case will be recorded as Blocked. All other following tests will not be run and remain as Not Run. The Test Set must be reset to be executed again, and it is recommended to look into the cause of the error (recorded in the Blocked Test Case results) and correct it before rescheduling the test. This Test Set will be marked with an error icon.

The same results are applied in the case where a Test Set contains a Test Case that references a testing extension that is not installed. Install the extension and re-run the Test Set.

Executing , Completed , and Error  Test Sets are marked with the icons next to their scheduled date in the Status screen. They will stay in the list until the next scheduled server poll. You cannot manually re-run them.

1.4. Running RemoteLaunch from a Build Script

Normally you schedule tests in SpiraTeam using the Planned Date field of the test sets and let the various instances of RemoteLaunch poll SpiraTeam for upcoming tests. In addition (as described in the *SpiraTeam User Manual*) you can execute a test set on the local machine immediately by clicking the “Execute” button within SpiraTeam.

However there are situations where you want to be able to launch an automated test script using one of the supported engines from an external batch file or build script (e.g. as part of a continuous integration environment) and have those tests report their results back into SpiraTeam. You can achieve this by using the special command-line argument `-testset` which is passed to RemoteLaunch. For more details on this parameter see the next section.

1.5 Command line arguments

For debugging and additional options when running the program, the following command-line arguments are available:

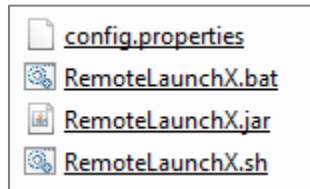
-status	Shows the Status screen upon startup. (Normal action is to run minimized to the system tray.)
-paused	Starts the application with timers Paused instead of active.
-poll	Forces the program to do an initial poll upon startup. (Normal action is to wait the pending

	time before doing the initial poll.)
-trace	Enables tracelogging to the EventLog for debugging and watching tests execute.
-logfile	Forces events to be written to a text file instead of the Application EventLog. This option enables -trace as well. Files are located in the Local Application Data folder. (C:\Users\ <user>\AppData\Local on Vista/Win7).</user>
-testset:[Test Set ID]	Allows you to tell RemoteLaunch to execute a specific test set on the remote computer (e.g. -testset:45 runs test set TX00045)
<filename>	Must be the last item on the command line. This is a TST file downloaded from SpiraTeam to start immediate execution on.

1.6. Using RemoteLaunchX

When you need to run automated tests on a variety of different platforms (Windows, MacOS X, Linux, Unix, etc.) the RemoteLaunchX cross-platform automated testing agent is a better choice than the standard RemoteLaunch® GUI application.

To start using RemoteLaunchX, please go to the Customer Area of the Inflectra website and download the latest version of the RemoteLaunchX application. It will be packaged as a simple .zip compressed folder that you can extract onto the target computer:



The following four files are included:

- **RemoteLaunchX.jar** – this is the main application, packaged as a Java JAR file. This version of RemoteLaunch requires Java 1.7 SE or later to be installed.
- **config.properties** – this contains all the settings used by RemoteLaunchX. You will need to edit this file in a text editor to configure RemoteLaunchX for use.
- **RemoteLaunchX.bat** – this is a sample Windows® batch file that can be used to simplify running RemoteLaunchX on Windows® systems.
- **RemoteLaunchX.sh** – this is a sample UNIX/Linux/MacOS X shell script that can be used to run RemoteLaunchX on UNIX, Linux or Mac OS X.

1.6.1. Configuring RemoteLaunchX

Once you have extracted the files listed above, open up the **config.properties** file in a text editor:

```
#This file contains the configuration data used by the RemoteLaunch-X
application

#Spira connection information
server-url = http://vm-win2012r2/SpiraTeam
server-login = fredbloggs
server-password = PleaseChange

#The automation host token
host-token = MyHost1

#The license key
license-organization: TBD
license-key: TBD

#The regular expressions for each of the possible execution statuses
pass-regex = .*
fail-regex = *(Error|Fail|Fatal).*
caution-regex = *(Warning|Caution).*
blocked-regex = *(Blocked).*
```

The following changes need to be made to this configuration file:

- **server-url** – This is the URL of the SpiraTest or SpiraTeam installation (hereafter referred to as just SpiraTest). Be sure to not put /Login.aspx or any other page in the string, this should be just the root URL of the application's install.
- **server-login** – This is the SpiraTest login id of the user that you want the tests reported as. Note that while the application is polling and updating test results, if the user is logged into a web browser session, they will get kicked out.
- **server-password** – The password of the SpiraTest login listed above.
- **host-token** – This field is required, and uniquely identifies the local testing machine. Any scheduled tests assigned to the Automation Host on SpiraTest will get polled for this machine. Except in special circumstances, this ID should be unique among all testing machines.



Important: This field must match the string that is entered into the Automation Host Details screen in the **Token:** field, or scheduled tests will not be recognized.

- **license-organization** – The name of the “Organization” that your RemoteLaunch license key was issued to. This is listed in the Customer Area of the Inflectra website alongside the license key. Note: RemoteLaunch and RemoteLaunchX use the same license keys, so you don't need to have a separate RemoteLaunchX one.
- **license-key** – The RemoteLaunch license key that is listed in the secure Customer Area of the Inflectra website

You should leave the four **regex** settings alone for now, they can be changed when you start executing tests and need to fine-tune how RemoteLaunchX interprets the results.

Now that you have configured the plugin, you can execute the RemoteLaunchX console application by either running the provided batch / shell command or just executing the JAR file directly:

```
Java -jar RemoteLaunchX.jar
```

When you run the application, the following should be output to the console:

```
Starting RemoteLaunch...
=====
Server URL: http://localhost/Spira
Server Login: fredbloggs
Automation Host: MyHost1
Checking License Key for: Inflectra Corporation
Production License Key in Use.
Testing connection to Spira...
Successfully connected to Spira.
WARNING: Unable to retrieve test runs for SpiraTest project PR2, so skipping
this project - Automation Host with token 'MyHost1' doesn't exist in project
PR2.
WARNING: Unable to retrieve test runs for SpiraTest project PR3, so skipping
this project - Automation Host with token 'MyHost1' doesn't exist in project
PR3.
Retrieved 0 test run(s) from SpiraTest.
Exiting RemoteLaunch...
=====
```

The system will report back zero Test Runs at this point because nothing has been scheduled in SpiraTest. In the next section we shall setup an automated test set that contains an automated test case.

1.6.2. Setting up Automated Tests in SpiraTest

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunchX on the various test automation hosts following the instructions above. Once those prerequisites are in place, please follow these steps:

- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the “Test Automation” link under **Integration**.
- Click the “Add” button to enter the new test automation engine details page. The fields required are as follows:

Command-Line Edit Automation Engine

← Back to Test Automation Engine Home

Please enter/edit the following information for the test automation engine.

Name*: Command-Line

Description: Engine that will execute a generic command-line executable

Token*: CommandLine

Active: Yes

Save Cancel

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For Command-Line this should be simply “**CommandLine**”.

Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Command-Line listed as an available automation engine.

Next you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

Automation

This section defines the automated test script associated with this test case:

Script Type*: Attached Linked Repository

Automation Engine: Command-Line Version: 1.0 Document Type: Functional Specification

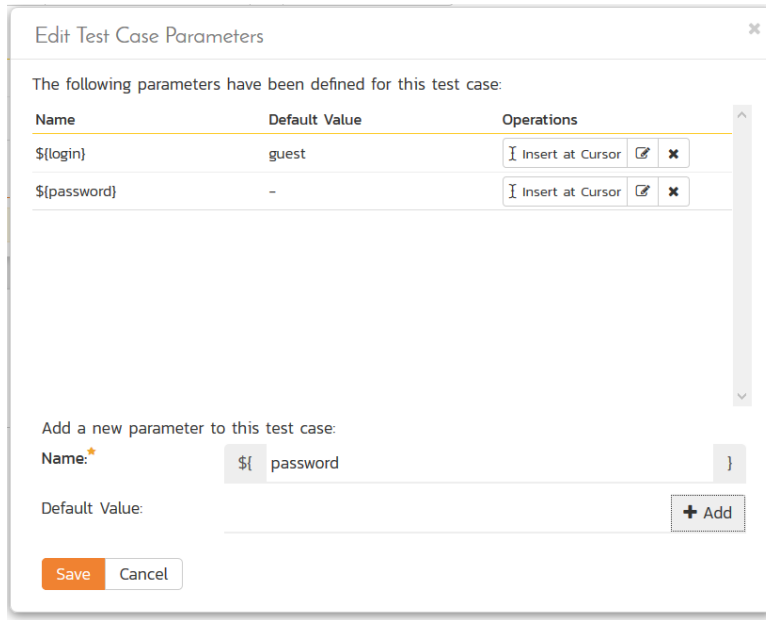
Filename: C:\Windows\System32\ipconfig. ... Document Folder: Root Folder

Test Script:

You need to enter the following fields:

- **Automation Engine** - Choose the Command-Line Automation Engine that you created in the previous section from the drop-down list.
 - **Script Type** – This can be set to Attached or Linked (see below for the difference).
 - **Filename** – This needs to consist of the following **three** sections separated by a pipe (|) character:
 1. The full path to the command-line tool being executed.
 2. Any arguments for the command-line tool. In addition, you can use the following additional tokens for some of the special RemoteLaunchX values:
 - [TestCaseId] – the ID of the test case
 - [TestSetId] – the ID of the test set
 - [ReleaseId] – the ID of the release (if specified)
 - [Filename] - This special token will be replaced by the actual filename of the test script when RemoteLaunchX downloads it from SpiraTeam.
 3. The mask for converting any parameter values from SpiraTeam into valid command line arguments. If parameters are not accepted by the command-line tool, you can leave this section out.
 - The mask can include any symbols together with “name” to refer to the parameter name and “value” to refer to the parameter value.
 - Example 1: If you want parameters to be provided in the form:
-param1=value1 -param2=value2
you would use the following mask:
-name=value
 - Example 2: If you want parameters to be provided in the form:
/param1:value1 /param2:value2
you would use the following mask:
/name:value
- ▷ Some example filenames would be:
C:\Temp\TestApp.exe|-arg1 -arg2|-name=value
C:\Temp\TestApp.exe|-arg1 -arg2 "-arg3=[Filename]"|
- where the first one is for a **Linked** test and the second one is for an **Attached** test.
- **Document Type** – You can choose which document type the automated test script will be categorized under.
 - **Document Folder** – You can choose which document folder the automated test script will be stored in.
 - **Version** – The version of the test script (1.0 is used if no value specified)
 - **Test Script** – For **Attached** test scripts, this needs to contain the complete test script in whatever language and syntax is being expected by the command-line application. For **Linked** test scripts, you should leave this blank.
 - ▷ If you would like to have SpiraTeam pass any parameter values to this test script you can specify them by using the syntax `${parameterName}` **inside the test script**.

- here is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your command-line automated testing tool. This is very useful if you want to have a data-driven test script that be executed multiple times with different parameter values.
- To setup the automated test case for parameters, click on the “Edit Parameters” hyperlink above the “Test Script” box:



- The name of the parameter `${login}` needs to match the name of a parameter accepted by the command-line tool.

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows_7_Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunchX application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case. Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Command-Line automated test cases and click on its hyperlink to display the test set details page:

Testing Cycle for Release 1.0
 Test Set [TX:1] Status: * Completed Execution Status: ■ ■

People

Owner: -- None --
 Creator: * Fred Bloggs

Properties

Release: 10.0.0 - Library System...
 Type: * Manual
 Automation Host: -- None --
 Configuration: -- None --
 Operating System: --- Please Select ---

Dates and Times

Creation Date: 1/1/2007 7:00:00 PM
 Last Executed: 12/1/2003 6:30:55 AM
 Last Updated: 11/8/2017 10:52:55 AM
 Planned Date:
 Recurrence: -- One Time --
 Schedule on Build:
 Post-Build Delay (s):

[Detailed Information](#)

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameters

The following parameter values have been specified at the test set level:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
\$(login)	librarian	-	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
\$(password)	librarian	-	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Cases

-- Show/Hide columns -- Est. Dur. 0.7h / Act. Dur. 3.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
<input type="checkbox"/>	Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
<input type="checkbox"/>	Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
<input type="checkbox"/>	Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
<input type="checkbox"/>	Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
<input type="checkbox"/>	Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
<input type="checkbox"/>	Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
<input type="checkbox"/>	Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

Show 15 rows per page Displaying page 1 of 1

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

The screenshot shows a web interface for a Test Configuration Set [TG-4]. It has tabs for Overview and Test Sets. Below the tabs are sections for Properties, Dates and Times, Detailed Information, and Test Configuration Entries. The Test Configuration Entries section includes buttons for Populate, Remove, and Refresh. A message indicates that 10 out of 10 entries are displayed. Below this is a table with columns for a checkbox, a login name, a browser name, and an ID.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

1.6.3. Running RemoteLaunchX

Once you have set the various test set fields (as described above), you are now ready to execute RemoteLaunchX. You can execute the RemoteLaunchX console application by either running the provided batch / shell command or just executing the JAR file directly:

```
Java -jar RemoteLaunchX.jar
```

When you run the application, the following should be output to the console:

```
Starting RemoteLaunch...
=====
Server URL: http://localhost/Spira
Server Login: fredbloggs
Automation Host: MyHost1
Checking License Key for: Inflectra Corporation
Production License Key in Use.
Testing connection to Spira...
Successfully connected to Spira.
WARNING: Unable to retrieve test runs for SpiraTest project PR2, so skipping
this project - Automation Host with token 'MyHost1' doesn't exist in projec
PR2.
WARNING: Unable to retrieve test runs for SpiraTest project PR3, so skipping
this project - Automation Host with token 'MyHost1' doesn't exist in projec
PR3.
Retrieved 1 test run(s) from SpiraTest.
Executing test case TC18 with filename 'C:\Windows\System32\ipconfig.exe|/all'
This is a Linked test script
Executing command 'C:\Windows\System32\ipconfig.exe' with arguments '/all'
Execution Status = Passed
Exiting RemoteLaunch...
=====
```

The console output will indicate which test sets are being executed and what the final result was. Inside SpiraTest, once execution begins the status of the test set will change from “Not Started” to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed (passed or failed) – or “Blocked” – RemoteLaunchX was not able to execute the test.

In addition, the individual test cases in the set will display a status based on the results of the command-line test that was executed:

- **Passed** – The automated test ran successfully and matched the PASS regular expression.
- **Failed** – The automated test ran successfully, and matched the FAIL regular expression.
- **Caution** – The automated test ran successfully, and matched the CAUTION regular expression.
- **Blocked** – The automated test did not run successfully or it matched the BLOCKED regular expression.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Test Run that was recorded and the Console output section will contain the underlying error message(s).

Once the tests have completed, you can log back into SpiraTest and see the execution status of your test cases. If you click on a Test Run that was generated by the command-line tool, you will see the following information:

The screenshot displays the SpiraTest interface for a test run. The 'Details' section shows the following information:

- Release #: 1.0.1.0 - Library System Release 1 SP1
- Tester Name: Fred Bloggs
- Test Set: RemoteLaunch X Test Set
- Test Case #: TC000018
- Build: -- None --
- Web Browser: -- Please Select --
- Notes: (Rich text editor)
- Estimated Duration: [] hours
- Actual Duration: 0.00 hours
- Execution Date: 12/19/2013 12:14:01 PM
- Execution Status: **Passed** (indicated by a green bar)
- Test Run Type: Automated
- Operating System: -- Please Select --

The 'Test Steps' section contains a table with the following columns: ID, Test Step Description, Expected Result, Sample Data, Test # / Step #, Actual Result, and Execution Status.

The 'Console Output' section shows the following details:

```

Runner Name: RemoteLaunch-X          Assert Count: 0
Automation Host:                      Test Name: C:\Windows\System32\ipconfig.exe/all
Message: Windows IP Configuration Host Name . . . . .

Details:
Standard Out
=====

Windows IP Configuration

Host Name . . . . . : TARDIS
Primary Dns Suffix . . . . . : corp.inflectra.com
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : corp.inflectra.com
  
```

This screen indicates the status of the test run that was reported back from command-line tool together with any messages or other information. The execution status will be set according to the rules described above, the Message field will contain the first line of console output and the large details box will contain the full console output from the command-line tool.

Congratulations... You are now able to run a custom command-line test, and have the results be recorded within SpiraTest / SpiraTeam.

1.6.4. Scheduling RemoteLaunchX

Unlike the main RemoteLaunch application, RemoteLaunchX does not have a built-in timer and so when executed it will run once, check for pending test sets and then exit. If you want to have it run on a periodic basis, you will need to schedule it externally. If you are using Microsoft Windows® you would use the Windows Task Scheduler and in other operating systems you would setup a CRON job. We recommend scheduling RemoteLaunchX to run every 5 minutes.

1.6.5. Customizing the Reporting

By default, RemoteLaunchX will use the following rules to determine if a test has passed, failed, blocked or passed with warnings (caution):

- **Passed** – The test completed and the console output didn't contain any of the error phrases listed in the other rules (below).
- **Failed** – The test completed and the console output contained the phrases "Error", "Fail" or "Fatal".
- **Caution** – The test completed and the console output contained the phrases "Warning", or "Caution".
- **Blocked** – The automated test did not run successfully or the console output contained the phrase "Blocked".

You can customize the reporting by changing the Regular Expressions (Regex) stored in the config.properties files:

```
#The regular expressions for each of the possible execution statuses
pass-regex = .*
fail-regex = .*(Error|Fail|Fatal).*
caution-regex = .*(Warning|Caution).*
blocked-regex = .*(Blocked).*
```

2. HP UFT / QTP

HP® QuickTest Professional® (hereafter QTP) is an automated functional test automation system that lets you record application operations and generate VBA test automation scripts that can be used to playback the test script against the test application.

HP® Unified Functional Testing® (hereafter UFT) is an updated version of QTP that also includes functionality for web service testing.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of QTP and UFT on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated QTP and UFT tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 9.0 of Quick Test Professional. For accessing UFT, you'd need at least version 4.0 of SpiraTest/Team and version 11.0 of UFT.

2.1. Installing the UFT/QTP Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [QuickTestProAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate QuickTestProX.dll or UftX.dll for the version of QTP or UFT that you are using.
 - If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "QuickTestProX.dll" or "UftX.dll" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

The screenshot shows a web form titled "Quick Test Pro Edit Automation Engine". At the top left, there is a button labeled "← Back to Test Automation Engine Home". Below this, a message reads: "Please enter/edit the following information for the test automation engine." The form contains the following fields:

- Name:** A text input field containing "Quick Test Pro".
- Description:** A larger text area containing "Engine that integrates with HP Quick Test Pro 9.0+".
- Token:** A text input field containing "QuickTestPro9".
- Active:** A radio button labeled "Yes" which is selected.

At the bottom of the form are two buttons: "Save" (in red) and "Cancel".

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.

- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
 - **Active:** If checked, the engine is active and able to be used for any project.
 - **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case.
 - For QTP this should be **QuickTestProX** where 'X' is the version number of the DLL file that you are using.
 - For UFT this should be **UftX** where 'X' is the version number of the DLL file that you are using.
- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with QTP listed as an available automation engine.

2.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated QTP or UFT test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Overview” tab and scroll down to the ‘Automation’ section:

The screenshot shows a web form for configuring automation. At the top, there's a dropdown menu for 'Automation' and a help icon. Below that, a yellow banner states: 'This section defines the automated test script associated with this test case:'. The form fields are:

- Automation Engine:** Quick Test Pro (dropdown)
- Document Type:** Functional Specification (dropdown)
- Document Folder:** Test Results (dropdown)
- Script Type:** Attached, Linked (selected), Repository (radio buttons)
- Version:** 1.0 (dropdown)
- Filename:** file:///c:/users/fredblogs/appdata/local/test_scripts/test_1 (text input)
- Test Script:** (empty text area)

 An '</> Edit Parameters' button is located at the bottom right of the form.

You need to enter the following fields:

- **Automation Engine** - Choose the QTP/UFT Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with QTP/UFT only supports referencing QTP/UFT test script folder paths and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This needs to be the full path to the QTP/UFT test script folder (i.e. the folder that you open in QTP/UFT to run the test). To make this easier across different machines, you can use several constants for standard Windows locations (see example in screenshot):
 - ▷ [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [CommonDocuments] – The Public Document’s folder.
 - ▷ [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.

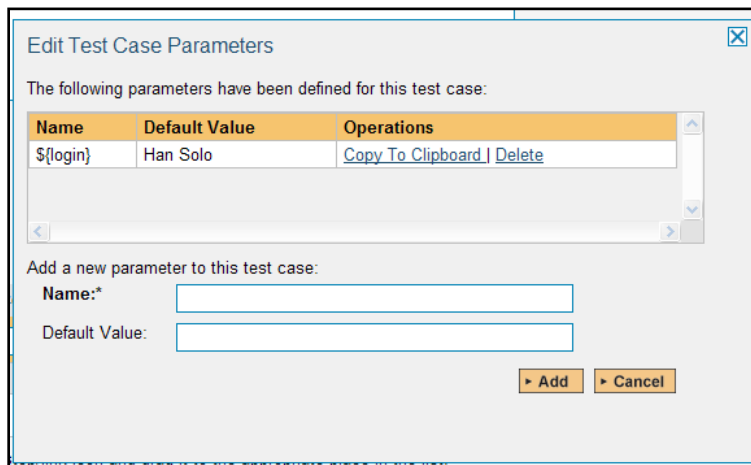
- ▷ [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it's the 64-bit directory.
- ▷ [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the QTP/UFT Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

2.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your QTP/UFT automated test script. This is very useful if you have a data-driven QTP/UFT test script that accepts input parameters from an external data source.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter \${login} needs to match the name of the input parameter defined within the QTP/UFT script in its input parameters configuration.

2.3. Executing the QTP/UFT Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

2.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

+ New Host ✕ Delete ↻ Refresh ⌵ Filter -- Show/Hide columns --

Displaying 1 - 5 out of 5 automation host(s) for this project.

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
<input type="checkbox"/> Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
<input type="checkbox"/> Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
<input type="checkbox"/> Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
<input type="checkbox"/> Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
<input type="checkbox"/> TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Show 15 rows per page ⏪ ⏩ Displaying page 1 of 1 ⏪ ⏩

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Displaying 1 - 3 out of 7 test set(s) for this project. Regression Test Sets --- All Releases ---

Name	Execution Status	Planned Date	Release	Last Executed	Owner	Status	ID	Edit
<input type="checkbox"/> [Empty]	---		---		---	---	TX	Edit
<input type="checkbox"/> QTP Testing						Not Started	TX:39	Edit
<input type="checkbox"/> Regression Testing for Windows 8 4					Fred Bloggs	Completed	TX:3	Edit
<input type="checkbox"/> Regression Testing for Windows Vista 4						Completed	TX:4	Edit

Show 15 rows per page ⏪ ⏩ Displaying page 1 of 1 ⏪ ⏩

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the QTP/UFT automated test cases and click on its hyperlink to display the test set details page.

Overview Test Runs Attachments Incidents History *

Details

Owner:	-- None --	Creator:	System Administrator	Release:	-- None --
Type:	Automated	Automation Host:	Windows 7 Host	Creation Date:	4/8/2016 11:57:59 AM
Status:	Not Started	Last Executed:	-	Planned Date:	04/12/2016 8:00 am
Recurrence:	-- One Time --	Execution Status:		Last Updated:	4/8/2016 12:02:29 PM

You need to add at least one automated test case to the test set:

▼ Test Cases

Est. Dur. - / Act. Dur. -

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
<input type="checkbox"/>	Book management		2 - High	0.1h			Not Run	TC:8	Edit

Show 15 rows per page

Displaying page 1 of 1

To change the order of the test cases, please click on the test case icon and drag it to the appropriate place in the list.

Then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameters

The following parameter values have been specified at the test set level:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	<input type="button" value="✎ Edit"/> <input type="button" value="✕ Delete"/>
\$(login)	librarian	-	<input type="button" value="✎ Edit"/> <input type="button" value="✕ Delete"/>
\$(password)	librarian	-	<input type="button" value="✎ Edit"/> <input type="button" value="✕ Delete"/>

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Cases

Est. Dur. 0.7h / Act. Dur. 3.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
<input type="checkbox"/>	Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
<input type="checkbox"/>	Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
<input type="checkbox"/>	Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
<input type="checkbox"/>	Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
<input type="checkbox"/>	Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
<input type="checkbox"/>	Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
<input type="checkbox"/>	Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

Show 15 rows per page

Displaying page 1 of 1

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Dialog box titled "Edit Test Case Parameters" with a close button (X). A yellow message box says "Please fill out the parameters for this test case entry:". Below are input fields for browserName, login (admin), operatingSystem, password (admin), and url. At the bottom are "Save" and "Cancel" buttons.

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview Test Sets #

Properties + Dates and Times +

Detailed Information +

Test Configuration Entries

+ Populate x Remove Refresh

Displaying 10 out of 10 entries in this test configuration set.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

2.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

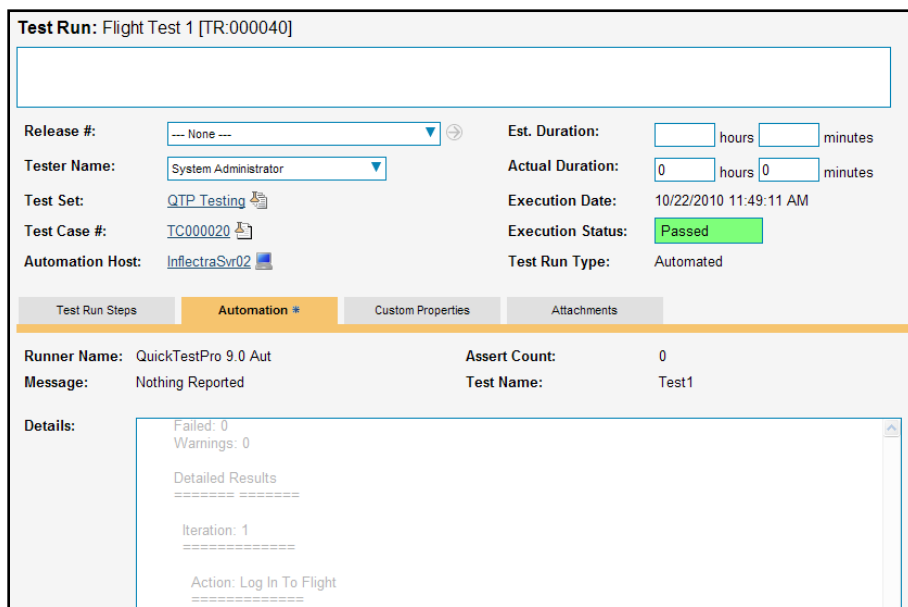
In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the QTP/UFT test:

- **Passed** – The QTP/UFT automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The QTP/UFT automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The QTP/UFT automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser or application windows launch as QuickTest Pro (QTP) or Unified Functional Testing (UFT) executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by QTP/UFT, you will see the following information:



This screen indicates the status of the test run that was reported back from QTP/UFT together with any messages or other information. The Test Name indicates the name of the test inside QTP/UFT, and the execution status corresponds the matching status inside QTP/UFT as illustrated below:

QTP/UFT Status	SpiraTeam Status
Passed	Passed
Failed	Failed
Warning	Caution
Stopped	Blocked
Not Applicable	N/A
(Any other status)	Not Run

In addition, the detailed test report from QTP/UFT is available in the large text-box below. It will contain messages such as:

```
QuickTest Professional
  Test: Test1
  Results Name: Res21
  Run Started: 10/22/2010 - 11:49:06
  Run Ended: 10/22/2010 - 11:49:10

  Summary Results
  =====
  Passed: 0
  Failed: 0
  Warnings: 0

  Detailed Results
  =====

  Iteration: 1
  =====

  Action: Log In To Flight
  =====

  Step: Login: Dialog
  Step: Agent Name:.SetText: "Bobba Fett"
  Step: Agent Name:.Type: "&lt;_MicTab&gt;"
  Step: Password:.SetSecureText:
"4cc08e88683135b35bb8a7dab8442c69b8441f3e"
  Step: OK.Click:
  Step: Flight Reservations: Dialog
  Step: OK.Click:
```

Congratulations... You are now able to run QTP/UFT automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

3. SmarteScript

SmarteSoft™ SmarteScript™ (hereafter SmarteScript) is a Graphic User Interface (GUI) script-free functional test automation system that lets you record application operations by capturing the various testable objects of the application and then playback the operations to automatically test the application.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of SmarteScript on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated SmarteScript tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 5.0 of SmarteScript.

3.1. Installing the SmarteScript Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- ▶ Download and extract the [SmarteScriptAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate SmarteScriptX.dll for the version of SmarteScript that you are using.
 - ▷ If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "SmarteScriptX.dll" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | New Engine
[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

Active

- **Name**: This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description**: This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active**: If checked, the engine is active and able to be used for any project.
- **Token**: This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For

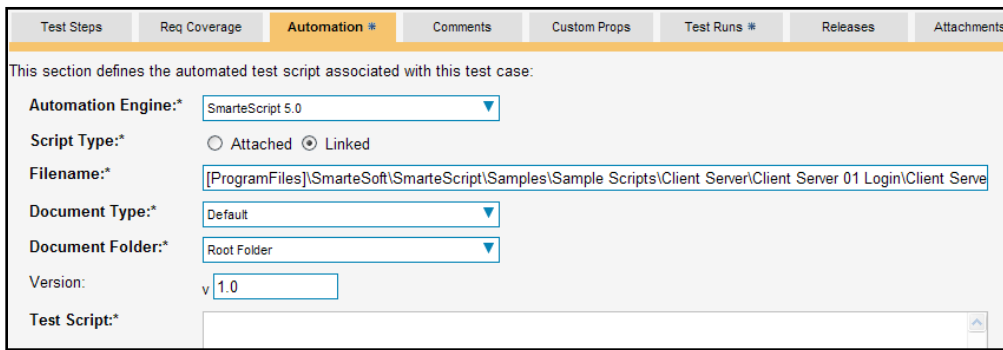
SmarteScript this should be **SmarteScriptX** where 'X' is the version number of the DLL file that you are using.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with SmarteScript listed as an available automation engine.

3.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated SmarteScript test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:



The screenshot shows the 'Automation' tab in SpiraTeam. The form is titled 'This section defines the automated test script associated with this test case:'. It contains the following fields:

- Automation Engine:** A dropdown menu with 'SmarteScript 5.0' selected.
- Script Type:** Radio buttons for 'Attached' and 'Linked', with 'Linked' selected.
- Filename:** A text box containing the path: `[ProgramFiles]\SmarteSoft\SmarteScript\Samples\Sample Scripts\Client Server\Client Server 01 Login\Client Serve`.
- Document Type:** A dropdown menu with 'Default' selected.
- Document Folder:** A dropdown menu with 'Root Folder' selected.
- Version:** A text box with 'v 1.0' entered.
- Test Script:** An empty text box.

You need to enter the following fields:

- **Automation Engine** - Choose the SmarteScript Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with SmarteScript only supports referencing SmarteScript test script file (.ses) and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This needs to be the full path to the SmarteScript test script (i.e. the .ses file that you open in SmarteScript to run the test). To make this easier across different machines, you can use several constants for standard Windows locations (see example in screenshot):
 - ▷ [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [CommonDocuments] – The Public Document’s folder.
 - ▷ [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
 - ▷ [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)

- **Test Script** – This is not used with the SmarteScript Engine since it only supports linked test scripts.

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

3.2.1. Using Parameterized Test Cases

SmarteScript does not support the passing of input test parameters so the SmarteScript automation engine does not support this feature of SpiraTeam or RemoteLaunch.

3.3. Executing the SmarteScript Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

3.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
TC 7.0 Testing (1)	In Progress	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	Completed	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmarteScript Testing (1)	Completed	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	Completed	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	Completed	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	Completed	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the SmarteScript automated test cases and click on its hyperlink to display the test set details page:

Test Set: SmarteScript Testing [TX:000012]

Name: SmarteScript Testing

Description:
 -- Font -- -- Size -- **B** **I** **U** **Text** **Align** **Color** **Background** **Link** **Image** **Table** **Code** **Print** **Help**

Owner: -- None -- Creator: System Administrator

Release: --- None --- Type: Automated

Automation Host: InflectraSvr02 Created On: 10/26/2010 11:55:47 AM

Status: Completed Last Executed: -

Planned Date: 10/26/2010 05:10:00 PM Last Updated: 11/3/2010 4:26:59 PM

Test Cases # Test Runs # Comments Custom Props Attachments History #

> Add Tests Remove Tests Refresh Edit Parameters Execute Tests Est. Duration: 0.0h / Actual Duration: 0.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Duration	Act. Duration	Last Executed	Execution Status	Test Case #	Edit
<input type="checkbox"/>	Login to SmarteATM				0.0h	26-Oct-2010	Failed	TC000021	Edit

Show 15 rows per page Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

3.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the SmarteScript test:

- **Passed** – The SmarteScript automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The SmarteScript automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The SmarteScript automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser or application windows launch as SmarteScript executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by SmarteScript, you will see the following information:

Test Run: Login to SmarteATM [TR:000043]

Release #: **Est. Duration:** hours minutes

Tester Name: **Actual Duration:** hours minutes

Test Set: [SmarteScript Testing](#) **Execution Date:** 10/26/2010 5:06:57 PM

Test Case #: [TC000021](#) **Execution Status:** Failed

Automation Host: [InflectraSvr02](#) **Test Run Type:** Automated

Automation *

Runner Name: SmarteScript 5.0 Aut **Assert Count:** 1

Message: Success Rate: 66331752% **Test Name:** Client Server 01 Login

This screen indicates the status of the test run that was reported back from SmarteScript together with any messages or other information. The Test Name indicates the name of the test inside SmarteScript, and the execution status corresponds the matching status inside SmarteScript.

Congratulations... You are now able to run SmarteScript automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

4. TestComplete

SmarteBear™ TestComplete™ (hereafter TestComplete) is an automated functional test automation system that lets you record application operations and generate test automation scripts in a variety of languages (JavaScript, C#, VBScript). These test scripts can then be used to playback the test script against the test application and verify that it works correctly.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of TestComplete on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated TestComplete tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 8.0 of TestComplete.

4.1. Installing the TestComplete Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- ▶ Download and extract the [TestCompleteAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate TestCompleteX.dll or TestExecuteX.dll for the version of TestComplete or TestExecute that you are using.
 - ▷ If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "TestCompleteX.dll" or "TestExecuteX.dll" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | TestComplete

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

Active

- **Name**: This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description**: This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active**: If checked, the engine is active and able to be used for any project.

- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For TestComplete this should be **TestCompleteX** where 'X' is the version number of the DLL file that you are using. For TestExecute this should be **TestExecuteX** where 'X' is the version number of the DLL file that you are using.

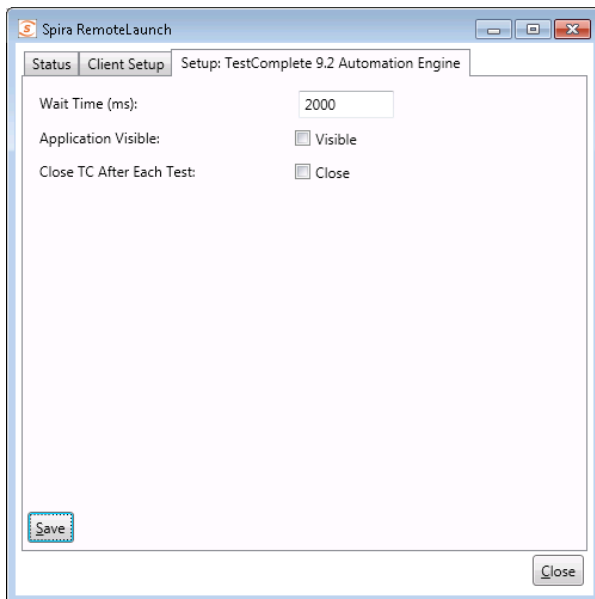
Note: We only use the major version numbers for the token name. So the DLLs TestComplete9.0.dll and TestComplete9.1.dll would both use Token = "TestComplete9".

- Once you have finished, click the "Insert & Close" button and you will be taken back to the Test Automation list page, with TestComplete listed as an available automation engine.

4.1.1. Advanced Settings

You can modify the TestComplete configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing "Configuration". That will bring up the RemoteLaunch configuration page.

The TestComplete engine adds its own tab to this page which allows you to configure how TestComplete operates:



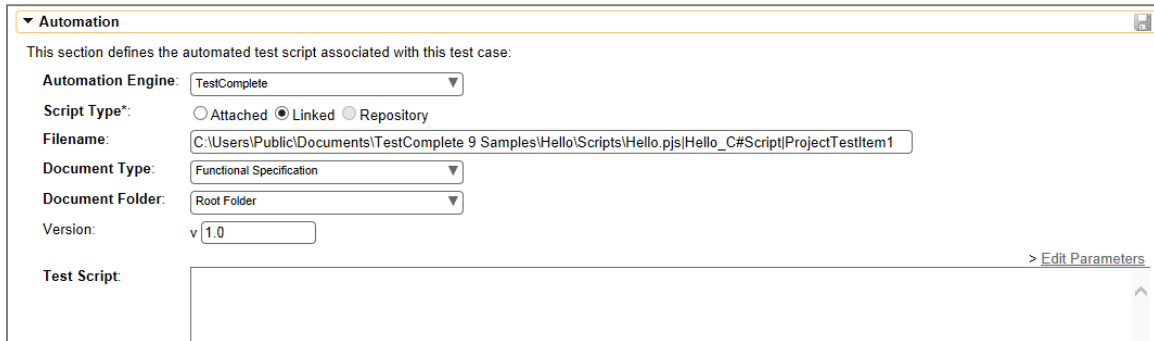
The following fields can be specified on this screen:

- **Wait Time** – This should be set to the amount of time TestComplete needs on this workstation to close the currently open test. The default value is 10000ms (10 seconds). If you get error messages that TestComplete is still open, you need to increase this value.
- **Application Visible** – This allows you to configure whether the TestComplete application is displayed during test execution or is kept hidden. The default is for it to be hidden.
- **Close TC After Each Test** – When this is selected, the plugin will close the TestComplete application after each test executes. We generally recommend leaving this disabled as the startup and closedown of TestComplete can sometimes interfere with the tests being executed.

4.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated TestComplete test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:



You need to enter the following fields:

- **Automation Engine** - Choose the TestComplete Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with TestComplete only supports referencing TestComplete test project/suite file paths and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This is actually a compound of several different components that need to be entered, separated by the pipe (|) symbol. The syntax depends on whether we want to associate the SpiraTeam test case with a specific *project item* or with a specific *test routine*. If you want to use parameterized test cases, you need to link it with a specific routine (see below for more details on parameters).
 - ▷ If you want to execute a specific **project item**, the filename should consist of
Suite Filename|Project Name|Project Item Name
(e.g. [CommonDocuments]\TestComplete 7 Samples\Open Apps\OrdersDemo\C#\TCProject\Orders.pjs|Orders_C#_C#Script|ProjectTestItem1)
 - ▷ If you want to execute a specific **test routine**, the filename should consist of
Suite Filename|Project Name|Unit Name|Routine Name
(e.g. [CommonDocuments]\TestComplete 7 Samples\Scripts\Hello\Hello.pjs|Hello_C#Script|hello_cs|Hello)
 - ▷ In the case of executing a specific test routine, the last component (the *routine name*) is actually the name of the function in the test script itself.
 - ▷ As illustrated in the examples, for the Test Suite filename, you can use several constants for standard Windows locations to make things easier:
 - ▷ [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [CommonDocuments] – The Public Document’s folder.

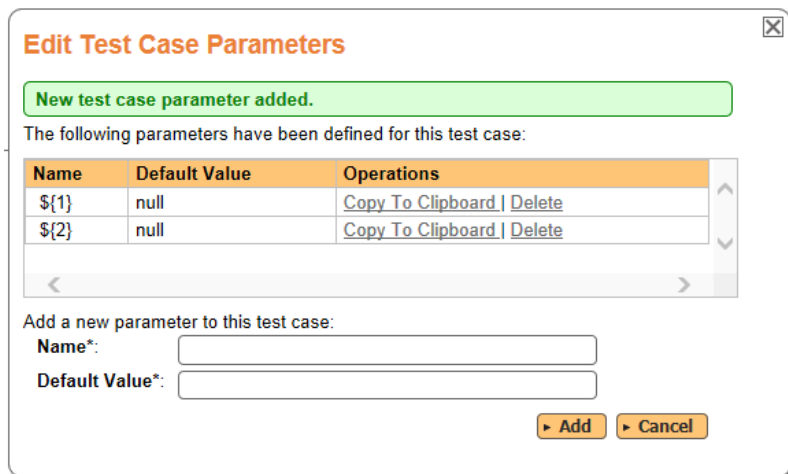
- ▷ [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
- ▷ [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
- ▷ [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the TestComplete Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

4.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your TestComplete automated test script. This is very useful if you have a data-driven TestComplete test script that accepts input parameters. To use this feature you need to use the option described above to link the SpiraTest test case to an explicit test routine inside TestComplete. If you choose the option to link to a Project Item, any parameters passed will be ignored.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



Since the parameters in SpiraTeam map to the function arguments inside a TestComplete test script the parameter names need to match the order of the arguments inside TestComplete (i.e. they are matched by position/order not by name).

Therefore we recommend using numbers for the parameter names so that it’s easy to see which parameter value will be passed to which argument in the test script function.

4.3. Executing the TestComplete Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

3. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
4. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

4.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows_7_Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
TestComplete Testing (1)	In Progress	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	Completed	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmarteScript Testing (1)	Completed	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	Completed	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	Completed	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	Completed	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the TestComplete automated test cases and click on its hyperlink to display the test set details page:

Test Set: TestComplete Demo Set [TX:000010]

Name: TestComplete Demo Set

Overview # Test Runs # Attachments Incidents History #

Details

Owner: -- None -- Creator*: System Administrator

Release: 1.0.0.0 - Library System Release 1 Type: Automated

Automation Host: Windows 7 Host Creation Date: 4/22/2013 4:41:22 PM

Status*: Not Started Last Executed: 4/22/2013 10:08:51 PM

Planned Date: 04/22/2013 4:30 pm -- One Time -- Last Updated: 4/22/2013 10:08:55 PM

Notes: -- Font -- -- Size -- B I U [List Icons]

Operating System: --- Please Select ---

Description

Comments

Test Cases

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Dur.: 0.00 / Act. Dur.: 0.00

<input type="checkbox"/>	<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
<input type="checkbox"/>	<input type="checkbox"/>	TestComplete Tests				0.0h	22-Apr-2013	Passed	TC000018	Edit

Show 15 rows per page Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameters

The following parameter values have been specified at the test set level: [+ Add Parameter Value](#)

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	Edit Delete
\$(login)	librarian	-	Edit Delete
\$(password)	librarian	-	Edit Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	C-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login:

operatingSystem: _____

password:

url: _____

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview | Test Sets *

Properties + | Dates and Times +

Detailed Information +

Test Configuration Entries

Displaying 10 out of 10 entries in this test configuration set.

login	browserName	ID
admin	Internet Explorer	22
admin	Mozilla / Firefox	23
admin	Safari	25
borrower	Internet Explorer	26
borrower	Mozilla / Firefox	27
borrower	Opera	28

4.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the TestComplete test:

- **Passed** – The TestComplete automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The TestComplete automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The TestComplete automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser or application windows launch as TestComplete executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by TestComplete, you will see the following information:

The screenshot displays the 'Test Run' details for 'TestComplete Tests [TR:000023]'. The interface includes tabs for 'Overview', 'Attachments', and 'Incidents'. The 'Details' section is expanded, showing the following information:

Release #:	1.0.0.0 - Library System Release 1	Estimated Duration:	0.00 hours
Tester Name*:	System Administrator	Actual Duration:	0.00 hours
Test Set:	TestComplete Demo Set	Execution Date:	4/22/2013 10:08:51 PM
Test Case #*:	TC000018	Execution Status*:	Passed
Build:	-- None --	Test Run Type*:	Automated
Web Browser:	-- Please Select --	Operating System:	-- Please Select --

Below the details is a 'Notes' section with a rich text editor toolbar.

This screen indicates the status of the test run that was reported back from TestComplete together with any messages or other information. The Test Name indicates the name of the test inside TestComplete, and the execution status corresponds to the matching status inside TestComplete as illustrated below:

TestComplete Status	SpiraTest Status
Passed	Passed
Failed	Failed
Warning	Caution

In addition, the detailed test report from TestComplete is available in the “Console Output” text-box below. It will contain messages such as:

▼ Console Output

Runner Name: TestComplete 9.2 Aut Assert Count: 0
Automation Host: Windows 7 Host Test Name: Hello_C#Script > ProjectTestItem1
Message: 0 errors and 0 warnings.

Details:

```
> Opening up logfile: C:\Users\Public\Documents\TestComplete 9 Samples\Hello\Scripts\C#Script\Log\4_22_2013_10_08 PM_16_804\{B2AB0825-A827-496F-A8CF-5D4D0A7E606D}
Other: The application "C:\Windows\system32\mspaint.exe" started.
Action: The 'Untitled - Paint' window was maximized.
Action: The window was clicked with the left mouse button.
Action: The window was clicked with the left mouse button.
Action: The window was clicked with the left mouse button.
Action: The window was clicked with the left mouse button.
Other: The radio button is already checked.
Other: The radio button is already checked.
Action: The button was clicked with the left mouse button.
```

For the most detail, the “Test Steps” section will contain a step-by-step breakdown of each action taken in the automated test:

▼ Test Steps

ID	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
RS000149	The application "C:\Windows\system32\mspaint.exe" started.			/	The process ID is 4372. > View Incidents	N/A
RS000150	The 'Untitled - Paint' window was maximized.			/	Tested object: Sys["Process"]("mspaint")["Window"]("MSPaintApp", "Untitled - Paint", 1) > View Incidents	Passed
RS000151	The window was clicked with the left mouse button.			/	A click at point (28, 11) with no key pressed. Tested object: Sys["Process"]("mspaint")["Window"]("MSPaintApp", "Untitled - Paint", 1)["Window"]("UIRibbonCommandBarDock", "UIRibbonDockTop", 3) ["Window"]("UIRibbonCommandBar", "Ribbon", 1) ["Window"]("UIRibbonWorkPane", "Ribbon", 1) ["Window"]("NUIPane", "", 1) ["PropertyPage"]("Ribbon")["Button"](1) ["GridDropDownButton"]("Application menu") > View Incidents	Passed
RS000152	The window was clicked with the left mouse button.			/	A click at point (108, 22) with no key pressed. Tested object: Sys["Process"]("mspaint")["Window"]("Net UI Tool Window Layered", "", 1) ["Window"]("NetUIHWND", "", 1) ["Pane"](0) ["Client"](0) ["Grouping"](0) ["MenuItem"]("New") > View Incidents	Passed

4.3.3. Screenshot Capture

During the execution of the test, TestComplete will capture screenshots of the application being tested. These screenshots are uploaded to SpiraTest so that you have a complete record of the testing activities:

Overview ***** Attachments ***** Incidents

> [Add New](#) | [Add Existing](#) | [Remove](#) | [Refresh](#) | [Apply Filter](#) | [Clear Filter](#) | Include Source Code Documents

Displaying 1 - 15 out of 57 attachment(s).

✓	Document Name ▲▼	Type ▲▼	Size ▲▼	Edited By ▲▼	Edited On ▲▼	Author ▲▼	ID ▲▼
<input type="checkbox"/>	<input type="text" value=""/>	-- Any --	<input type="text" value=""/>	-- Any --	<input type="text" value=""/>	-- Any --	DC <input type="text" value=""/>
<input type="checkbox"/>	visImage58.png	Functional Specification	36 KB	System Administrator	22-Apr-2013	System Administrator	DC000106
<input type="checkbox"/>	visImage57.png	Functional Specification	37 KB	System Administrator	22-Apr-2013	System Administrator	DC000105
<input type="checkbox"/>	visImage56.png	Functional Specification	35 KB	System Administrator	22-Apr-2013	System Administrator	DC000104
<input type="checkbox"/>	visImage55.png	Functional Specification	43 KB	System Administrator	22-Apr-2013	System Administrator	DC000103
<input type="checkbox"/>	visImage54.png	Functional Specification	11 KB	System Administrator	22-Apr-2013	System Administrator	DC000102

Congratulations... You are now able to run TestComplete automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

5. Selenium

Selenium Remote Control (RC) is a test tool that allows you to write automated web application user interface tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser¹. Selenium RC comes in two parts.

- A server which can automatically launch and kill supported browsers, and acts as a HTTP proxy for web requests from those browsers.
- Client applications that send commands to the Selenium-RC server in a special language (called Selenese) that tell it what operations to perform on the launched web browser.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of Selenium-RC (hereafter just referred to as Selenium) on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated Selenium web tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 1.0 of Selenium-Remote Control.

5.1. Installing the Selenium Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [SeleniumAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate SeleniumX.dll for the version of Selenium that you are using.
 - ▷ If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "SeleniumX.dll" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Also copy the [ThoughtWorks.Selenium.Core.dll](#) from the zipfile into the "extensions" sub-folder.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.

¹ *Selenium RC Home Page*. OpenQA. 2010 <<http://selenium-rc.openqa.org>>

- Click the “Add” button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Selenium RC 1.0

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

Active

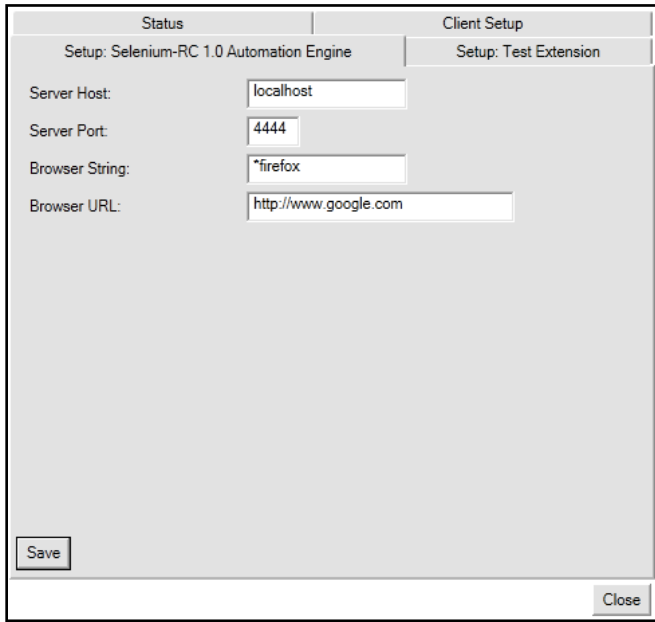
- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
 - **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
 - **Active:** If checked, the engine is active and able to be used for any project.
 - **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For Selenium this should be **SeleniumX** where 'X' is the version number of the DLL file that you are using.
- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Selenium listed as an available automation engine.

5.1.1. Advanced Settings

You can modify the Selenium configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

a) Selenium-RC 1.0

The Selenium 1.0 engine adds its own tab to this page which allows you to configure how Selenium operates:

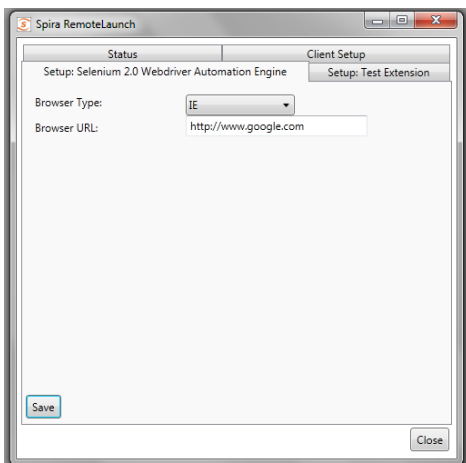


The following fields can be specified on this screen:

- **Server Host** – This should be the name / IP address of the Selenium server. Typically this will be localhost because RemoteLaunch is usually installed on the Selenium server itself.
- **Server Port** – This should be set to the custom port that the Selenium server uses as a proxy when intercepting requests to the browser. The default value is 4444.
- **Browser String** – This needs to be the name of the browser that the Selenium server will launch. Common values include:
 - ▷ ***firefox** – This will launch the Firefox web browser
 - ▷ ***iexplore** – This will launch the Microsoft Internet Explorer web browser
 - ▷ ***safari** – This will launch the Apple Safari web browser
- **Browser URL** – This needs to be the initial URL that you want the browser to open to

a) Selenium WebDriver 2.x

The Selenium 2.x engine adds its own tab to this page which allows you to configure how Selenium operates:



The following fields can be specified on this screen:

- **Browser Type** – This needs to be set to the type of browser that the Selenium webdriver will launch.
- **Browser URL** – This needs to be the initial URL that you want the browser to open to

5.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and either linking it to an existing Selenium test script file or entering a Selenium test script directly into SpiraTeam.

5.2.1. Attaching a Selenium Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

Automation

This section defines the automated test script associated with this test case:

Script Type: Attached Linked Repository

Automation Engine: Selenium WebDriver Version: 1.2 Document Type: Functional Specification

Filename: embedded_script.txt Document Folder: Test Scripts

Test Script: Edit Parameters

```
open|http://www.google.com/webhp
assert|title|Google
type|q|{query}
click|btn|G
wait|for|Page|to|Load||5000
is|Text|Present||{matchtext}
```

You need to enter the following fields:

- **Automation Engine** - Choose the Selenium Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Attached for this case
- **Filename** – Since the test script is going to be entered directly into SpiraTeam you can enter any name you like for the filename as long as it's logical and memorable.
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – This needs to contain the complete Selenium test script written in Selenium IDE *Selenese*. Selenium IDE test scripts consist of three parts:
 - ▷ The command
 - ▷ The target of the command
 - ▷ The data to be used
- You should enter the three components on each line separated by the Pipe (|) character. If you need to use a pipe character inside any of the components you can escape it with a backslash (\).

- ▷ An example command would be `type|q|hello`
- ▷ If the command doesn't need all three components, you can simply leave it out (for example `open||http://www.inflectra.com`)
- ▶ If you would like to have SpiraTeam pass any parameter values to this test script (this will be discussed in more detail later) you can specify them by using the syntax `${parameterName}`.
 - ▷ An example parameterized command would be `open||${url}`

A complete sample script is illustrated below:

```
open||http://www.google.com/webhp
assertTitle||Google
type|q|${query}
click|btnG
waitForPageToLoad||5000
isTextPresent||${matchtext}
```

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

5.2.2. Linking a Selenium Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the "Automation" tab:

The screenshot shows the 'Automation' configuration tab in SpiraTeam. At the top, there is a yellow informational box stating 'This section defines the automated test script associated with this test case.' Below this, the 'Script Type' dropdown is set to 'Linked'. The 'Automation Engine' is 'Selenium WebDriver', 'Version' is '1.2', 'Document Type' is 'Functional Specification', and 'Filename' is 'C:\MyTests\linked_script.htm'. There is also a 'Document Folder' field set to 'Test Scripts'. A 'Test Script' text area is present, and an 'Edit Parameters' button is located to its right.

You need to enter the following fields:

- ▶ **Automation Engine** - Choose the Selenium Automation Engine that you created in the previous section from the drop-down list.
- ▶ **Script Type** – This should be set to Linked for this case
- ▶ **Filename** – This needs to be the full path to the Selenium IDE test script file. To make this easier across different machines, you can use several constants for standard Windows locations:
 - ▷ [MyDocuments] – The user's "My Documents" folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [CommonDocuments] – The Public Document's folder.
 - ▷ [DesktopDirectory] – The user's Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it's the 64-bit directory.
 - ▷ [ProgramFilesX86] – Translated to the 32-bit Program Files directory.

- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used when you are using the linked test script option*

The linked test script needs to be an HTML document that contains a table with *three columns*. Each row corresponds to a single Selenium action. Each of the columns in the row corresponds to the three Selenium command components:

- ▷ The command
- ▷ The target of the command
- ▷ The data to be used

An example Selenium test script is illustrated below:

```

<html>
  <body>
    <table>
      <tr>
        <td>
          open
        </td>
        <td>
          http://www.google.com/webhp
        </td>
      </tr>
      <tr>
        <td>
          assertTitle</td>
        <td>
          &nbsp;</td>
        <td>
          Google</td>
      </tr>
      <tr>
        <td>
          type</td>
        <td>
          q</td>
        <td>
          ${query}</td>
      </tr>
      <tr>
        <td>
          click</td>
        <td>
          btnG</td>
        <td>
          &nbsp;</td>
      </tr>
      <tr>
        <td>
          waitForPageToLoad</td>
        <td>
          &nbsp;</td>
        <td>

```

```

5000</td>
</tr>
<tr>
<td>
isTextPresent</td>
<td>
&nbsp;</td>
<td>
${matchtext}</td>
</tr>
</table>
</body>
</html>

```

When opened in an HTML editing tool it looks like:

open		http://www.google.com/webhp
assertTitle		Google
type	q	\${query}
click	btnG	
waitForPageToLoad		5000
isTextPresent		\${matchtext}

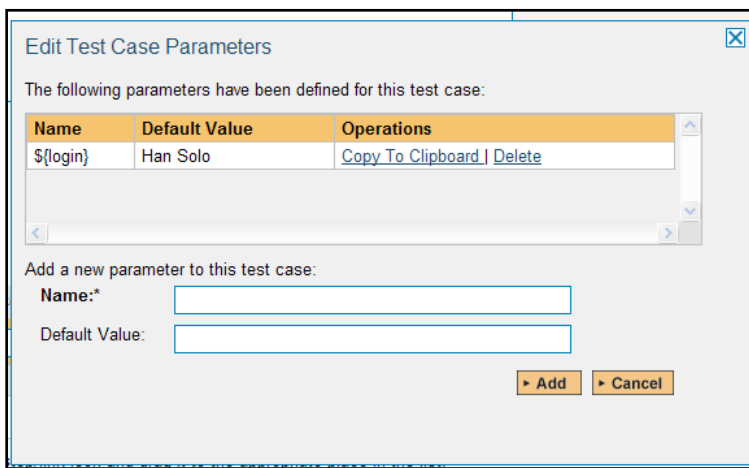
- If you would like to have SpiraTeam pass any parameter values to this test script (this will be discussed in more detail later) you can specify them by using the syntax `${parameterName}`.
 - ▷ An example parameterized command is displayed in the third and sixth rows of the table above (`${query}` and `${matchtext}`).

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

5.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your Selenium automated test script. This is very useful if you want to have a data-driven Selenium test script that be executed multiple times with different parameter values.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter `login` needs to match the name of the input parameter defined within the Selenium script.

5.3. Executing the Selenium Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

5.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
IC 7.0 Testing (1)	In Progress	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	Completed	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmarteScript Testing (1)	Completed	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	Completed	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	Completed	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	Completed	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Selenium automated test cases and click on its hyperlink to display the test set details page:

Test Set: Selenium Testing [TX:000013]

Name*: Selenium Testing

Description: -- Font -- -- Size -- **B** **I** **U**

Owner: -- None -- **Creator*:** Fred Bloggs

Release: --- None --- **Type*:** Automated

Automation Host: InflectraSvr03 **Created On:** 10/28/2010 4:10:34 PM

Status*: Completed **Last Executed:** -

Planned Date: 10/31/2010 09:32:00 PM **Last Updated:** 11/3/2010 4:26:59 PM

Test Cases * Test Runs * Comments Custom Props Attachments History *

> **Add Tests** **Remove Tests** **Refresh** **Edit Parameters** **Execute Tests** Est. Duration: 0.0h / Actual Duration: 0.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Duration	Act. Duration	Last Executed	Execution Status	Test Case #	Edit
<input type="checkbox"/>	Google Test (Linked)				0.0h	31-Oct-2010	Passed	TC000023	
<input type="checkbox"/>	Google Test (Attached)				0.0h	31-Oct-2010	Passed	TC000022	
<input type="checkbox"/>	Google Test (Attached)				0.0h	31-Oct-2010	Passed	TC000022	

Show 15 rows per page Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameters

The following parameter values have been specified at the test set level: [+ Add Parameter Value](#)

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	
\$(login)	librarian	-	
\$(password)	librarian	-	

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	C-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login: admin

operatingSystem: _____

password: admin

url: _____

Save Cancel

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview Test Sets *

Properties + Dates and Times +

Detailed Information +

Test Configuration Entries

Populate Remove Refresh

Displaying 10 out of 10 entries in this test configuration set.

\$(login)	\$(browserName)	ID
admin	Internet Explorer	22
admin	Mozilla / Firefox	23
admin	Safari	25
borrower	Internet Explorer	26
borrower	Mozilla / Firefox	27
borrower	Opera	28

5.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their

list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the Selenium test:

- **Passed** – The Selenium automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The Selenium automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The Selenium automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser windows launch as the Selenium server executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by Selenium, you will see the following information:

The screenshot displays a 'Test Run' summary for 'Google Test (Attached) [TR:000108]'. It includes fields for Release # (None), Tester Name (System Administrator), Test Set (Selenium Testing), Test Case # (TC000022), and Automation Host (InflectraSvr03). Execution details show an estimated duration of 0 hours and 0 minutes, an actual duration of 0 hours and 0 minutes, an execution date of 10/31/2010 9:32:35 PM, and an execution status of 'Passed'. The test run type is 'Automated'. Below these fields are tabs for 'Test Run Steps', 'Automation *', 'Custom Properties', and 'Attachments'. The 'Automation *' tab is active, showing the runner name 'Selenium-RC 1.0 Auto', an assert count of 0, and a message: 'Tests completed with 6 successful commands and 0 failures.' The test name is 'Google Test'. A 'Details' section shows a list of Selenium commands and their results: 'open (. http://www.google.com/webhp) - OK', 'assertTitle (. Google) - OK', 'type (q, Philomene Long) - OK', 'click (btnG,) - OK', 'waitForPageToLoad (. 5000) - OK', and 'isTextPresent (. Philomene Long) - OK.true'.

This screen indicates the status of the test run that was reported back from Selenium together with any messages or other information. The execution status will be set to PASSED if all the Selenium commands report back OK and all the tests passed. If any of the commands failed or the tests don't pass, the overall execution status will be listed as FAILED.

The Message field will contain a summary of the number of commands executed and the number of failed commands, with the large details box containing the full command execution log as reported back from Selenium:

```
open (, http://www.google.com/webhp) - OK
assertTitle (, Google) - OK
type (q, Philomene Long) - OK
click (btnG, ) - OK
waitForPageToLoad (, 5000) - OK
isTextPresent (, Philomene Long) - OK,true
```

Congratulations... You are now able to run Selenium automated web tests and have the results be recorded within SpiraTest / SpiraTeam.

6. Squish

Froglogic® Squish® (hereafter Squish) is a functional test automation system that lets you record application operations and generate test automation scripts in a variety of different scripting languages (JavaScript, Tcl, Python) that can be used to playback the test script against the test application.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of Squish on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated Squish tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 4.0 of Squish running on a Windows® platform.

6.1. Installing the Squish Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [SquishAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate SquishX.dll for the version of Squish that you are using.
 - ▷ If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "SquishX.dll" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Squish 4.0
[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

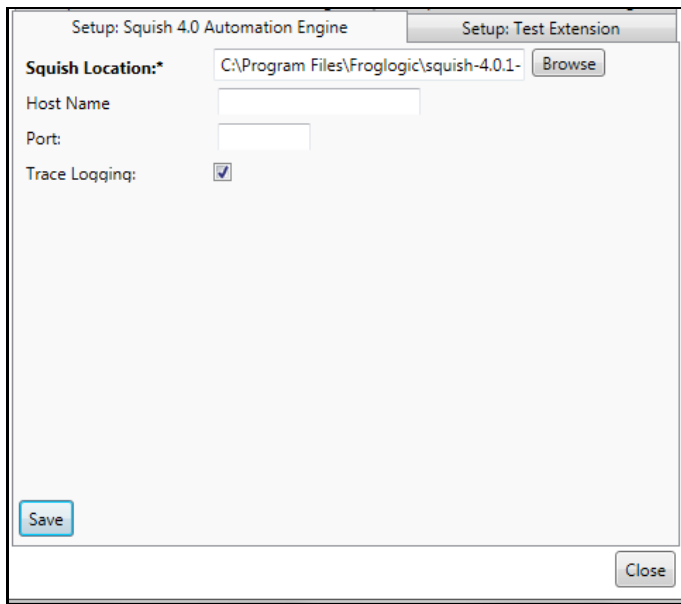
Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.

- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For Squish this should be **SquishX** where 'X' is the version number of the DLL file that you are using.
- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Squish listed as an available automation engine.

6.1.1. Squish RemoteLaunch Settings

You will need to modify the Squish configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page. The Squish engine adds its own tab to this page which allows you to configure how Squish operates:



The following fields can be specified on this screen:

- **Squish Location** – This should be folder containing the “SquishRunner” executable that will be used to actually run the automated tests.
- **Server Host** – This field can be set to the name of a remote Squish server if you did not install RemoteLaunch on the machine running the Squish server (optional).
- **Server Port** – This field can be set to the port being used by a remote Squish server if you did not install RemoteLaunch on the machine running the Squish server (optional).
- **Trace Logging** – This checkbox can be selected if you need to provide debugging information to Inflectra support personnel. Normally this should remain unchecked

Note: In most cases, the second and third fields can be left empty.

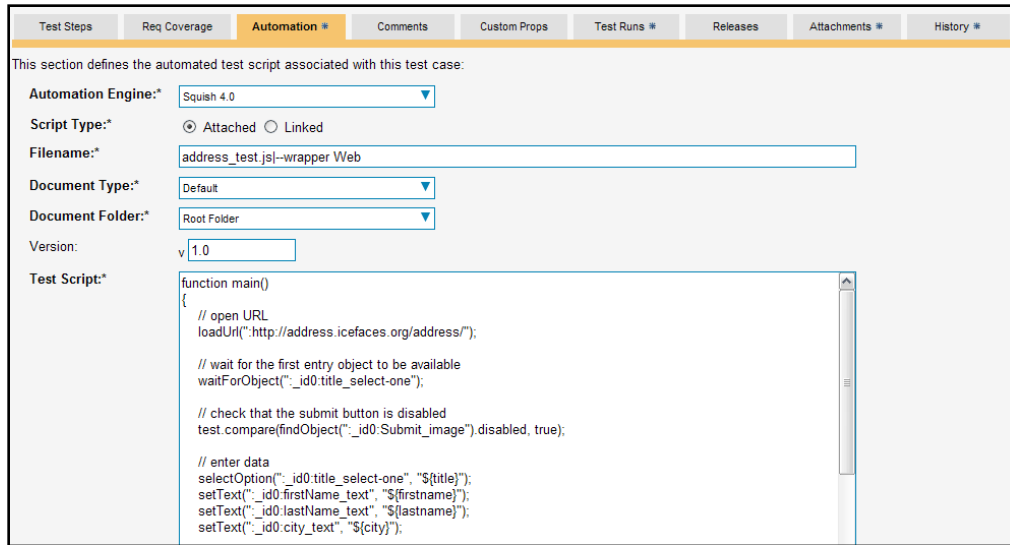
6.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and either linking it to an existing Squish test suite, test case or entering a Squish test script directly into SpiraTeam.

*Note: that the Squish engine only supports passing parameters to an **attached test script** and not to a **linked test script**.*

6.2.1. Attaching a Squish Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:



The screenshot shows the 'Automation' tab in SpiraTeam. The interface includes a navigation bar with tabs: Test Steps, Req Coverage, Automation (selected), Comments, Custom Props, Test Runs, Releases, Attachments, and History. Below the navigation bar, a message states: 'This section defines the automated test script associated with this test case:'. The configuration fields are as follows:

- Automation Engine:** Squish 4.0 (dropdown)
- Script Type:** Attached (radio selected), Linked (radio)
- Filename:** address_test.js|--wrapper Web (text input)
- Document Type:** Default (dropdown)
- Document Folder:** Root Folder (dropdown)
- Version:** 1.0 (text input)
- Test Script:** A code editor containing the following JavaScript code:

```
function main()
{
  // open URL
  loadUrl("http://address.icefaces.org/address/");

  // wait for the first entry object to be available
  waitForObject("_id0:title_select-one");

  // check that the submit button is disabled
  test.compare(findObject("_id0:Submit_image").disabled, true);

  // enter data
  selectOption("_id0:title_select-one", "${title}");
  setText("_id0:firstName_text", "${firstname}");
  setText("_id0:lastName_text", "${lastname}");
  setText("_id0:city_text", "${city}");
}
```

You need to enter the following fields:

- **Automation Engine** - Choose the Squish Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Attached for this case
- **Filename** – Since the test script is going to be entered directly into SpiraTeam you can enter any filename you like as long as the file extension matches the scripting language that you’re using. After that you need to add any command-line parameters after the filename, separated by a pipe (|) symbol.
 - ▷ For example, to launch a web test using Javascript, you’d use:
address_test.js|--wrapper Web
 - ▷ For example, to launch an application test using Python, you’d use:
address_test.py|--aut <application>
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – This needs to contain the complete Squish test script. Squish test scripts can be written in JavaScript, Python or TCL.
 - ▷ If you would like to have SpiraTeam pass any parameter values to this test script (this will be discussed in more detail later) you can specify them by using the syntax `${parameterName}`.

A complete sample script (illustrating the use of parameters) is illustrated below:

```
function main()
{
    // open URL
    loadUrl("http://address.icefaces.org/address/");

    // wait for the first entry object to be available
    waitForObject(":_id0:title_select-one");

    // check that the submit button is disabled
    test.compare(findObject(":_id0:Submit_image").disabled, true);

    // enter data
    selectOption(":_id0:title_select-one", "${title}");
    setText(":_id0:firstName_text", "${firstname}");
    setText(":_id0:lastName_text", "${lastname}");
    setText(":_id0:city_text", "${city}");

    // check that after entering city, the state is automatically chosen correctly
    var state = "${state}";
    setFocus(":_id0:state_text");
    if (!test.verify(waitFor("findObject(':_id0:state_text').value == state", 10000)))
    {
        clickButton(":_id0:Reset_image");
        continue;
    }

    // input ZIP
    selectOption(":_id0:zipSelect_select-one", "${zip}");

    // check that submit button is enabled now
    setFocus(":_id0:lastName_text");
    if (!test.verify(waitFor("findObject(':_id0:Submit_image').disabled == false", 10000)))
    {
        clickButton(":_id0:Reset_image");
    }

    // submit
    clickButton(":_id0:Submit_image");

    // wait for results page
    waitForContextExists(":response.iframe");
    waitForObject(":_idl_id3_SPAN");

    // verify that data is stored and displayed correctly
    test.compare(findObject(":_idl_id3_SPAN").innerText, firstName);
    test.compare(findObject(":_idl_id6_SPAN").innerText, state);

    // close browser
    closeWindow(":[Window]");
}
```

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

6.2.2. Linking a Squish Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

You need to enter the following fields:

- **Automation Engine** - Choose the Squish Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked for this case
- **Filename** – This needs to be the full path to the Squish test case or test suite folder.
 - ▷ If specifying a test case folder, you need to also provide the configuration command-line parameters after the filename, separated by a pipe (|) symbol. These are not needed if executing a test suite, since they are contained in the suite.conf file instead.
 - For example, to launch a web **test case** you'd use:
`[ProgramFiles]\Froglogic\squish-4.0.1-web-win32\examples\web\suite_examples\tst_icefaces_addressbook_datadriven|--wrapper Web`
 - For example, to launch a web **test suite** you'd simply use:
`[ProgramFiles]\Froglogic\squish-4.0.1-web-win32\examples\web\suite_examples`
 - For example, to launch a web **test case within a test suite** you'd use the path of the test suite, followed by the pipe (|) symbol, followed by the test case name:
`[ProgramFiles]\Froglogic\squish-4.0.1-web-win32\examples\web\suite_examples|tst_icefaces`
 - ▷ To make this easier across different machines, you can use several constants for standard Windows locations:
 - [MyDocuments] – The user's "My Documents" folder. The user indicated is the user that ran RemoteLaunch.
 - [CommonDocuments] – The Public Document's folder.
 - [DesktopDirectory] – The user's Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it's the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.

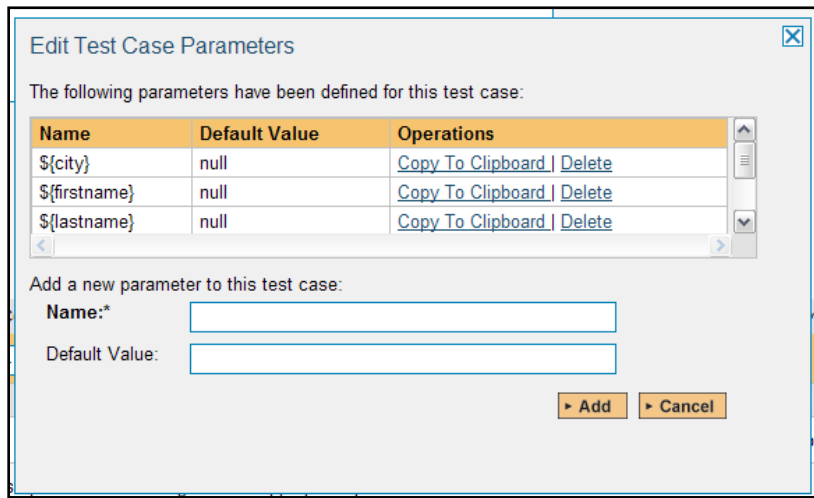
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used when you are using the linked test script option*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

6.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your *attached* (not linked) Squish automated test script. This is very useful if you want to have a data-driven Squish test script that be executed multiple times with different parameter values.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter `${city}` needs to match the name of the parameter defined within the attached Squish script.

6.3. Executing the Squish Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

6.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
TC 7.0 Testing (1)	In Progress	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	Completed	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmartScript Testing (1)	Completed	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	Completed	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	Completed	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	Completed	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Squish automated test cases and click on its hyperlink to display the test set details page:

Test Set: Squish Testing [TX:000014]

Name*: Squish Testing

Description:

Owner: -- None -- Creator*: System Administrator

Release: --- None --- Type*: Automated

Automation Host: TestHost (VM) Created On: 11/2/2010 12:36:27 PM

Status*: Completed Last Executed: -

Planned Date: 11/2/2010 05:02:00 PM Last Updated: 11/3/2010 4:26:59 PM

Test Cases * | Test Runs * | Comments | Custom Props | Attachments | History *

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Duration: 0.0h / Actual Duration: 0.0h

Test Case Name	Owner	Priority	Est. Duration	Act. Duration	Last Executed	Execution Status	Test Case #	Edit
Address Book (Attached)				0.0h	2-Nov-2010	Failed	TC000026	Edit
Address Book (Test Case)				0.0h	2-Nov-2010	Failed	TC000025	Edit
Address Book (Test Suite)				0.0h	2-Nov-2010	Failed	TC000024	Edit

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.

- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	Edit Delete
\$(login)	librarian	-	Edit Delete
\$(password)	librarian	-	Edit Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login: admin

operatingSystem: _____

password: admin

url: _____

[Save](#) [Cancel](#)

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

The screenshot shows a web interface for a Test Configuration Set [TG-4]. It has tabs for 'Overview' and 'Test Sets *'. Below the tabs are sections for 'Properties', 'Dates and Times', and 'Detailed Information'. Under 'Test Configuration Entries', there are buttons for '+ Populate', 'x Remove', and 'Refresh'. A message states 'Displaying 10 out of 10 entries in this test configuration set.' Below this is a table with columns for selection, login name, browser name, and ID.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

6.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the Squish test:

- **Passed** – The Squish automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The Squish automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The Squish automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see application or browser windows launch as the Squish server executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by Squish, you will see the following information:

Test Run: Address Book (Test Suite) [TR:000146]

Release #: --- None --- Est. Duration: 0 hours 0 minutes
 Tester Name: Remote Launch Actual Duration: 0 hours 1 minutes
 Test Set: Squish Testing Execution Date: 11/2/2010 4:51:13 PM
 Test Case #: TC000024 Execution Status: **Passed**
 Automation Host: TestHost (VM) Test Run Type: Automated

Automation * Custom Properties Attachments

Runner Name: Squish 4.0 Automatio Assert Count: 0
 Message: 10 tests completed with 0 errors, 0 fatals, 0 fails, 10 passes, and 0 warnings Test Name: [ProgramFiles]\Froglogic\squish-4.0.1-web-win32\examples\web\suite_examples

Details:

```

2010-11-02T16:50:01-04:00 START_TEST suite_examples
2010-11-02T16:50:01-04:00 START_TEST tst_icefaces_addressbook_datadriven
2010-11-02T16:50:05-04:00 PASS Comparison 'true' and 'true' are equal
2010-11-02T16:50:25-04:00 PASS Verified True expression
2010-11-02T16:50:30-04:00 PASS Verified True expression
2010-11-02T16:50:32-04:00 PASS Comparison 'Reginald' and 'Reginald' are equal
2010-11-02T16:50:33-04:00 PASS Comparison 'CA' and 'CA' are equal
2010-11-02T16:50:40-04:00 PASS Comparison 'true' and 'true' are equal
2010-11-02T16:50:59-04:00 PASS Verified True expression
2010-11-02T16:51:09-04:00 PASS Verified True expression
2010-11-02T16:51:12-04:00 PASS Comparison 'Tanja' and 'Tanja' are equal
2010-11-02T16:51:12-04:00 PASS Comparison 'NY' and 'NY' are equal
  
```

This screen indicates the status of the test run that was reported back from Squish together with any messages or other information. The execution status will be set according to the worst-case assessment reported back from Squish. The various Squish statuses are mapped to their nearest equivalent SpiraTeam statuses as illustrated below:

Squish Status	SpiraTeam Status
PASS	Passed
FAIL	Failed
WARNING	Caution
FATAL	Blocked
ERROR	Failed

In addition, the Message field will contain a summary of the number of tests completed and the number of tests that reported an error, fatal, fail, pass or warning status.

```

2010-11-02T16:50:01-04:00 START_TEST suite_examples
2010-11-02T16:50:01-04:00 START_TEST tst_icefaces_addressbook_datadriven
2010-11-02T16:50:05-04:00 PASS Comparison 'true' and 'true' are equal
2010-11-02T16:50:25-04:00 PASS Verified True expression
2010-11-02T16:50:30-04:00 PASS Verified True expression
2010-11-02T16:50:32-04:00 PASS Comparison 'Reginald' and 'Reginald' are equal
2010-11-02T16:50:33-04:00 PASS Comparison 'CA' and 'CA' are equal
2010-11-02T16:50:40-04:00 PASS Comparison 'true' and 'true' are equal
2010-11-02T16:50:59-04:00 PASS Verified True expression
2010-11-02T16:51:09-04:00 PASS Verified True expression
2010-11-02T16:51:12-04:00 PASS Comparison 'Tanja' and 'Tanja' are equal
2010-11-02T16:51:12-04:00 PASS Comparison 'NY' and 'NY' are equal
  
```

Congratulations... You are now able to run Squish automated web tests and have the results be recorded within SpiraTest / SpiraTeam.

7. Command-Line

In addition to the various pre-built plug-ins for different test automation engines, there is a generic command-line engine available that lets RemoteLaunch execute an arbitrary command-line program, capture the console output and send the output back to SpiraTeam as the test results. This is useful when you want to be able to use SpiraTeam to manage the scheduling and execution of automated testing using an in-house tool or a third-party tool that Inflectra has not yet built a plug-in for.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of a command-line application on different computers and have the “testing” results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam’s test management capabilities to include automation

Note: This integration requires at least version 4.0 of SpiraTest/Team and RemoteLaunch.

7.1. Installing the Command-Line Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [CommandLineAutomationEngine.zip](#) file from the Inflectra website and locate the CommandLine.dll
- Copy the file “CommandLine.dll” into the “extensions” sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the “Test Automation” link under **Integration**.
- Click the “Add” button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Command-Line

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

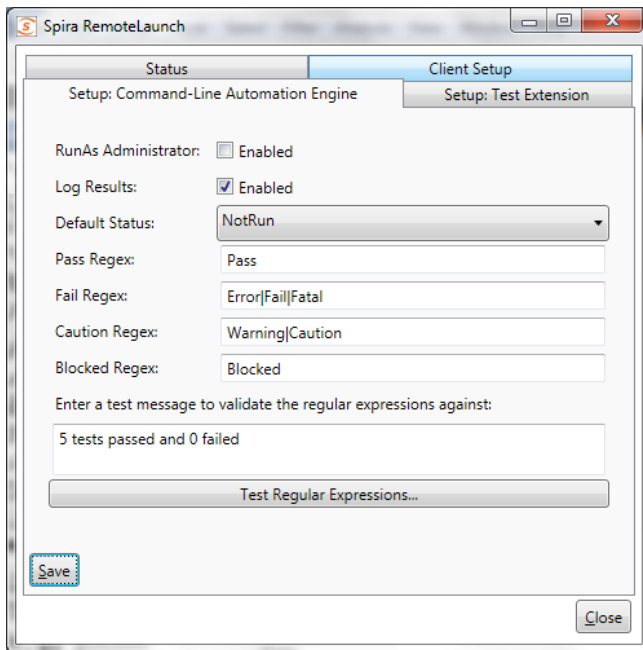
Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For Command-Line this should be simply “**CommandLine**”.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Command-Line listed as an available automation engine.

7.1.1. Command-Line RemoteLaunch Settings

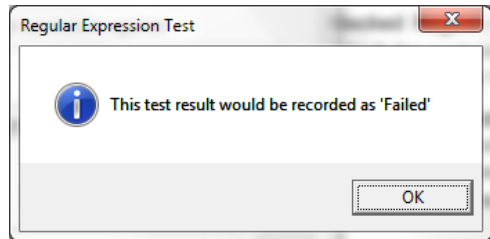
You may need to modify the Command-Line configuration for some of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page. The Command-Line engine adds its own tab to this page which allows you to configure how the Command-Line engine operates:



The following fields can be specified on this screen:

- **RunAs Administrator** – This normally should not be checked. However if your automation tool requires Windows UAC elevation to operate, you will need to select this option. We recommend initially trying your tool with the value unchecked. Then, if you get an error message “requires elevation” in the test results you will need to select the option.
- **Log Results** – Normally the command-line engine will capture the output results from the command-line and send the results back to SpiraTeam as the test result. When you are executing a tool that directly integrates with SpiraTeam (e.g. a NUnit test suite that is already integrated with SpiraTeam) you don’t want two different results to be sent back. In such a scenario, deselecting this option will prevent the command-line engine from sending back its own test result.
- **Default Status** – This specifies the execution status that will be returned to SpiraTeam in the event that none of the regular expressions (Regex) specified match the results returned from the test application. By default, the system will return “Passed” if none of the other regular expressions match correctly.
- **Pass Regex** – This is the regular expression that is used to match a passed test result. By default the system will search for the phrase “Pass” in the test output and return a Passed status if the match is successful.
- **Fail Regex** – This is the regular expression that is used to match a failed test result. By default the system will search for the phrases “Fail”, “Error” and “Fatal” in the test output and return a Fail status if any of the matches are successful.

- **Caution Regex** – This is the regular expression that is used to match a caution test result. By default the system will search for the phrases “Warning” and “Caution” in the test output and return a Caution status if any of the matches are successful.
- **Blocked Regex** – This is the regular expression that is used to match a blocked test result. By default the system will search for the phrase “Blocked” in the test output and return a Blocked status if the match is successful.
- **Test Regular Expressions** – This text box lets you enter in some sample text and see how the Command-Line extension would interpret it. Once you have entered in the text, click “Test Regular Expression...” and the system will display a popup message box letting you know what the outcome of such a test would be interpreted as:



7.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and either linking it to an existing test script file or entering a test script directly into SpiraTeam.

7.2.1. Attaching a Command-Line Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

You need to enter the following fields:

- **Automation Engine** - Choose the Command-Line Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Attached for this case
- **Filename** – This needs to consist of the following sections separated by a pipe (|) character:
 - ▷ The full path to the command-line tool. To make this easier across different machines, you can use several constants for standard Windows locations:
 - [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.

- [CommonDocuments] – The Public Document's folder.
 - [DesktopDirectory] – The user's Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it's the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- ▷ Any arguments for the command-line tool, with the filename specified as `${filename}`. This special token will be replaced by the actual filename of the test script when RemoteLaunch downloads it from SpiraTeam. In addition, you can use the following additional tokens for some of the special SpiraTeam ID values:
- [TestCaselId] – the ID of the test case
 - [TestSetId] – the ID of the test set
 - [ReleaseId] – the ID of the release (if specified)
 - [ProjectId] – the ID of the project
- ▷ An example filename would be:
`C:\Temp\TestApp.exe|-arg1 -arg2 "-arg3=${filename}"|`
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
 - **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
 - **Version** – The version of the test script (1.0 is used if no value specified)
 - **Test Script** – This needs to contain the complete test script in whatever language and syntax is being expected by the command-line application
 - If you would like to have SpiraTeam pass any parameter values to this test script (this will be discussed in more detail later) you can specify them by using the syntax `${parameterName}` inside the test script.

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

7.2.2. Linking a Command-Line Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the "Automation" tab:

You need to enter the following fields:

- **Automation Engine** - Choose the Command-Line Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked for this case
- **Filename** – This needs to consist of the following sections separated by a pipe (|) character:
 - ▷ The full path to the command-line tool. To make this easier across different machines, you can use several constants for standard Windows locations:
 - [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - [CommonDocuments] – The Public Document’s folder.
 - [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
 - ▷ Any arguments for the command-line tool, including the filepath of the test script file that the command-line tool will be executing. In addition, you can use the following additional tokens for some of the special SpiraTeam ID values:
 - [TestCaseId] – the ID of the test case
 - [TestSetId] – the ID of the test set
 - [ReleaseId] – the ID of the release (if specified)
 - [ProjectId] – the ID of the project
 - ▷ The mask for converting any parameter values from SpiraTeam into valid command line arguments. If parameters are not accepted by the command-line tool, you can leave this section out.
 - The mask can include any symbols together with “name” to refer to the parameter name and “value” to refer to the parameter value.
 - Example 1: If you want parameters to provided in the form:
 -param1=value1 -param2=value2
 you would use the following mask:
 -name=value
 - Example 2: If you want parameters to provided in the form:
 /param1:value1 /param2:value2
 you would use the following mask:
 /name:value
 - ▷ An example filename would be:
 C:\Temp\TestApp.exe|-arg1 -arg2|-name=value
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)

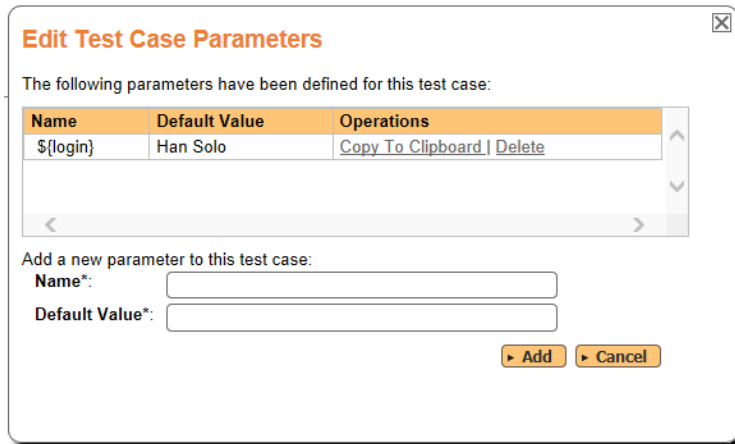
- **Test Script** – *This is not used when you are using the linked test script option*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

7.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your command-line automated testing tool. This is very useful if you want to have a data-driven test script that be executed multiple times with different parameter values.

To setup the automated test case for parameters, click on the “Edit Parameters” hyperlink above the “Test Script” box:



The name of the parameter `${login}` needs to match the name of a parameter accepted by the command-line tool.

7.3. Executing the Command-Line Test Sets from SpiraTeam

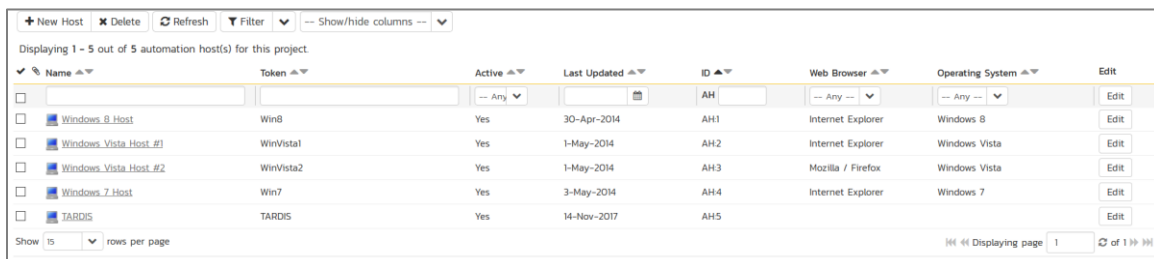
There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

7.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:



Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field

must be set to the same token that is specified in the RemoteLaunch application on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
TC 7.0 Testing (1)	In Progress	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	Completed	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmartScript Testing (1)	Completed	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	Completed	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	Completed	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	Completed	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Command-Line automated test cases and click on its hyperlink to display the test set details page:

Test Set: Command Line Testing [TX:000028]

Name: Command Line Testing

Overview | Test Runs | Attachments | Incidents | History #

Details

Owner: -- None -- Creator*: System Administrator
 Release: -- None -- Type: Manual
 Automation Host: -- None -- Creation Date: 4/23/2013 10:29:08 PM
 Status*: Not Started Last Executed: -
 Planned Date: -- One Time -- Last Updated: 4/23/2013 10:30:55 PM

Notes: [Rich Text Editor]

Operating System: --- Please Select ---

Description

Comments

Test Cases

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Dur.: 0.00 / Act. Dur.: 0.00

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
Test App (Linked)						Not Run	TC000064	Edit

Show 15 rows per page

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameters

The following parameter values have been specified at the test set level: [+ Add Parameter Value](#)

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	Edit Delete
\$(login)	librarian	-	Edit Delete
\$(password)	librarian	-	Edit Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Cases

[+ Add](#) [Remove](#) [Refresh](#) [Edit Parameters](#) -- Show/hide columns -- [Execute Tests](#) Est. Dur. 0.7h / Act. Dur. 3.0h

<input type="checkbox"/> Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
<input type="checkbox"/> Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
<input type="checkbox"/> Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
<input type="checkbox"/> Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
<input type="checkbox"/> Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
<input type="checkbox"/> Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
<input type="checkbox"/> Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
<input type="checkbox"/> Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

Show 15 rows per page Displaying page 1 of 1

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login: admin _____

operatingSystem: _____

password: admin _____

url: _____

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

The screenshot shows the SpiraTeam interface for a Test Configuration Set [TG:4]. The interface includes tabs for Overview and Test Sets. Below the tabs are sections for Properties, Dates and Times, Detailed Information, and Test Configuration Entries. The Test Configuration Entries section contains a table with 10 entries, each representing a unique combination of login and browser.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

7.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the command-line test:

- **Passed** – The automated test ran successfully and the results output to the console did not include any of the phrases – FAIL, ERROR, FATAL, WARNING, CAUTION
- **Failed** – The automated test ran successfully, but one of the phrases – FAIL, ERROR, FATAL – was included in the console output
- **Caution** – The automated test ran successfully, but one of the phrases – WARNING, CAUTION – was included in the console output
- **Blocked** – The automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see application windows launch as the command-line tool server executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by the command-line tool, you will see the following information:

Test Run: Test App (Linked) [TR:000159]

Release #:
Est. Duration: hours minutes
Tester Name:
Actual Duration: hours minutes
Test Set: [Command Line Testing](#)
Execution Date: 11/3/2010 4:51:00 PM
Test Case #: [TC000031](#)
Execution Status: Failed
Automation Host: [InflectraSvr03](#)
Test Run Type: Automated

Test Run Steps | Automation * | Custom Properties | Attachments

Runner Name: Command-Line Automat **Assert Count:** 1
Message: ment = '-arg2' argument = '-result=fail' *** Finishing TestApp *** **Test Name:** Unknown?

Details:

```

*** Starting TestApp ***
argument = -arg1
argument = -arg2
argument = '-result=fail'
*** Finishing TestApp ***

```

This screen indicates the status of the test run that was reported back from command-line tool together with any messages or other information. The execution status will be set according to the rules described above, the Message field will contain the first line of console output and the large details box will contain the full console output from the command-line tool.

Congratulations... You are now able to run a custom command-line run tests and have the results be recorded within SpiraTest / SpiraTeam.

8. LoadRunner

HP® LoadRunner is a load testing system that lets you record application performance by a number of 'virtual users'.

This section covers installing and using the Engine to report back statistics of run scenarios.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 11 of LoadRunner. The extension has not been tested on previous versions of LoadRunner due to lack of availability of previous released versions.

8.1. Installing the LoadRunner11 Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [LoadRunner11Engine.zip](#) file from the Inflectra website.
- Copy the files in the ZIP file into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Load Runner 11

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in bold:

Name*:

Description:

Token*:

Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users, and will be displayed in the dropdown when the user selects the Tester.
 - **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
 - **Active:** If checked, the engine is active and able to be used for any project.
 - **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For LoadRunner, it needs to be "**LoadRunner11**".
- Once you have finished, click the "Insert & Close" button and you will be taken back to the Test Automation list page, with LoadRunner listed as an available automation engine.

8.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to a Load Runner Scenario.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

The screenshot shows the 'Automation' tab in SpiraTeam. The form is titled 'This section defines the automated test script associated with this test case:'. It contains the following fields:

- Automation Engine:***: A dropdown menu with 'Load Runner 11' selected.
- Script Type:***: Radio buttons for 'Attached' and 'Linked', with 'Linked' selected.
- Filename:***: A text input field containing 'C:\Program Files\HP\LoadRunner\tutorial\demo_scenario.lrs'.
- Document Type:***: A dropdown menu with 'Functional Specification' selected.
- Document Folder:***: A dropdown menu with 'Root Folder' selected.
- Version:**: A text input field with 'v 1.0'.
- Test Script:***: An empty text input field.

You need to enter the following fields:

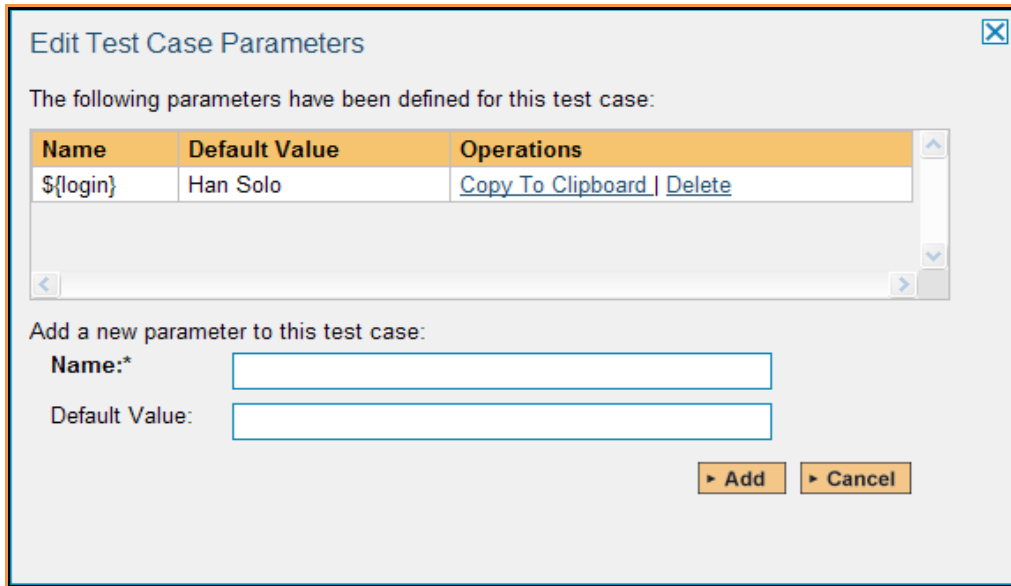
- **Automation Engine** - Choose the LoadRunner Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – For LoadRunner, all scenarios must be stored on the local testing machine so 'Linked' must be selected. If you select 'Attached', when the scenario is attempted to be executed it will be marked as blocked and skipped.
- **Filename** – This needs to be the full path to the LoadRunner Scenario (*.lrs) file. Certain tokens are allowed to be able to specify common locations across different operating systems. Note that the tokens are case-sensitive, and there are no spaces in them. A list of tokens are:
 - [MyDocuments] – The user's "My Documents" folder. The user indicated is the user that ran RemoteLaunch.
 - [CommonDocuments] – The Public Document's folder.
 - [DesktopDirectory] – The user's Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it's the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated scenario will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated scenario will be stored in.
- **Version** – The version of the scenario (1.0 is used if no value specified)
- **Test Script** – *Not used.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

8.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your LoadRunner scenario. This is very useful if you have a data-driven LoadRunner scenario that has custom parameters used that you would like to change based on the test.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter \${login} needs to match the name of the custom parameter defined in the LoadRunner scenario. Invalid parameters will be silently ignored by the LoadRunner engine. Parameters must have a unique name.

8.3. Executing the LoadRunner Scenario from SpiraTeam

There are two ways to execute a scenario in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

8.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Displaying 1 - 5 out of 5 automation host(s) for this project.

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Show 15 rows per page

Displaying page 1 of 1

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified as the Host name in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

✓	Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
<input type="checkbox"/>	<input type="text"/>	-- Any --	<input type="text"/>	<input type="text"/>	-- Any --	-- Any --	-- Any --	TX	<input type="button" value="Filter"/>
<input type="checkbox"/>	TC 7.0 Testing (1)	 	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	<input type="button" value="Edit"/>
<input type="checkbox"/>	QTP Testing (1)	 	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	<input type="button" value="Edit"/>
<input type="checkbox"/>	SmarteScript Testing (1)	 	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	<input type="button" value="Edit"/>
<input type="checkbox"/>	Selenium Testing (3)	 	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	<input type="button" value="Edit"/>
<input type="checkbox"/>	Squish Testing (3)	 	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	<input type="button" value="Edit"/>
<input type="checkbox"/>	Command Line Testing (1)	 	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	<input type="button" value="Edit"/>

Show 15 rows per page Displaying page 1 of 1

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the LoadRunner test cases and click on its hyperlink to display the test set details page:

Test Set: QTP Testing [TX:000011]

Name*:

Description:

Owner: Creator*:

Release: Type*:

Automation Host: Created On: 10/21/2010 3:06:20 PM

Status*: Last Executed: -

Planned Date: Last Updated: 11/3/2010 4:26:59 PM

Est. Duration: 0.0h / Actual Duration: 0.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Duration	Act. Duration	Last Executed	Execution Status	Test Case #	Edit
<input type="checkbox"/>	Flight Test 1				0.0h	22-Oct-2010	Passed	TC000020	<input type="button" value="Edit"/>

Show 15 rows per page Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	<input type="checkbox"/> Edit <input type="checkbox"/> Delete
\$(login)	librarian	-	<input type="checkbox"/> Edit <input type="checkbox"/> Delete
\$(password)	librarian	-	<input type="checkbox"/> Edit <input type="checkbox"/> Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login: admin _____

operatingSystem: _____

password: admin _____

url: _____

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview Test Sets *

Properties + Dates and Times +

Detailed Information +

Test Configuration Entries

+ Populate ✕ Remove ↻ Refresh

📌 Displaying 10 out of 10 entries in this test configuration set.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

8.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be executed. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the LoadRunner execution:

- **Passed** – The Scenario ran and reported no error messages. Warning, Debug, and Informational messages may have been logged, however.
- **Failed** – The Scenario ran and at least one error message was reported.
- **Blocked** – There was an error with the Test Set or LoadRunner application.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser or application windows launch as LoadRunner runs the scenario and connects VUsers to their tasks.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by LoadRunner, you will see the following information:

Runner Name:	LoadRunner 11 Automa	Assert Count:	8
Message:	Execution Statistics: Duration- 00:04:03; Failed- 20; Passed- 0; Hits Per Minute- 0;	Test Name:	demo_scenario.lrs
Details:	<pre> Scenario: demo_scenario.lrs ----- Result Directory: c:\program files\mercury interactive\mercury loadrunner\results\tutorial_demo_res Duration- 00:04:03 Failed- 20 Passed- 0 Hits Per Minute- 0 Execution Messages: : C:\Program Files\HP\LoadRunner\tutorial\demo_scenario.lrs : c:\program files\mercury interactive\mercury loadrunner\results\tutorial_demo_res\tutorial_demo_res.lrr Script Messages: 3/16/2011 10:42:24 AM [ERROR] VUser demo_script.1:1 at demo_script.1:vuser_init:14: vuser_init.c(14): Error -27796: Failed to connect to server "127.0.0.1:1080": [10061] Connection refused 3/16/2011 10:42:24 AM [ERROR] VUser demo_script.1:2 at demo_script.1:vuser_init:14: vuser_init.c(14): Error </pre>		

This screen indicates the status of the scenario that was reported back from LoadRunner together with any messages or other information. Because LoadRunner only reports statistics on the scenarios that was run, the test set will always be marked as passed – regardless of how long it took, unless there were errors reported. If any errors are reported, the test will be marked as Failed.

The Message of the test will report the duration the scenario took, along with the count of VUsers that reported an error, the number that reported a pass, and the hits per minute on the application.

A more detailed report will be included in the test run's details – the information above, and then any added Execution Messages and messages logged by the script in time order.

Note: LoadRunner's engine is very basic at this stage. If you have issues with a scenario not reporting or executing properly, please let Inflectra's support team know.

9. SoapUI

SmartBear SoapUI (hereafter SoapUI) is an open source Web Service testing tool for Service Oriented Architectures (SOAs). There is also a Pro version that is released as a commercial product. Its functionality mainly covers Web Service Inspection, Invoking, Development, Simulation and Mocking, Functional testing, Load and Compliance testing. Productivity enhancement features can be found in the soapUI pro version.²

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of soapUI on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated web service testing.

Note: This integration requires at least version 4.0 of SpiraTest/Team and an instance of SoapUI or SoapUI Pro running on a Windows® platform.

9.1. Installing the SoapUI Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [SoapUIEngine.zip](#) file from the Inflectra website.
- Extract the file "soapUIEngine.dll" from the compressed archive into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Automation Engine | SOAP-UI

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in bold:

Name:*

Description:

Token:*

Active

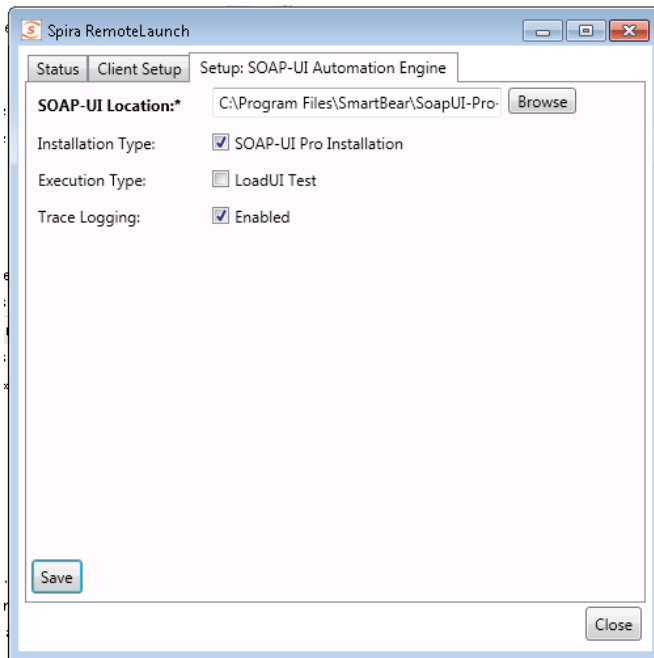
- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users. **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.

² <http://en.wikipedia.org/wiki/SoapUI>

- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For soapUI this should be **SoapUI**.
- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with SoapUI listed as an available automation engine.

9.1.1. SoapUI RemoteLaunch Settings

You will need to modify the SoapUI configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page. The SoapUI engine adds its own tab to this page which allows you to configure how SoapUI operates:



The following fields can be specified on this screen:

- **SOAP-UI Location** – This should be SOAP-UI Bin folder that contains the “TestRunner.bat” batch file that will be used to actually run the automated tests.
- **Installation Type** – This allows you to take advantage of the enhanced reporting available in the commercial “Pro” edition of SoapUI. Check the “SOAP-UI Pro Installation” box only if you are using the commercial version of SoapUI (known as SoapUI Pro).
- **Execution Type** – If this is a LoadUI performance test rather than a standard SoapUI functional test, check the box and RemoteLaunch will know to parse the load-test report format.
- **Trace Logging** – Normally this can be left unchecked unless you are diagnosing configuration issues and need additional logging.

9.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an existing SoapUI test suite and test case.

9.2.1. Linking a SoapUI Test Script

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and expand the “Automation” section of the Test Case Overview tab:

▼ Automation

This section defines the automated test script associated with this test case:

Automation Engine: SOAP-UI

Script Type*: Attached Linked Repository

Filename: [MyDocuments]\SpiraTest-4-0-Web-Service-soapui-project.xml|Requirements Testing|Get Requirements

Document Type: Functional Specification

Document Folder: Root Folder

Version: v 1.0

You need to enter the following fields:

- **Automation Engine** - Choose the SoapUI Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked for this case
- **Filename** – This needs to be the full path to the SoapUI test project XML file or composite folder together with the test suite name and test case name separated by the pipe (|) symbol. You can also pass custom command line switches as an optional final segment
 - ▷ For standard tests, you use the format:
`Project XML File|Test Suite Name|Test Case Name|Switches`
 - ▷ For composite folder tests, you use the format:
`Test Folder|Test Suite Name|Test Case Name|Switches`
 - ▷ For example if the test suite was named “Requirements Testing” and the test case was named “Get Requirements” you’d use:
`[MyDocuments]\SpiraTest-4-0-Web-Service-soapui-project.xml|Requirements Testing|Get Requirements`
 - ▷ To make this easier across different machines, you can use several constants for standard Windows locations:
 - [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - [CommonDocuments] – The Public Document’s folder.
 - [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – You can choose which document type the automated test script will be categorized under.

- **Document Folder** – You can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used when you are using the linked test script option*

Note: The example filename shown above was taken from a test project in SoapUI that has the following structure:

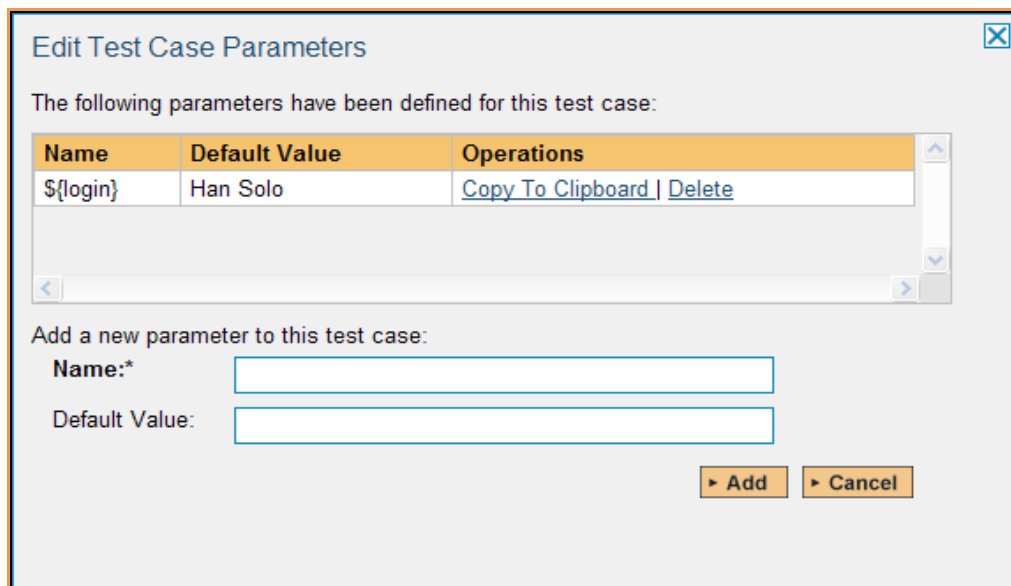


Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

9.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your SoapUI automated test. This is very useful if you have a data-driven SoapUI test that has custom project properties used that you would like to change based on the test.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter `${login}` needs to match the name of the custom parameter defined in the SoapUI project properties. Invalid parameters will be silently ignored by the SoapUI engine. Parameters must have a unique name. *Note that the plugin currently only supports “Project Properties” and not Global or System Properties.*

9.3. Executing the SoapUI Test Sets from SpiraTeam

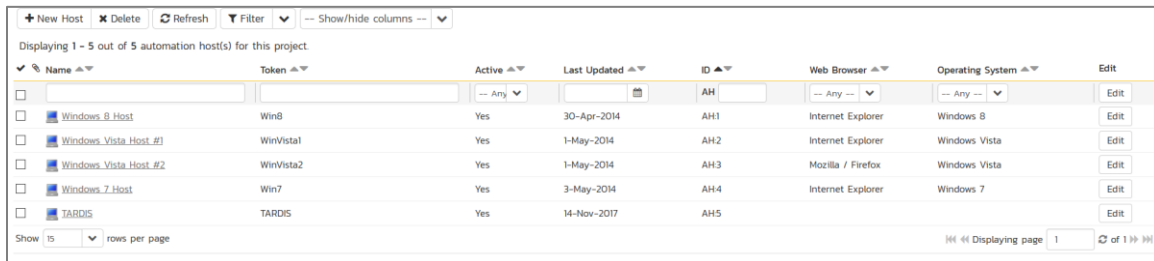
There are three ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.
3. Execute the test cases from the command-line or a build script

We shall outline each of these three scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

9.3.1. Configuring the Automation Hosts and Test Sets

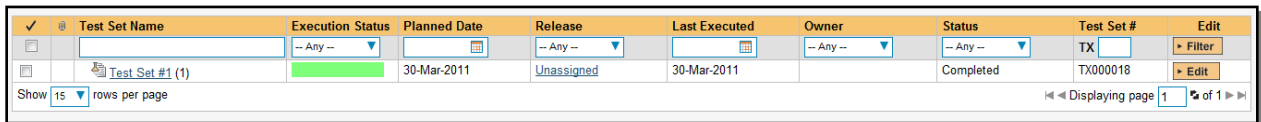
Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:



Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:



Test Set Name	Execution Status	Planned Date	Release	Last Executed	Owner	Status	Test Set #	Edit
Test Set #1 (1)	Completed	30-Mar-2011	Unassigned	30-Mar-2011		Completed	TX000018	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the SoapUI automated test cases and click on its hyperlink to display the test set details page:

Test Set: Soap UI [TX:000011]

Name: Soap UI

Overview | Test Runs | Attachments | Incidents | History

Details

Owner: -- None -- Creator: System Administrator
 Release: 1.0.0.0 - Library System Release 1 Type: Automated
 Automation Host: Windows 7 Host Creation Date: 9/3/2014 11:28:05 AM
 Status: Completed Last Executed: 9/3/2014 12:53:16 PM
 Planned Date: 09/03/2014 11:00 am -- One Time -- Last Updated: 9/3/2014 12:53:17 PM

Notes: -- Font -- -- Size -- B I U [List Icons]

Operating System: -- Please Select --

Description

Comments

Test Cases

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Dur.: 0.00 / Act. Dur.: 0.00

<input type="checkbox"/>	<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
<input type="checkbox"/>	<input type="checkbox"/>	Soap UI Test				0.0h	3-Sep-2014	Failed	TC000020	Edit

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameters

The following parameter values have been specified at the test set level: + Add Parameter Value

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	<input type="checkbox"/> Edit <input type="checkbox"/> Delete
\$(login)	librarian	-	<input type="checkbox"/> Edit <input type="checkbox"/> Delete
\$(password)	librarian	-	<input type="checkbox"/> Edit <input type="checkbox"/> Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	C-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login: admin

operatingSystem: _____

password: admin

url: _____

Save Cancel

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview Test Sets *

Properties + Dates and Times +

Detailed Information +

Test Configuration Entries

Populate Remove Refresh

Displaying 10 out of 10 entries in this test configuration set.

\$(login)	\$(browserName)	ID
admin	Internet Explorer	22
admin	Mozilla / Firefox	23
admin	Safari	25
borrower	Internet Explorer	26
borrower	Mozilla / Firefox	27
borrower	Opera	28

9.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their

list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to **immediately execute the test case on your local computer**, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

If you want to run the tests as part of a build script, just call RemoteLaunch.exe with the appropriate test set id passed into the command-line:

```
RemoteLaunch.exe -testset:18
```

In all cases, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the SoapUI test:

- **Passed** – The SoapUI automated test ran successfully and all the assertions in the test script passed
- **Failed** – The SoapUI automated test ran successfully, but at least one assertion in the test script failed.
- **Blocked** – The SoapUI automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see application or browser windows launch as the SoapUI server executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by SoapUI, you will see the following information:

Test Run: Soap UI Test [TR:000082]

Overview ▾ Attachments Incidents

▼ Details

Release #:	1.0.0.0 - Library System Release 1	Estimated Duration:	0.00 hours
Tester Name:*	System Administrator	Actual Duration:	0.00 hours
Test Set:	Soap UI	Execution Date:	9/3/2014 12:53:16 PM
Test Case #:*	TC000020	Execution Status:*	Failed
Build:	-- None --	Test Run Type:*	Automated

This screen indicates the status of the test run that was reported back from SoapUI together with any messages or other information. The execution status will be set according to the worst-case assessment reported back from SoapUI. If you have zero(0) failures, then the status will display as Passed, otherwise it will display as Failed.

Under **Console Output** section you will see more detailed logging information (in both SoapUI and SoapUI Pro):

Runner Name: SOAP-UI Automation E Assert Count: 0
Automation Host: Windows 7 Host Test Name: Simple TestSuite / Simple Search TestCase
Message: 3 test steps completed with 3 request assertions, 0 failed assertions.

Details:

```

SoapUI Pro 5.1.1 TestCase Runner
Configuring log4j from [C:\Program Files\SmartBear\SoapUI-Pro-5.1.1\bin\soapui-log4j.xml]
12:53:05,048 INFO [DefaultSoapUICore] initialized soapui-settings from [C:\Users\adam.sandman\soapui-settings.xml]
12:53:09,931 INFO [SoapUIProGroovyScriptEngineFactory] Setting Script Library to [C:\Program Files\SmartBear\SoapUI-Pro-5.1.1\bin\scripts]
12:53:09,931 INFO [DefaultSoapUICore] Adding listeners from [C:\Program Files\SmartBear\SoapUI-Pro-5.1.1\bin\listeners\demo-listeners.xml]
12:53:12,739 INFO [WsdlProject] Loaded project from [file:/C:/Users/adam.sandman/SoapUI-Tutorials/Sample-SOAP-Project-soapui-project.xml]
12:53:13,098 INFO [SoapUIProGroovyScriptEngineFactory] Setting Script Library to [C:\Program Files\SmartBear\SoapUI-Pro-5.1.1\bin\scripts]
12:53:13,956 INFO [SoapUIProTestCaseRunner] Running SoapUI tests in project [Sample SOAP Project Pro]
12:53:13,987 INFO [SoapUIProTestCaseRunner] Running TestCase [Simple Search TestCase]
12:53:14,018 INFO [SoapUIProTestCaseRunner] Running SoapUI testcase [Simple Search TestCase]
12:53:14,034 INFO [SoapUIProTestCaseRunner] running step [Properties - Username and Password]
12:53:14,112 INFO [SoapUIProTestCaseRunner] running step [Property Transfer - Move Username and Password]
12:53:14,720 INFO [SoapUIProTestCaseRunner] running step [Test Request - login]
12:53:16,374 DEBUG [SoapUIMultiThreadedHttpConnectionManager$SoapUIDefaultClientConnection] Connection closed
12:53:16,374 DEBUG [SoapUIMultiThreadedHttpConnectionManager$SoapUIDefaultClientConnection] Connection shut down
12:53:16,374 ERROR [WsdlSubmit] Exception in request: org.apache.http.conn.HttpHostConnectException: Connection to http://127.0.0.1:8088 refused
12:53:16,374 ERROR [SoapUI] An error occurred [Connection to http://127.0.0.1:8088 refused], see error log for details
12:53:16,421 INFO [SoapUIProTestCaseRunner] Assertion [SOAP Response] has status UNKNOWN
12:53:16,421 INFO [SoapUIProTestCaseRunner] Assertion [Schema Compliance] has status UNKNOWN

```

The Message field will contain a summary of the number of test steps completed, the number of assertions and the number of failed assertions. The Details field will contain the detailed trace of what happened, captured from the summary output log that is generated by SoapUI.

SoapUI Pro

If you have the commercial SoapUI Pro product and have configured RemoteLaunch so that it knows to use SoapUI Pro, in addition, the **Test Steps** section of the test run will contain more detailed reporting:

▼ Test Steps						
ID	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
RS000197	Step 1 [Properties - Username and Password] OK: took 72 ms			/	OK > View Incidents	Passed
RS000198	Step 2 [Property Transfer - Move Username and Password] OK: took 577 ms -> Performed transfer [TransferUser] -> Performed transfer [TransferPass]			/	OK > View Incidents	Passed
RS000199	Step 3 [Test Request - login] FAILED: took 1696 ms -> org.apache.http.conn.HttpHostConnectException: Connection to http://127.0.0.1:8088 refused			/	FAILED > View Incidents	Failed

Where each test step corresponds to a step recorded in the SoapUI Pro results file.

Congratulations... You are now able to run SoapUI automated web-service tests and have the results be recorded within SpiraTest / SpiraTeam.

10. FitNesse

FitNesse is a lightweight, open-source automated software testing framework that uses web-based Wikis to define the inputs and expected results from different combinations of input values and then compare the results with what is actually generated during testing. For more details on FitNesse, please refer to the FitNesse website: <http://fitnesse.org>

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of FitNesse on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated FitNesse acceptance tests.

Note: This integration requires at least version 4.0 of SpiraTest/Team and RemoteLaunch.

10.1. Installing the FitNesse Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [FitNesseEngine.zip](#) file from the Inflectra website and *copy the file "FitNesseEngine.dll" into the "extensions" sub-folder of the RemoteLaunch installation.*
- You may also need to verify that you have the full Microsoft .NET Framework 4.0 installed since that is needed by the FitNesse engine. RemoteLaunch itself only needs the .NET 4.0 Client Profile, so make sure you have the .NET 4.0 Framework Extended entry listed in the Program & Features section of the Windows Control Panel.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Automation Engine | FitNesse
[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in bold:

Name:

Description:

Token:

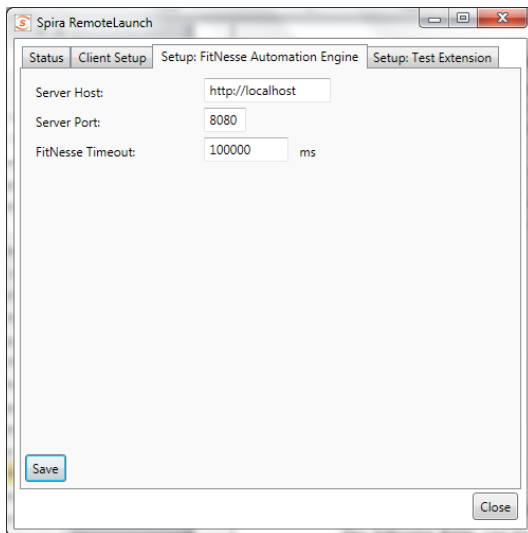
Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For FitNesse this should be simply **FitNesse**.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with FitNesse listed as an available automation engine.

10.1.1. Advanced Settings

You can modify the FitNesse configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page. The FitNesse engine adds its own tab to this page which allows you to configure how FitNesse operates:



The following fields can be specified on this screen:

- **Server Host** – This should be the base URL for accessing the installation of FitNesse. Each of the FitNesse test cases will be a URL relative to this base URL.
- **Server Port** – This should be set to the TCP port that the FitNesse web server uses for displaying the FitNesse wiki web pages.
- **FitNesse Timeout** – This allows you to extend the timeout for executing FitNesse tests. This is useful if you find that the FitNesse tests take a long time to execute and RemoteLaunch is aborting the execution before they are finished.

10.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an existing FitNesse test case wiki page. Note: The FitNesse automation engine only supports Linked test scripts in SpiraTeam (not Attached).

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and go to the “Automation” section located in the “Overview” tab:

Automation

This section defines the automated test script associated with this test case:

Automation Engine: FitNesse

Script Type*: Attached Linked Repository

Filename:

Document Type: Functional Specification

Document Folder: Root Folder

Version: v

Test Script:

You need to enter the following fields:

- **Automation Engine** - Choose the FitNesse Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked for FitNesse tests.
- **Filename** – This needs to be the relative URL of the FitNesse test case. I.e. if the FitNesse URL is <http://myserver/FitNesse.UserGuide.TwoMinuteExample> and the base URL setup in RemoteLaunch is <http://myserver> then the “filename” would be just `FitNesse.UserGuide.TwoMinuteExample`.
- **Document Type** – You can choose which document type the automated test script will be categorized under.
- **Document Folder** – You can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used when you are using the linked test script option*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

10.2.1. Using Parameterized Test Cases

The FitNesse automation engine does not currently support the passing of parameter values from SpiraTeam to the FitNesse test.

10.3. Executing the FitNesse Test Sets from SpiraTeam

There are three ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.
3. Execute the test cases from the command-line or a build script

We shall outline each of these three scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

10.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Name	Execution Status	Planned Date	Last Executed	Owner	Status	ID	Edit
Functional Test Sets		4-Feb-2007			In Progress	TX000008	Edit
Testing Cycle for Release 1.0 (7)		4-Feb-2007	1-Dec-2003	Joe P Smith	In Progress	TX000001	Edit
Testing Cycle for Release 1.1 (9)		6-Feb-2007	1-Dec-2003	Joe P Smith	Not Started	TX000002	Edit
Testing New Functionality (4)		9-Feb-2007		Fred Bloggs	In Progress	TX000005	Edit
Exploratory Testing (2)				Fred Bloggs	Deferred	TX000006	Edit
Regression Test Sets					Completed	TX000009	Edit
Regression Testing for Windows 8 (4)				Fred Bloggs	Completed	TX000003	Edit
Regression Testing for Windows Vista (4)					Completed	TX000004	Edit
FitNesse Tests (1)		3-Oct-2014	3-Oct-2014		Completed	TX000010	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the FitNesse automated test cases and click on its hyperlink to display the test set details page:

Test Set: FitNesse Tests [TX:000010]

Name:

Overview | Test Runs | Attachments | Incidents | History

Details

Owner: -- None -- | Creator*: System Administrator

Release: -- None -- | Type: Automated

Automation Host: Tardis | Creation Date: 10/3/2014 9:40:30 AM

Status*: Not Started | Last Executed: 10/3/2014 10:15:03 AM

Planned Date: 10/03/2014 9:00 am | -- One Time -- | Last Updated: 10/3/2014 10:15:09 AM

Notes:

Operating System: -- Please Select --

Description

Comments

Test Cases

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests | Est. Dur.: 0.00 / Act. Dur.: 0.00

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
Two Minute Example				0.0h	3-Oct-2014	Failed	TC000018	Edit

Show 15 rows per page | Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

10.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be executed. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the FitNesse test:

- **Passed** – The FitNesse automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The FitNesse automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The FitNesse automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see command windows appear as the FitNesse server executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by FitNesse, you will see the following information:

Test Run: Two Minute Example [TR:000022]

Overview | Attachments | Incidents

Details

Release #: -- None -- Estimated Duration: _____ hours

Tester Name:* System Administrator Actual Duration: 0.00 hours

Test Set: FitNesse Tests Execution Date: 10/3/2014 10:15:03 AM

Test Case #:* TC000018 Execution Status:* **Failed**

Build: _____ Test Run Type:* Automated

This screen indicates the status of the test run that was reported back from FitNesse together with any messages or other information. The execution status will be set to PASSED if all the FitNesse rows report back OK and all the tests passed. If any of the rows failed or the tests don't pass, the overall execution status will be listed as FAILED.

You can see a step-by-step record of what happened by scrolling down to the "Test Steps" section:

Test Steps

ID	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
RS000041	decisionTable_0_0		{id='decisionTable_0_0', instruction='make', instanceName='decisionTable_0', className='eg.Division', args=[]}	/		Passed
RS000042	decisionTable_0_4		{id='decisionTable_0_4', instruction='call', instanceName='decisionTable_0', methodName='setNumerator', args=[10]}	/		Not Run
RS000043	decisionTable_0_5		{id='decisionTable_0_5', instruction='call', instanceName='decisionTable_0', methodName='setDenominator', args=[2]}	/		Not Run
RS000044	decisionTable_0_7		{id='decisionTable_0_7', instruction='call', instanceName='decisionTable_0', methodName='quotient', args=[]}	/		Passed
RS000045	decisionTable_0_9		{id='decisionTable_0_9', instruction='call', instanceName='decisionTable_0', methodName='setNumerator', args=[12.6]}	/		Not Run
RS000059	decisionTable_0_32	33	{id='decisionTable_0_32', instruction='call', instanceName='decisionTable_0', methodName='quotient', args=[]}	/	25.0 > View Incidents	Failed

In addition, you can scroll down to the "Console Output" section to get the FitNesse specific information:

Console Output

Runner Name: FitNesse Automation Assert Count: 0

Automation Host: Tardis Test Name: TwoMinuteExample

Message: Test completed with: 5 right, 1 wrong, 0 ignored and 0 exceptions

Details:

- Expand
- Collapse

Hidden

variable defined: TEST_SYSTEM=slim
[A One-Minute Description](#)

An Example FitNesse Test

If you were testing the division function of a calculator application, you might like to see some examples working. You might want to see what you get back if you ask it to divide 10 by 2. (You might be hoping for a 5!)

In FitNesse, tests are expressed as tables of **input** data and **expected output** data. Here is one way to specify a few division tests in FitNesse:

eg.Division

The Message field will contain a summary of the number of tests executed and the number of wrong results and exceptions. The large details box contains the full command execution log as reported back from FitNesse:

Congratulations... You are now able to run FitNesse automated acceptance tests and have the results be recorded within SpiraTest / SpiraTeam.

11. NeoLoad

Neotys NeoLoad is a performance and load testing system that lets you record application performance by a number of 'virtual users' and measure the performance against specified Service Level Agreement (SLA) metrics for the application. When you use NeoLoad with SpiraTest you can report back pass/fail/caution by comparing the actual results against the specified SLA metrics.

This section covers installing and using the Engine to report back statistics of run scenarios as well as the results of the test compared to the required SLAs.

Note: This integration requires at least version 4.0 of SpiraTest/Team and has been tested against version 5.0 of NeoLoad.

11.1. Installing the NeoLoad Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [NeoLoadEngine.zip](#) file from the Inflectra website.
- Copy the files in the ZIP file into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

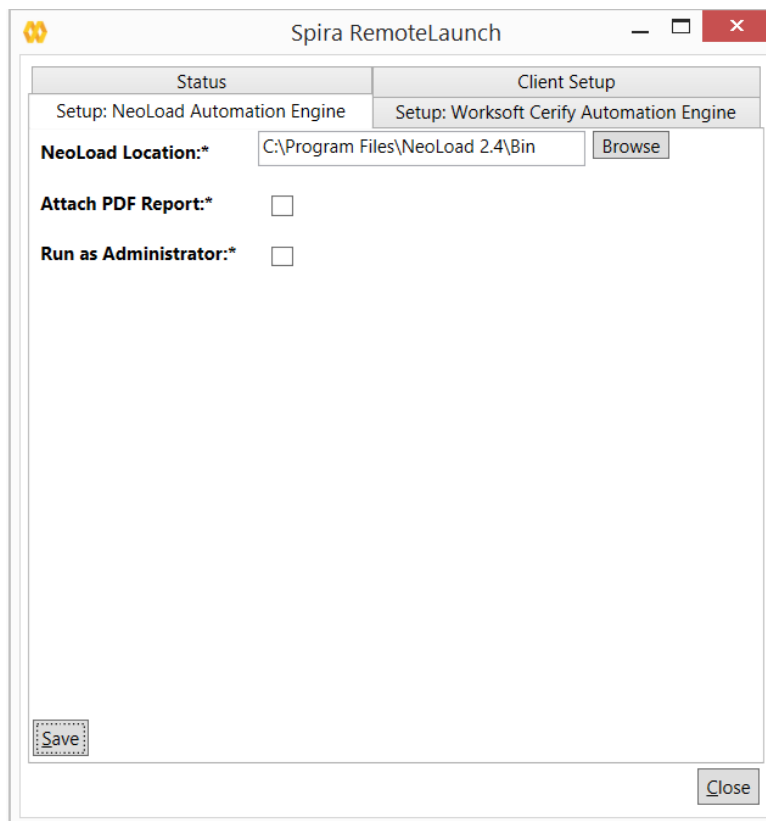
The screenshot shows the 'NeoLoad Edit Automation Engine' form. At the top left is the 'NeoLoad' logo and the text 'Edit Automation Engine'. Below this is a button labeled '← Back to Test Automation Engine Home'. The main instruction reads: 'Please enter/edit the following information for the test automation engine.' The form contains four fields: 'Name' with a red asterisk and the value 'NeoLoad'; 'Description' with a text area containing 'Engine that integrates with NeoTys NeoLoad'; 'Token' with a red asterisk and the value 'NeoLoad'; and 'Active' with a checked checkbox labeled 'Yes'. At the bottom are two buttons: 'Save' (orange) and 'Cancel' (white).

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users, and will be displayed in the dropdown when the user selects the Tester.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For NeoLoad, it needs to be simply "**NeoLoad**".

Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with NeoLoad listed as an available automation engine.

11.1.1. NeoLoad RemoteLaunch Settings

You will need to modify the NeoLoad configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page. The NeoLoad engine adds its own tab to this page which allows you to configure how NeoLoad operates:



The following fields can be specified on this screen:

- **NeoLoad Location** – This should be folder containing the “NeoLoadCmd.exe” executable that will be used to actually run the automated tests.
- **Attach PDF Report** – NeoLoad has a built-in report generator that can create detailed Acrobat (PDF) format reports. Enabling this option will attach these reports to the test runs recorded in SpiraTeam.
- **Run as Administrator** – Sometimes NeoLoad needs to be run as a Windows elevated process, in which case, choose the “Run as Administrator” option.

11.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to a NeoLoad project and scenario.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and go to the “Automation” section in the main “Overview” tab:

Automation

This section defines the automated test script associated with this test case:

Script Type: Attached Linked Repository

Automation Engine: NeoLoad Version: 1.0

Filename: [MyDocuments]\Neoload Projects\InflectraWebsite\InflectraWebsite.nlp\Website clicks Document Type: Functional Specification

Document Folder: Root Folder

Test Script:

You need to enter the following fields:

- **Automation Engine** - Choose the NeoLoad Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – For NeoLoad, all scenarios must be stored on the local testing machine so 'Linked' must be selected. If you select 'Attached', when the scenario is attempted to be executed it will be marked as blocked and skipped.
- **Filename** – This needs to be the full path to the NeoLoad project file (*.nlp) file followed by the name of the NeoLoad scenario. The two components need to be separated by a pipe (|) character.

Certain tokens are allowed to be able to specify common locations across different operating systems. Note that the tokens are case-sensitive, and there are no spaces in them. A list of tokens are:

- [MyDocuments] – The user's "My Documents" folder. The user indicated is the user that ran RemoteLaunch.
- [CommonDocuments] – The Public Document's folder.
- [DesktopDirectory] – The user's Desktop folder. The user indicated is the user that ran RemoteLaunch.
- [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it's the 64-bit directory.
- [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – You can choose which document type the automated scenario will be categorized under.
- **Document Folder** – You can choose which document folder the automated scenario will be stored in.
- **Version** – The version of the scenario (1.0 is used if no value specified)
- **Test Script** – *Not used.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

11.2.1. Using Parameterized Test Cases

Currently the NeoLoad automation engine does not support the passing of parameter values from SpiraTeam to NeoLoad.

11.3. Executing the NeoLoad Scenario from SpiraTeam

There are three ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.
3. Execute the test cases from the command-line or a build script

We shall outline each of these three scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

11.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified as the Host name in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Release	Last Executed	Owner	Status	Test Set #	Edit
Test Set #1 (2)	Completed	30-Mar-2011	Unassigned	1-Apr-2011		Completed	TX000018	Edit
Selenium Testing (1)	Completed	4-Apr-2011		4-Apr-2011		Completed	TX000019	Edit
FitNesse Testing (1)	Completed			4-Apr-2011		Completed	TX000020	Edit
NeoLoad Testing (1)	Completed			5-Apr-2011		Completed	TX000021	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the NeoLoad test cases and click on its hyperlink to display the test set details page:

Test Set: Load Testing [TX:000017]

Name: Load Testing

Overview # | Test Runs # | Attachments | Incidents | History #

Details

Owner: -- None -- | Creator*: System Administrator

Release: 1.0.0.0 - Release 1.0 | Type: Automated

Automation Host: Dalek | Creation Date: 1/7/2015 4:21:12 PM

Status*: Not Started | Last Executed: 1/8/2015 12:29:05 PM

Planned Date: 01/07/2015 4:00 pm | Last Updated: 1/8/2015 12:29:06 PM

Description

Comments

Test Cases

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests | Est. Dur.: 0.00 / Act. Dur.: 0.03

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
<input type="checkbox"/>	Simple Load Test of Inflectra Website				0.0h	8-Jan-2015	Failed	TC000053	Edit

Show 15 rows per page | << Displaying page 1 of 1 >>

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

11.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be executed. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the NeoLoad execution:

- **Passed** – The scenario ran and reported no error messages and all SLAs were passed.
- **Caution** – The scenario ran and at least one SLA reported back as acceptable
- **Failed** – The scenario ran and at least one error message was reported or at least one SLA was reported back as failed.
- **Blocked** – There was an error with the Test Set or NeoLoad application.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser or application windows launch as NeoLoad runs the scenario and connects VUsers to their tasks.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by NeoLoad, you will see the following test run summary information:

Details

Release #: 1.0.0.0 - Release 1.0 Estimated Duration: hours

Tester Name: System Administrator Actual Duration: 0.03 hours

Test Set: Load Testing Execution Date: 1/8/2015 12:29:05 PM

Test Case #: TC000053 Execution Status: **Failed**

Build: -- None -- Test Run Type: Automated

Test Steps

ID	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
RS000304	avg_hits/s			/	6.6 > View Incidents	N/A
RS000305	avg_pages/s			/	2.1 > View Incidents	N/A
RS000306	avg_reqresponsetime			/	0.07s > View Incidents	N/A
RS000307	avg_pageresponsetime			/	0.21s > View Incidents	Failed
RS000308	avg_throughput			/	1.64 Mb/s > View Incidents	N/A
RS000309	total_pages			/	260 > View Incidents	N/A
RS000310	total_hits			/	800 > View Incidents	N/A
RS000311	total_users_launched			/	10 > View Incidents	N/A
RS000312	total_throughput			/	24.8 MB > View Incidents	N/A
RS000313	total_iterations_completed			/	50 > View Incidents	N/A
RS000314	total_errors			/	0 > View Incidents	N/A
RS000315	error_percentile			/	0% > View Incidents	N/A
RS000316	total_logical_actions_errors			/	0 > View Incidents	N/A
RS000317	total_alerts_percentage			/	76.7% > View Incidents	N/A

This section of the screen indicates how long the test took to execute, the overall status, which release was being executed, which test set it was a part of and each of the key summary statistics, together with information on how they compared to the defined SLA:

- **N/A** – There was no SLA defined for this metric
- **Passed** – There is an SLA defined for this metric and it was passed.
- **Caution** – There is an SLA defined for this metric and it was considered less than a pass, but still acceptable.
- **Failed** – There is an SLA defined for this metric and it was not met successfully.

In addition, if you scroll down, in the “Console Output” section of the report there is more detailed information:

Console Output

Runner Name: NeoLoad Automation E Assert Count: 0
Automation Host: Dalek Test Name: InflectraWebsite / Website clicks
Message: 260 total pages, 800 total hits, 10 total users, 0 hit errors, 0 action errors

Details:

Top 5 errors

Top 5 alerts

1. Average Response Time : Actions >= 0.2 sec. - 1

Top 5 average response times

1. / - 0.418
2. /SpiraTest/ - 0.221
3. /SpiraTeam/ - 0.212
4. /_1 - 0.149
5. /_utm.gif - 0.143

Top 5 maximum response times

1. / - 1.11
2. /SpiraTest/ - 0.328
3. /_1 - 0.328
4. /SpiraTeam/ - 0.266

The Message of the test will report the number of total pages, number of total hits, number of total users, number of errors as well as the total count of virtual users.

In addition, more detailed information is displayed in the test run details:

- Top 5 errors by page
- Top 5 alerts by page
- Top 5 average response times by page
- Top 5 maximum response times by page

Finally, if you have chosen the option to attach the NeoLoad PDF report, in the Attachments section of the Test Run, that will be listed:

Overview | **Attachments** | Incidents

> Add New | Add Existing | Remove | Refresh | Apply Filter | Clear Filter | Include Source Code Documents

Displaying 1 - 1 out of 1 attachment(s).

Document Name ▲▼	Type ▲▼	Size ▲▼	Edited By ▲▼	Edited On ▲▼	Author ▲▼	ID ▲▼
<input type="checkbox"/> <input type="text"/>	-- Any --	<input type="text"/>	-- Any --	<input type="text"/>	-- Any --	DC <input type="text"/>
<input type="checkbox"/> NeoLoad-Detailed-Report.pdf	Default	375 KB	System Administrator	8-Jan-2015	System Administrator	DC000074

Show 15 rows per page Displaying page 1 of 1

Congratulations... You are now able to run automated NeoLoad performance scenarios and have the results be recorded within SpiraTest / SpiraTeam.

12. TestPartner

Micro Focus™ TestPartner™ (hereafter TestPartner) is a Graphic User Interface (GUI) functional test automation system that lets you record application operations by capturing the various testable objects of the application and then playback the operations to automatically test the application.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of TestPartner on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated TestPartner tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 6.0 of TestPartner.

12.1. Installing the TestPartner Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [TestPartnerAutomationEngine.zip](#) file from the Inflectra website and locate the TestPartner.dll inside the zip archive.
- Copy the file "TestPartner.dll" into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Test Partner
[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

Active

- **Name**: This is the short display name of the automation engine. It can be anything that is meaningful to your users.
 - **Description**: This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
 - **Active**: If checked, the engine is active and able to be used for any project.
 - **Token**: This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For TestPartner this should just be **TestPartner**.
- Once you have finished, click the "Insert & Close" button and you will be taken back to the Test Automation list page, with TestPartner listed as an available automation engine.

12.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated TestPartner test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

All	Name	Project	Creator	Creation Date	Last Modified By	Modified Date	Description	Version
Visual test	Notepad	Common	Admin	4/29/2011 10:18:57 AM	Admin	4/29/2011 10:19:39 AM		1
Test script								
Result								

You need to enter the following fields:

- **Automation Engine** - Choose the TestPartner Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with TestPartner only supports referencing TestPartner test scripts (stored in the internal database) and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This needs contain the project and test name from TestPartner with the appropriate parameter name describing which is the project name and which is the test name. The test name can be either a test script of a visual test. The syntax is:
 - ▷ `-visualtest <test name> -project <project name>` OR
 - ▷ `-testscript <script name> -project <project name>`
- **Document Type** – If using SpiraTeam (not SpiraTest) you can choose which document type the automated test script will be categorized under.
- **Document Folder** – If using SpiraTeam (not SpiraTest) you can choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the TestPartner Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

12.2.1. Using Parameterized Test Cases

TestPartner does not support the passing of input test parameters so the TestPartner automation engine does not support this feature of SpiraTeam or RemoteLaunch.

12.3. Executing the TestPartner Test Sets from SpiraTeam

There are three ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.
3. Execute the test cases from the command-line or a build script

We shall outline each of these three scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

12.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows_7_Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
TC 7.0 Testing (1)	In Progress	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	Completed	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmarteScript Testing (1)	Completed	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	Completed	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	Completed	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	Completed	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the TestPartner automated test cases and click on its hyperlink to display the test set details page:

Test Set: TP Test Set [TX:00026]

Name*: TP Test Set

Description:

Owner: -- None -- **Creator*:** System Administrator

Release: -- None --- **Type*:** Automated

Automation Host: VM2 **Created On:** 4/29/2011 3:06:08 PM

Status*: Completed **Last Executed:** -

Planned Date: 4/29/2011 03:20:00 PM **Last Updated:** 4/29/2011 3:53:02 PM

Test Cases * | Test Runs * | Comments | Custom Props * | Attachments | History *

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Duration: 0.0h / Actual Duration: 0.0h

	Test Case Name	Owner	Priority	Est. Duration	Act. Duration	Last Executed	Execution Status	Test Case #	Edit
<input type="checkbox"/>	Notepad Test				0.0h	29-Apr-2011	Passed	TC000039	<input type="button" value="Edit"/>

Show 15 rows per page | Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

12.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the TestPartner test:

- **Passed** – The TestPartner automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The TestPartner automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The TestPartner automated test did not run successfully

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you may see browser or application windows launch as TestPartner executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by TestPartner, you will see the following information:

Test Set: TP Test Set	Execution Date: 4/29/2011 3:52:44 PM
Test Case #: TC000039	Execution Status: Passed
Automation Host: VM2	Test Run Type: Automated

Test Run Steps	Automation *	Custom Properties *	Attachments
----------------	--------------	---------------------	-------------

Runner Name: Micro Focus TestPart	Assert Count: 0
Message: 15 Commands Executed, 15 Commands Passed, 0 Command Errors	Test Name: Unknown?

Details:
<pre>1 - <<Start>> > RESULT: OK 2 - Using 'Index=1' > RESULT: OK 3 - Click 'Caption=start' at 40, 19 > RESULT: OK 4 - Using 'Start Menu Window' > RESULT: OK 5 - Select 'Run...' > RESULT: OK 6 - Using 'Run Window' > RESULT: OK 7 - Set text to 'notepad' > RESULT: OK 8 - Click 'Caption=OK' > RESULT: OK 9 - Using 'Untitled - Notepad Window' > RESULT: OK 10 - Click 'Index=1' at 316, 104 > RESULT: OK 11 - Enter 'This is a test matey' > RESULT: OK 12 - Close > RESULT: OK 13 - Using 'Notepad Window' > RESULT: OK 14 - Click 'Caption=&No' > RESULT: OK 15 - <<End>> > RESULT: OK</pre>

This screen indicates the status of the test run that was reported back from TestPartner together with any messages or other information. The Test Name indicates the name of the test inside TestPartner, and the execution status corresponds the matching status inside TestPartner.

Congratulations... You are now able to run TestPartner automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

13. BadBoy

Badboy is an automated website functional test automation system that lets you record website operations in Internet Explorer and generate test automation scripts that can be used to playback the test script against the website.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of Badboy on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated Badboy tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 2.1 of Badboy.

13.1. Installing the Badboy Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [BadboyAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate BadboyX.dll for the version of Badboy that you are using.
 - If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "*BadboyX.dll*" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Bad Boy
[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in bold:

Name*:

Description:

Token*:

Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For

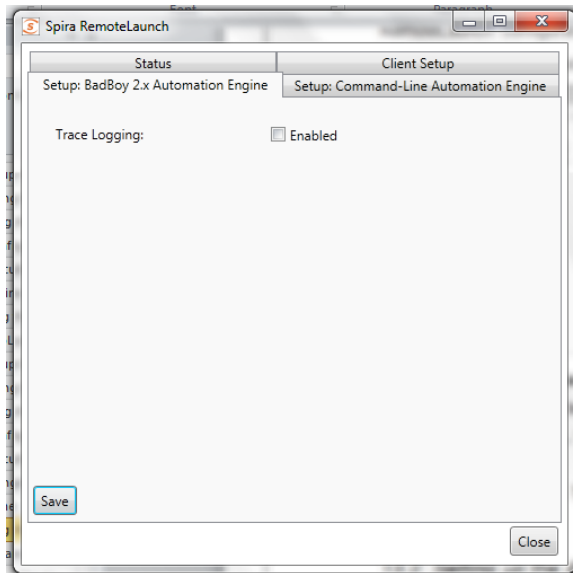
Badboy this should be **BadboyX** where 'X' is the version number of the DLL file that you are using.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Badboy listed as an available automation engine.

13.1.1. Advanced Settings

You can modify the Badboy configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The Badboy 2.x engine adds its own tab to this page which allows you to configure how Badboy operates:



The following fields can be specified on this screen:

- **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.

13.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated Badboy test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

The screenshot shows the 'Automation' tab in a software interface. It contains the following fields and options:

- Automation Engine*:** A dropdown menu with 'Bad Boy' selected.
- Script Type*:** Radio buttons for 'Attached' and 'Linked' (which is selected).
- Filename*:** A text input field containing '[MyDocuments]BadBoy-SampleScript.bb'.
- Document Type*:** A dropdown menu with 'Default' selected.
- Document Folder*:** A dropdown menu with 'Root Folder' selected.
- Version:** A text input field with 'v' followed by a blank space.
- Test Script*:** A large empty text area.

At the bottom right of the form, there is a link that says '> Edit Parameters'.

You need to enter the following fields:

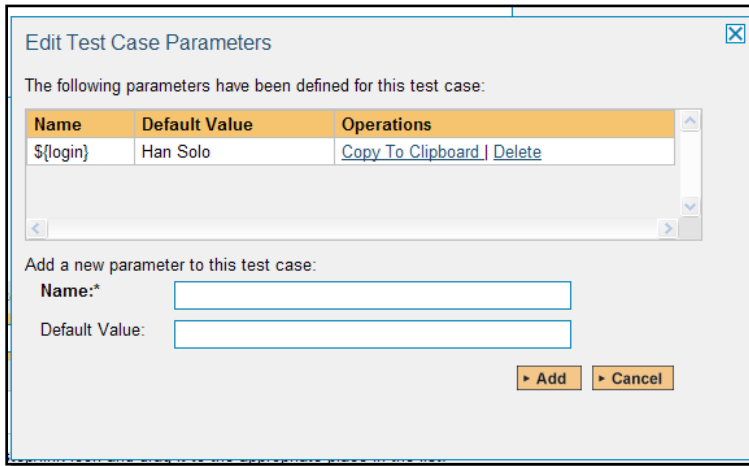
- **Automation Engine** - Choose the Badboy Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with Badboy only supports referencing Badboy test script files and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This needs to be the full path to the Badboy test script. To make this easier across different machines, you can use several constants for standard Windows locations (see example in screenshot):
 - ▷ [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [CommonDocuments] – The Public Document’s folder.
 - ▷ [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - ▷ [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
 - ▷ [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
- **Document Type** – This allows you to choose which document type the automated test script will be categorized under.
- **Document Folder** – This allows you to choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the Badboy Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

13.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your Badboy automated test script. This is very useful if you have a data-driven Badboy test script that defines input variables from an external data source.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter \${login} needs to match the name of the variable defined within the Badboy script in its variables configuration.

13.3. Executing the Badboy Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

13.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Displaying 1 - 5 out of 5 automation host(s) for this project

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows_7_Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Show 15 rows per page

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

✓	Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
<input type="checkbox"/>	<input type="text" value=""/>	-- Any --	<input type="text" value=""/>	<input type="text" value=""/>	-- Any --	-- Any --	-- Any --	TX <input type="text" value=""/>	<input type="button" value="Filter"/>
<input type="checkbox"/>	TC 7.0 Testing (1)	 	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	<input type="button" value="Edit"/>
<input type="checkbox"/>	QTP Testing (1)	 	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	<input type="button" value="Edit"/>
<input type="checkbox"/>	SmartScript Testing (1)	 	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	<input type="button" value="Edit"/>
<input type="checkbox"/>	Selenium Testing (3)	 	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	<input type="button" value="Edit"/>
<input type="checkbox"/>	Squish Testing (3)	 	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	<input type="button" value="Edit"/>
<input type="checkbox"/>	Command Line Testing (1)	 	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	<input type="button" value="Edit"/>

Show 15 rows per page Displaying page 1 of 1

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Badboy automated test cases and click on its hyperlink to display the test set details page:

Test Set: [BadBoy Tests \[TX:000012\]](#)

Name*:

Description: **B** *I* U

Owner: Creator*:

Release: Type*:

Automation Host: Creation Date: 8/29/2011 12:21:42 PM

Status*: Last Executed: -

Planned Date: Last Updated: 8/29/2011 12:21:58 PM

Est. Dur.: 0.0h / Act. Dur.: 0.0h

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
BadBoy Test Case						Not Run	TC000024	<input type="button" value="Edit"/>

Show 15 rows per page Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
\$(login)	librarian	-	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
\$(password)	librarian	-	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login: admin

operatingSystem: _____

password: admin

url: _____

Save **Cancel**

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview | Test Sets *

Properties | Dates and Times

Detailed Information

Test Configuration Entries

+ Populate | - Remove | Refresh

Displaying 10 out of 10 entries in this test configuration set

\$(login)	\$(browserName)	ID
admin	Internet Explorer	22
admin	Mozilla / Firefox	23
admin	Safari	25
borrower	Internet Explorer	26
borrower	Mozilla / Firefox	27
borrower	Opera	28

13.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the Badboy test:

- **Passed** – The Badboy automated test ran successfully and all the test steps in the test script passed and no assertions were thrown.
- **Failed** – The Badboy automated test ran successfully, but at least one test step failed or at least one assertion failed.
- **Caution** – The Badboy automated test run successfully, but at least one warning was logged in one of the test steps.
- **Blocked** – The Badboy automated test did not run successfully or at least one timeout error was recorded.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see browser windows launch as Badboy executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by Badboy, you will see the following information:

Test Run: Sample Test [TR:000060]

Release #: -- None -- Estimated Duration: hours

Tester Name:* System Administrator Actual Duration:* 0.0 hours

Test Set: Sample Test Set Execution Date: 8/28/2011 5:46:00 PM

Test Case #:* TC000014 Execution Status:* **Passed**

Automation Host: Tardis Test Run Type:* Automated

Test Run Steps **Automation *** Custom Props Attachments

Runner Name: BadBoy 2.x Automatio Assert Count: 0

Message: 12 played, 12 succeeded, 0 failures, 0 assertions, 0 warnings, 0 timeouts, maxResponseTime: 2495, averageResponseTime: 1267 Test Name: BadBoy-SampleScript

Details:

```

Suite: Test Suite 1
=====
Test: Test 3
-----
12 played, 12 succeeded, 0 failures, 0 assertions, 0 warnings, 0 timeouts-----
Step: Step 2
-----
12 played, 12 succeeded, 0 failures, 0 assertions, 0 warnings, 0 timeouts-----

```

This screen indicates the status of the test run that was reported back from Badboy together with any messages or other information. The Test Name indicates the name of the test inside Badboy and the execution status corresponds the matching status inside Badboy as illustrated below:

Badboy Status	SpiraTeam Status
Succeeded	Passed
Failure	Failed
Warning	Caution
Assertion	Failed
Timeout	Blocked

In addition, the detailed test report from Badboy is available in the large text-box below. It will contain messages such as:

```

Suite: Test Suite 1
=====
Test: Test 3
-----
12 played, 12 succeeded, 0 failures, 0 assertions, 0 warnings, 0 timeouts
-----
Step: Step 2
-----
12 played, 12 succeeded, 0 failures, 0 assertions, 0 warnings, 0 timeouts
-----

```

Congratulations... You are now able to run Badboy automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

14. JMeter

Apache JMeter is a free, open source Java desktop application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

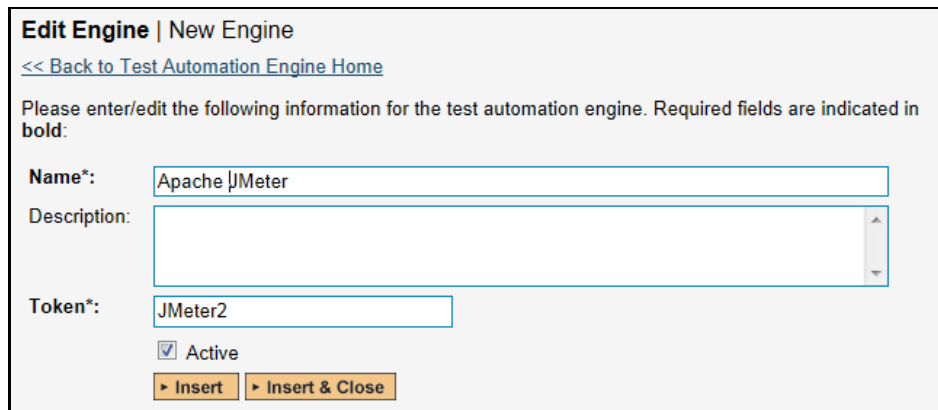
This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of JMeter on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated JMeter performance tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 2.5 of JMeter.

14.1. Installing the JMeter Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [JMeterEngine.zip](#) file from the Inflectra website and locate the appropriate JMeterX.dll for the version of JMeter that you are using.
 - If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "JMeterX.dll" (where X is the appropriate version) into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:



Edit Engine | New Engine
[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in bold:

Name*:

Description:

Token*:

Active

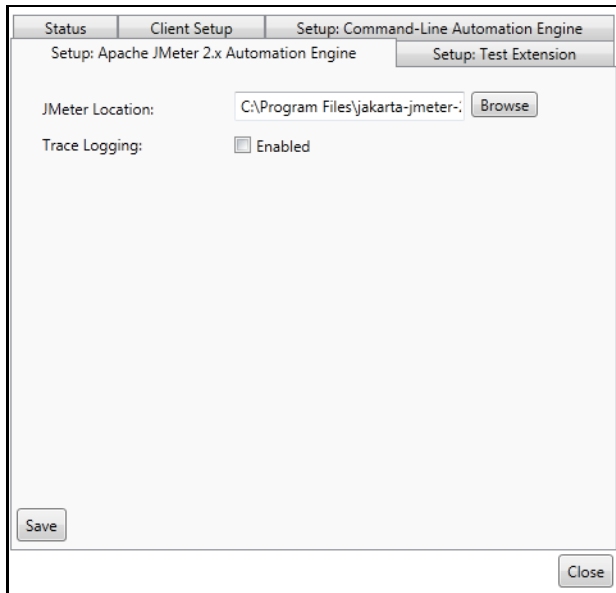
- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For JMeter this should be **JMeterX** where 'X' is the version number of the DLL file that you are using.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with JMeter listed as an available automation engine.

14.1.1. Advanced Settings

You can modify the JMeter configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The JMeter 2.x engine adds its own tab to this page which allows you to configure how JMeter operates:



The following fields can be specified on this screen:

- **JMeter Location** – This should point to the location on the host computer where JMeter is installed. You can click on the browse button and navigate to the location of the JMeter.bat file.
- **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.

14.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated JMeter test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

Test Case: Sample JMeter Test Case [TC:000025]

Name*: Sample JMeter Test Case

Description:

Author*: System Administrator **Est. Dur.:** hours

Owner: -- None -- **Creation Date:** 9/2/2011 11:07:54 AM

Priority: -- None -- **Execution Status:** Not Run

Active*: Yes **Last Executed:**

Test Steps Req. Coverage **Automation** Comments Custom Props Test Runs Releases Attachments

This section defines the automated test script associated with this test case:

Automation Engine*: Apache JMeter

Script Type*: Attached Linked

Filename*: [MyDocuments]JMeterJMeter-SampleScript.jmx

Document Type*: Default

Document Folder*: Root Folder

Version: v

Test Script*: [> Edit Parameters](#)

You need to enter the following fields:

- **Automation Engine** - Choose the JMeter Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with JMeter only supports referencing JMeter test script files and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This consists of the following elements:
 - ▷ The full path to the JMeter test script. To make this easier across different machines, you can use several constants for standard Windows locations (see example in screenshot):
 - [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - [CommonDocuments] – The Public Document’s folder.
 - [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
 - ▷ Optionally you can include JMeter command-line arguments by separating them with a pipe (|) character.
 - ▷ Examples of Filenames you can enter in SpiraTeam include:
 - [MyDocuments]JMeter\JMeter-SampleScript.jmx
 - [MyDocuments]JMeter\JMeter-SampleScript.jmx|-P 81
 - [MyDocuments]JMeter\JMeter-SampleScript.jmx|-P 81 -H 192.168.117.25
- **Document Type** – This allows you to choose which document type the automated test script will be categorized under.

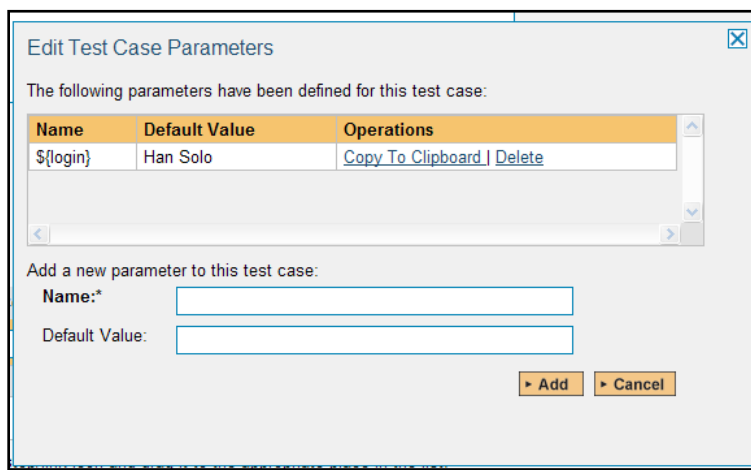
- **Document Folder** –This allows you to choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the JMeter Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

14.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your JMeter automated test script. This is very useful if you have a data-driven JMeter test script that expects specific JMeter properties to be passed to the test script.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter \${login} needs to match the name of the property defined within the JMeter script.

14.3. Executing the JMeter Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

14.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	Automation Host	Test Set #	Edit
TC 7.0 Testing (1)	 	21-Oct-2010	21-Oct-2010		In Progress	InflectraSvr01	TX000010	Edit
QTP Testing (1)	 	22-Oct-2010	22-Oct-2010		Completed	InflectraSvr02	TX000011	Edit
SmartScript Testing (1)	 	26-Oct-2010	26-Oct-2010		Completed	InflectraSvr02	TX000012	Edit
Selenium Testing (3)	 	31-Oct-2010	31-Oct-2010		Completed	InflectraSvr03	TX000013	Edit
Squish Testing (3)	 	2-Nov-2010	2-Nov-2010		Completed	TestHost (VM)	TX000014	Edit
Command Line Testing (1)	 	3-Nov-2010	3-Nov-2010		Completed	InflectraSvr03	TX000017	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the JMeter automated test cases and click on its hyperlink to display the test set details page:

Test Set: JMeter Tests [TX:000013]

Name*: JMeter Tests

Description:

Owner: -- None -- Creator*: System Administrator

Release: -- None -- Type*: Automated

Automation Host: Test Host (VM) Creation Date: 9/2/2011 11:20:06 AM

Status*: Completed Last Executed: -

Planned Date: 9/2/2011 02:00:00 PM Last Updated: 9/2/2011 12:22:28 PM

Test Cases * | Test Runs * | Comments | Custom Props | Attachments | History *

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Dur.: 0.0h / Act. Dur.: 0.0h

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
Sample JMeter Test Case				0.0h	2-Sep-2011	Failed	TC000025	Edit

Show 15 rows per page

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)

- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	Edit Delete
\$(login)	librarian	-	Edit Delete
\$(password)	librarian	-	Edit Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	C-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

browserName: _____

login:

operatingSystem: _____

password:

url: _____

[Save](#) [Cancel](#)

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

The screenshot shows a web interface for a Test Configuration Set [TG:4]. It has tabs for 'Overview' and 'Test Sets *'. Below the tabs are sections for 'Properties', 'Dates and Times', and 'Detailed Information'. Under 'Test Configuration Entries', there are buttons for '+ Populate', 'x Remove', and 'Refresh'. A message states 'Displaying 10 out of 10 entries in this test configuration set.' Below this is a table with columns for a checkbox, a login name, a browser name, and an ID.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

14.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be executed. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the JMeter test:

- **Passed** – The JMeter automated test ran successfully and no failures or errors were logged.
- **Failed** – The JMeter automated test ran successfully, but at least one error or failure was logged.
- **Blocked** – The JMeter automated test did not run successfully.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see a Windows command prompt open as JMeter executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by JMeter, you will see the following information:

Test Run: Sample JMeter Test Case [TR:000078]

Release #: <input type="text" value="-- None --"/>	Estimated Duration: <input type="text" value=""/> hours
Tester Name*: <input type="text" value="System Administrator"/>	Actual Duration*: <input type="text" value="0.0"/> hours
Test Set: JMeter Tests	Execution Date: 9/2/2011 12:36:01 PM
Test Case #*: TC000025	Execution Status*: Failed
Automation Host: Tardis	Test Run Type*: Automated

Test Run Steps
Automation *
Custom Props
Attachments

Runner Name: Apache JMeter 2.x Au	Assert Count: 0
Message: Ran with 2 failures and 0 errors	Test Name: JMeter-SampleScript

Details:

```

Response Assertion (http://www.inflectra.com/): failure=true, error=false, message='Test failed: text expected to
contain /(?)Purchase Our Products Online/'
Response Assertion (http://www.inflectra.com/SpiraTest/Default.aspx): failure=true, error=false, message='Test failed:
text expected to contain /(?)Purchase Our Products Online/'
Response Assertion (http://www.inflectra.com/Purchase/Default.aspx): failure=false, error=false, message=""
Response Assertion (https://www.inflectra.com/Purchase/Default.aspx): failure=false, error=false, message=""

```

This screen indicates the status of the test run that was reported back from JMeter together with any messages or other information. The Test Name indicates the name of the test inside JMeter and the execution status corresponds the rules described above.

In addition, the detailed test report from JMeter is available in the large text-box below. It will contain messages such as:

```

Response Assertion (http://www.inflectra.com/): failure=true, error=false,
message='Test failed: text expected to contain /(?)Purchase Our Products Online/'
Response Assertion (http://www.inflectra.com/SpiraTest/Default.aspx): failure=true,
error=false, message='Test failed: text expected to contain /(?)Purchase Our Products
Online/'
Response Assertion (http://www.inflectra.com/Purchase/Default.aspx): failure=false,
error=false, message=''
Response Assertion (https://www.inflectra.com/Purchase/Default.aspx): failure=false,
error=false, message=''

```

Congratulations... You are now able to run JMeter automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

15. Ranorex

Ranorex is an automated functional test automation system that lets you record application operations and generate .NET language (C#, VB.NET) test automation scripts that can be used to playback the test script against the test application.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of Ranorex on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated Ranorex tests.

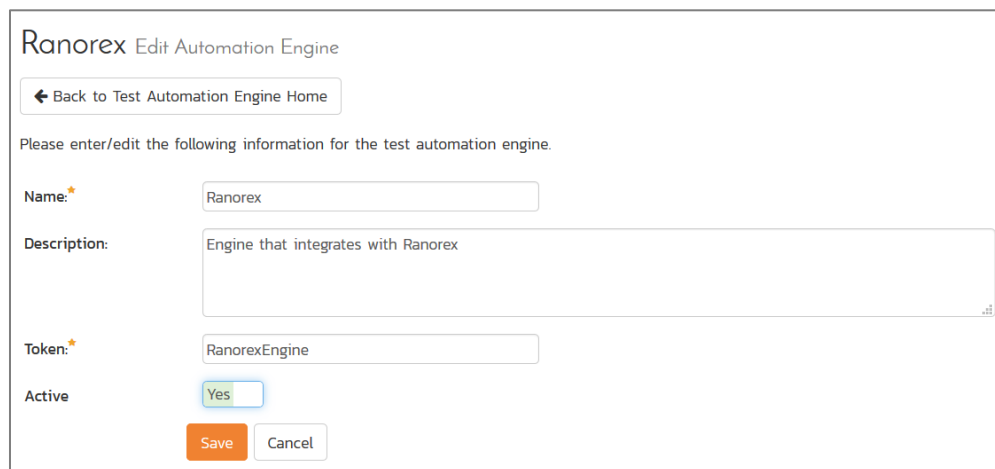
This plugin was developed by one of our partners (step2IT GmbH) but has been formally tested by Inflectra and is fully supported by Inflectra.

Note: This integration requires at least version 3.0 of SpiraTest/Team and version 3.0 of Ranorex.

15.1. Installing the Ranorex Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [RanorexAutomationEngine.zip](#) file from the Inflectra website and locate the appropriate RanorexAutomationEngine.dll for the version of Ranorex that you are using.
 - If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "RanorexAutomationEngine.dll" into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:



Ranorex Edit Automation Engine

[← Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine.

Name:

Description:

Token:

Active: Yes

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.

- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For Ranorex this should always be **RanorexEngine**.
- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Ranorex listed as an available automation engine:

Test Automation Engines

SpiraTeam is able to integrate with a variety of external test automation systems using its flexible, open architecture and library of available test automation engines.

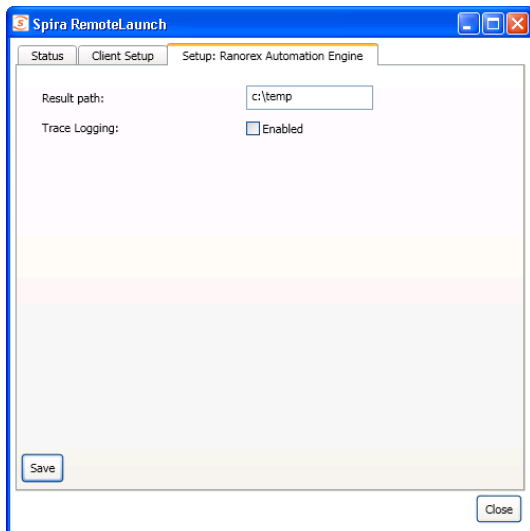
This page allows you to view, add and modify the list of test automation engines and make changes to their configuration:

Name	Token	Description	Active	Operations
Bad Boy	BadBoy2	Engine that integrates with Bad Boy	No	Edit Delete
Command-Line	CommandLine	Engine that will execute a generic command-line executable	Yes	Edit Delete
FitNesse	FitNesse	Engine that integrates with FitNesse framework	No	Edit Delete
IBM RFT	RTFAutomationEngine	Engine that integrates with IBM Rational Functional Tester	No	Edit Delete
JMeter	JMeter2	Engine that integrates with Apache JMeter	No	Edit Delete
LoadRunner	LoadRunner11	Engine that integrates with HP LoadRunner 11.0+	No	Edit Delete
NeoLoad	NeoLoad	Engine that integrates with NeoTys NeoLoad	Yes	Edit Delete
Quick Test Pro	QuickTestPro9	Engine that integrates with HP Quick Test Pro 9.0+	Yes	Edit Delete
Ranorex	RanorexEngine	Engine that integrates with Ranorex	Yes	Edit Delete

15.1.1. Advanced Settings

You can modify the Ranorex configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The Ranorex engine adds its own tab to this page which allows you to configure how Ranorex operates:



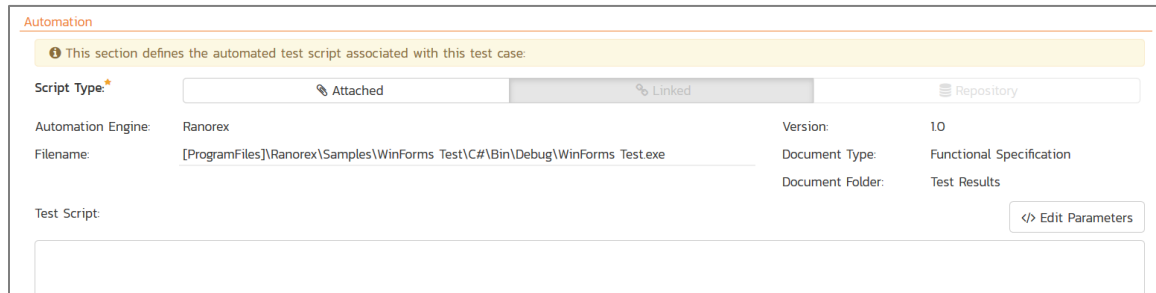
The following fields can be specified on this screen:

- **Result Path** – This is the folder where the results of Ranorex tests will be stored. The currently logged-in user needs to have Read/Write permissions over this folder.
- **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.

15.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated Ranorex test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:



You need to enter the following fields:

- **Automation Engine** - Choose the Ranorex Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with Ranorex only supports referencing Ranorex test script files and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This needs to be the full path to the Ranorex test suite.
 - ▷ To make this easier across different machines, you can use several constants for standard Windows locations (see example in screenshot):
 - [MyDocuments] – The user’s “My Documents” folder. The user indicated is the user that ran RemoteLaunch.
 - [CommonDocuments] – The Public Document’s folder.
 - [DesktopDirectory] – The user’s Desktop folder. The user indicated is the user that ran RemoteLaunch.
 - [ProgramFiles] – Translated to the Program Files directory. For 64-bit machines, it’s the 64-bit directory.
 - [ProgramFilesX86] – Translated to the 32-bit Program Files directory.
 - ▷ If you’d like to pass parameters to Ranorex you can specify them by separating them from the filename with a pipe (“|”) character. For example to run a specific Ranorex test case you can use the following “filename”:
 - `c:\test\mytestsuit.exe|/testcase:addDataTest`
- **Document Type** – This allows you to choose which document type the automated test script will be categorized under.
- **Document Folder** – This allows you to choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the Ranorex Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

15.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your Ranorex automated test suite. This is very useful if you have a data-driven Ranorex test suite that defines input variables from an external data source.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:

Name	Default Value	Operations
\${login}	guest	Insert at Cursor, Edit, Delete

Add a new parameter to this test case:

Name:

Default Value:

The name of the parameter `${login}` needs to match the name of the variable defined within the Ranorex script in its variables configuration.

15.3. Executing the Ranorex Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

15.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Name	Execution Status	Planned Date	Release	Last Executed	Owner	Status	ID	Edit
Regression Testing for Windows 8	Completed				Fred Bloggs	Completed	TX3	Edit
Regression Testing for Windows Vista	Completed					Completed	TX4	Edit
Sample Ranorex Tests	Not Started	22-Nov-2017	1.0.0.0			Not Started	TX8	Edit

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Ranorex automated test cases and click on its hyperlink to display the test set details page:

Sample Ranorex Tests

Test Set [TX:8] Status: * Not Started

Execution Status:

Overview | Test Runs | Attachments | Incidents | History *

People	Properties	Dates and Times
Owner: -- None --	Release: 1.0.0.0 - Library System...	Creation Date: 11/22/2017 5:34:58 PM
Creator: ● System Administrator	Type: * Manual	Last Executed: -
	Automation Host: -- None --	Last Updated: 11/22/2017 5:36:42 PM
	Configuration: -- None --	Planned Date: 11/22/2017 12:00 AM
	Operating System: --- Please Select ---	Recurrence: -- One Time --
		Schedule on Build: <input type="button" value="No"/>
		Post-Build Delay (s):

Lower down, the list of test cases in the test set are displayed:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	ID	Edit
Book_management		2 - High	0.1h			Regression	Not Run	TC8	Edit
Author_management		2 - High	0.1h			Regression	Not Run	TC9	Edit

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you can set their values in three different ways:

- **Test Set Parameter Values** – this lets you set the same value of a parameter for all the test cases in the test set:

Parameter	Test Set Value	Default Value	Operations
\$(browserName)	Safari	browser	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
\$(login)	librarian	-	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete
\$(password)	librarian	-	<input type="checkbox"/> Edit <input checked="" type="checkbox"/> Delete

- **Test Case Parameter Values** – this lets you set a specific value for a parameter for a particular test case in the test set:

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
Ability to create new book		1 - Critical	0.2h	0.0h	8-Nov-2017	Functional	Passed		TC-2	Edit
Ability to edit existing author		2 - High	0.1h			Functional	Not Run	login: admin password: admin	TC-5	Edit
Ability to edit existing book		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Passed		TC-3	Edit
Ability to create new author		1 - Critical	0.1h	15h	1-Dec-2003	Functional	Failed		TC-4	Edit
Ability to reassign book to different author		2 - High	0.1h			Functional	Not Run		TC-6	Edit
Book management		2 - High	0.1h			Regression	Not Run		TC-8	Edit
Author management		2 - High	0.1h			Regression	Not Run		TC-9	Edit

You set these values, by right-clicking on a row and choosing “Edit Parameters”:

Dialog box titled "Edit Test Case Parameters" with a close button (X). A yellow message box says "Please fill out the parameters for this test case entry:". Below are input fields for browserName, login (admin), operatingSystem, password (admin), and url. At the bottom are "Save" and "Cancel" buttons.

- **Test Configurations** – this lets you create a data grid of possible test parameters and execute the test set multiple times, once for each unique combination:

Testing all of the browsers and logins

Test Configuration Set [TG:4]

Overview Test Sets #

Properties + Dates and Times +

Detailed Information +

Test Configuration Entries

+ Populate x Remove Refresh

Displaying 10 out of 10 entries in this test configuration set.

<input type="checkbox"/>	\$(login)	\$(browserName)	ID
<input type="checkbox"/>	admin	Internet Explorer	22
<input type="checkbox"/>	admin	Mozilla / Firefox	23
<input type="checkbox"/>	admin	Safari	25
<input type="checkbox"/>	borrower	Internet Explorer	26
<input type="checkbox"/>	borrower	Mozilla / Firefox	27
<input type="checkbox"/>	borrower	Opera	28

15.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the Ranorex test:

- **Passed** – The Ranorex automated test ran successfully and all the test steps in the test script passed and no assertions were thrown.
- **Failed** – The Ranorex automated test ran successfully, but at least one test step failed or at least one assertion failed.
- **Caution** – The Ranorex automated test run successfully, but at least one warning was logged in one of the test steps.
- **Blocked** – The Ranorex automated test did not run successfully or at least one timeout error was recorded.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see browser windows launch as Ranorex executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by Ranorex, you will see the following information:

This screen indicates the status of the test run that was reported back from Ranorex together with any messages or other information. The Test Name indicates the name of the test inside Ranorex and the execution status corresponds the matching status inside Ranorex as illustrated below:

Ranorex Status	SpiraTeam Status
Success	Passed
Failed	Failed

In addition, the detailed test report from Ranorex is available in the large text-box below. It will contain messages such as:

```
[2011/10/14 14:08:32.795][Debug ][Logger]: Console logger starting.
[2011/10/14 14:08:32.874][Info  ][Test]: Test Suite 'WinForms Test' started.
[2011/10/14 14:08:32.889][Info  ][Test]: Test Case 'VS2005_Application_Test' started.
[2011/10/14 14:08:33.467][Success][Test]: Test Module 'StartWinformsSample' completed
with status 'Success'.
[2011/10/14 14:08:33.467][Info  ][Test]: Test Module 'CheckIfApplicationAlive'
started.
[2011/10/14 14:08:33.608][Info  ][Validation]: Validating Exists on item
'WinFormsApp.ButtonTest_PushButton'.
[2011/10/14 14:09:55.718][Failure][Test]: Test Case 'VS2005_Application_Test'
completed with status 'Failed'.
[2011/10/14 14:09:55.718][Failure][Test]: Test Suite 'WinForms Test' completed with
status 'Failed'.
```

Congratulations... You are now able to run Ranorex automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

16. Rational Functional Tester

IBM Rational Functional Tester (hereafter RFT) is software test automation tool used by quality assurance teams to perform automated regression testing. Testers create scripts by using a test recorder which captures a user's actions against their application under test. The recording mechanism creates a test script from the actions. The test script is produced as either a Java or Visual Basic.net application.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of RFT on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated RFT tests.

Note: This integration requires at least version 3.0 of SpiraTest/Team.

16.1. Installing the RFT Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [RFTEngine.zip](#) file from the Inflectra website and locate the appropriate RFTAutomationEngine.dll for the version of RFT that you are using.
 - If you don't see the version listed, just use the nearest version that is *lower* than your current version.
- Copy the file "RFTAutomationEngine.dll" into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Engine | Rational Functional Tester

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in **bold**:

Name*:

Description:

Token*:

Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.

- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For RFT this should always be **RFTAutomationEngine**.
- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with RFT listed as an available automation engine.

Test Automation Engines

SpiraTeam is able to integrate with a variety of external test automation systems using its flexible, open architecture and library of available test automation engines.

This page allows you to view, add and modify the list of test automation engines and make changes to their configuration:

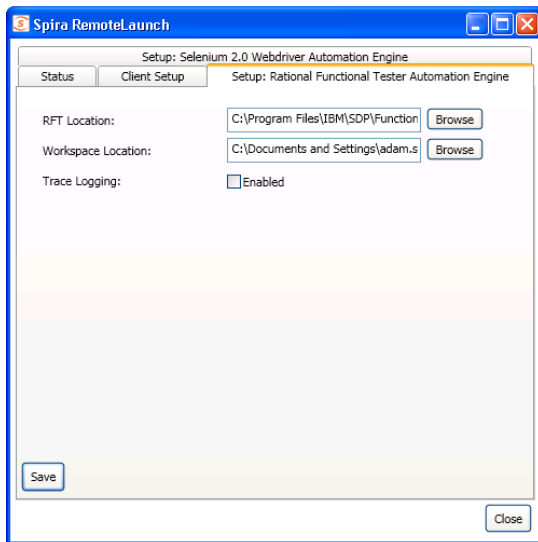
Engine Name	Token	Description	Active	Operations
Quick Test Pro	QTP	Engine that integrates with HP/Mercury Quick Test Pro	Yes	> Edit Delete
Ranorex 3	RanorexEngine	This plugin allows SpiraTest to manage and execute automated Ranorex tests.	Yes	> Edit Delete
Rational Functional Tester	RFTAutomationEngine		Yes	> Edit Delete
Selenium	Selenium2	Engine that integrates with the open-source Selenium RemoteControl (RC)	Yes	> Edit Delete
SmarteScript	SeS	Engine that integrates with SmarteSoft SmarteScript	Yes	> Edit Delete
TestComplete	TestComplete	Engine that integrates with AutomatedQA TestComplete	Yes	> Edit Delete

➤ [Add](#)

16.1.1. Advanced Settings

You can modify the RFT configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The RFT engine adds its own tab to this page which allows you to configure how RFT operates:



The following fields can be specified on this screen:

- **RFT Location** – this is where the installation of RFT can be found. Typically it's C:\Program Files\IBM\SDP\FunctionalTester\bin
- **Workspace Location** – This is the folder where the RFT test scripts and generated log files will be stored. The currently logged-in user needs to have Read/Write permissions over this folder. Typically it's:

- ▷ C:\Documents and Settings\[User Name]\IBM\rationalsdp\workspace on a Windows XP workstation or Windows 2003 server.
 - ▷ C:\Users\[User Name]\IBM\rationalsdp\workspace on a Windows Vista, 7, 2008 or 2008 R2 computer.
- **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.

16.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated RFT test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” tab:

The screenshot shows the 'Automation' tab in SpiraTeam. The form contains the following fields and values:

- Automation Engine*:** Rational Functional Tester
- Script Type*:** Attached Linked
- Filename*:** Test Project|Script|java
- Document Type*:** Default
- Document Folder*:** Root Folder
- Version:** v 1.0
- Test Script*:** (Empty field with a '> Edit Parameters' link)

You need to enter the following fields:

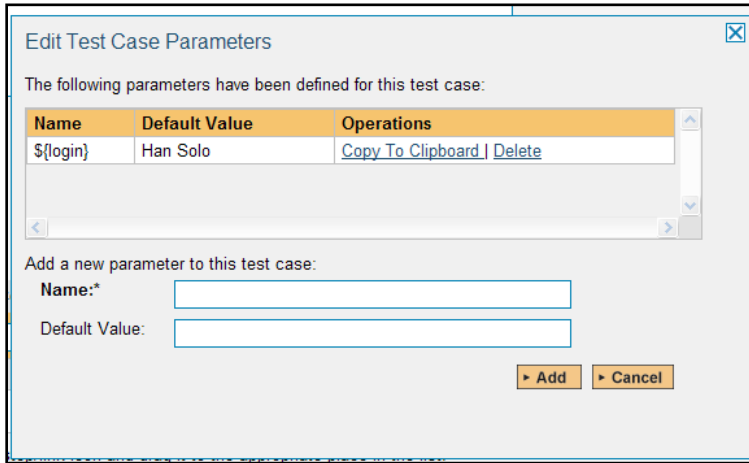
- **Automation Engine** - Choose the RFT Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with RFT only supports referencing RFT test script files and not physically uploading the test scripts into SpiraTeam.
- **Filename** – This needs to consist of the following three components separated by a pipe (|) character (see the screenshot for an example):
 - ▷ The name of the RFT project that the test is mapped to
 - ▷ The name of the RFT script in the project that the test is mapped to
 - ▷ Either “java” or “net” depending on whether you have a Java or .NET test script
- **Document Type** – This allows you to choose which document type the automated test script will be categorized under.
- **Document Folder** – This allows you to choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the RFT Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

16.2.1. Using Parameterized Test Cases

There is an advanced feature of SpiraTest/Team and RemoteLaunch that lets you pass parameters from SpiraTeam to your RFT automated test suite. This is very useful if you have a data-driven RFT test suite that defines input variables from an external data source.

To setup the automated test case for parameters, click on the “Test Steps” tab and click on “Edit Parameters”:



The name of the parameter `${login}` is actually not used when passing the data to RFT, only the values are passed. Therefore it's important that the parameters are stored in the order they are expected by your RFT test script.

16.3. Executing the RFT Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

16.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
		-- Any --		AH	-- Any --	-- Any --	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

✓	Test Set Name	Execution Status	Planned Date	Last Executed	Owner	Status	ID	Edit
<input type="checkbox"/>	Ranorex_Set (1)	Completed	14-Oct-2011	14-Oct-2011		Completed	TX000010	Edit
<input type="checkbox"/>	Selenium_Set (1)	Completed	18-Oct-2011	18-Oct-2011		Completed	TX000011	Edit
<input type="checkbox"/>	RFT Test Set (1)	Not Started	3-Nov-2011	7-Nov-2011		Completed	TX000012	Edit

Show 15 rows per page | Displaying page 1 of 1

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the RFT automated test cases and click on its hyperlink to display the test set details page:

Test Set: RFT Test Set [TX:000012]

Name*: RFT Test Set

Description:

Owner: -- None --

Release: -- None --

Automation Host: Ricky WinXP Test Host

Status*: Not Started

Planned Date: 11/3/2011 05:15:00 PM

Creator*: System Administrator

Type*: Automated

Creation Date: 11/3/2011 4:14:32 PM

Last Executed: -

Last Updated: 11/3/2011 4:15:18 PM

Test Cases * | Test Runs | Comments | Custom Props | Attachments | History #

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests | Est. Dur.: 0.0h / Act. Dur.: 0.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
<input type="checkbox"/>	RFT Test Case						Not Run	TC000020	Edit

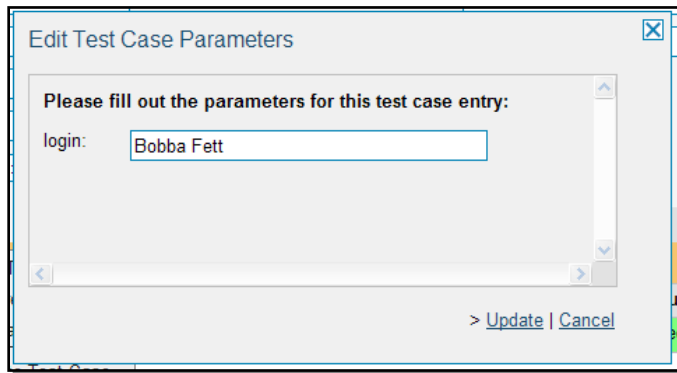
Show 15 rows per page | Displaying page 1 of 1

To change the order of the test cases, please click on the test case icon and drag it to the appropriate place in the list.

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

If you have parameterized test cases inside the automated test set you need to set their values by right-clicking on the test case and choosing “Edit Parameters”:



Enter the parameter values and click “Update” to commit the change. This allows you to have the same test case in the test set multiple times with different data for each instance.

16.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the RFT test:

- **Passed** – The RFT automated test ran successfully and all the test steps in the test script passed and no assertions were thrown.
- **Failed** – The RFT automated test ran successfully, but at least one test step failed or at least one assertion failed.
- **Caution** – The RFT automated test run successfully, but at least one warning was logged in one of the test steps.
- **Blocked** – The RFT automated test did not run successfully.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see browser windows launch as RFT executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by RFT, you will see the following information:

Test Run: RFT Test Case [TR:000084]

Release #: -- None --

Tester Name: System Administrator

Test Set: RFT Test Set

Test Case #: TC000020

Automation Host: Ricky WinXP Test Host

Estimated Duration: 0.0 hours

Actual Duration: 0.0 hours

Execution Date: 11/7/2011 3:01:33 PM

Execution Status: **Failed**

Test Run Type: Automated

Test Run Steps: Automation * Custom Props Attachments

Runner Name: Rational Functional

Assert Count: 2

Message: Executed 6 events with 2 failures and 1 warnings

Test Name: Test Project / Script1

Details:

```

07-Nov-2011 03:00:05.004 PM: Script Start - INFORMATION - Script start [Script1]
07-Nov-2011 03:00:05.035 PM: Simplified Script Group - INFORMATION - firefox.exe: self improvement - QuickStart
Tutorials for Rational Functional Tester (RFT) - Stack Overflow - Mozilla Firefox
07-Nov-2011 03:00:05.035 PM: Timer Start - INFORMATION - Start timer: firefoxexeselfimprovementQuickSta_1
07-Nov-2011 03:00:25.535 PM: General - WARNING - Object Recognition is weak (above the warning threshold)
07-Nov-2011 03:00:49.488 PM: General - FAIL - Script1.testMain had an unhandled exception.
07-Nov-2011 03:00:49.488 PM: Script End - FAIL - Script end [Script1]
Exception occurred during playback of script [Script1] [CRFCN0019E: RationalTestScriptException on line 49 of script
Script1 - com.rational.test.ft.ObjectNotFoundException: CRFCN0661W: The recognition score of the found object does
not qualify the object as a match.
Looking for [GuiSubitemTestObject(Name: goToAWebSitetext, Map: GoToAWebSite)], best failing candidate score was
[22500] with best failing description [{.class=.Text, .name=Go to a Web Site, .classIndex=0}].

```

This screen indicates the status of the test run that was reported back from RFT together with any messages or other information. The Test Name indicates the name of the test inside RFT and the execution status corresponds the matching status inside RFT as illustrated below:

RFT Status	SpiraTeam Status
PASS	Passed
FAIL	Failed
WARNING	Caution

In addition, the detailed test report from RFT is available in the large text-box below. It will contain messages such as:

```

07-Nov-2011 03:00:05.004 PM: Script Start - INFORMATION - Script start [Script1]
07-Nov-2011 03:00:05.035 PM: Simplified Script Group - INFORMATION - firefox.exe: self
improvement - QuickStart Tutorials for Rational Functional Tester (RFT) - Stack
Overflow - Mozilla Firefox
07-Nov-2011 03:00:05.035 PM: Timer Start - INFORMATION - Start timer:
firefoxexeselfimprovementQuickSta_1
07-Nov-2011 03:00:25.535 PM: General - WARNING - Object Recognition is weak (above the
warning threshold)
07-Nov-2011 03:00:49.488 PM: General - FAIL - Script1.testMain had an unhandled
exception.
07-Nov-2011 03:00:49.488 PM: Script End - FAIL - Script end [Script1]
Exception occurred during playback of script [Script1] [CRFCN0019E:
RationalTestScriptException on line 49 of script Script1 -
com.rational.test.ft.ObjectNotFoundException: CRFCN0661W: The recognition score of the
found object does not qualify the object as a match.
Looking for [GuiSubitemTestObject(Name: goToAWebSitetext, Map: GoToAWebSite)], best
failing candidate score was [22500] with best failing description [{.class=.Text,
.name=Go to a Web Site, .classIndex=0}].

```

Congratulations... You are now able to run RFT automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

17. TestingAnywhere

TestingAnywhere is a powerful and easy to use automated software testing tool that allows users to automate any type of testing. Powerful GUI based recording capabilities and a no-programming required user interface allows testers to quickly set up even complex test cases. A built-in editor allows users to build tests that can be easily edited to allow for changes in the test cases.

This section describes how you can use SpiraTest / SpiraTeam (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of TestingAnywhere on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated TestingAnywhere tests.

Note: This integration requires at least version 4.0 of SpiraTest/Team and RemoteLaunch.

17.1. Installing the TestingAnywhere Engine

This section assumes that you already have a working installation of SpiraTest or SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [TestingAnywhereEngine.zip](#) file from the Inflectra website and locate the TestingAnywhereAutomationEngine.dll file contained within.
- Copy the file "TestingAnywhereAutomationEngine.dll" into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

Edit Automation Engine | TestingAnywhere

[<< Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine. Required fields are indicated in bold:

Name:*

Description:

Token:*

Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For TestingAnywhere this should simply be **TestingAnywhere**.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with TestingAnywhere listed as an available automation engine.

Test Automation Engines

SpiraTeam is able to integrate with a variety of external test automation systems using its flexible, open architecture and library of available test automation engines.

This page allows you to view, add and modify the list of test automation engines and make changes to their configuration:

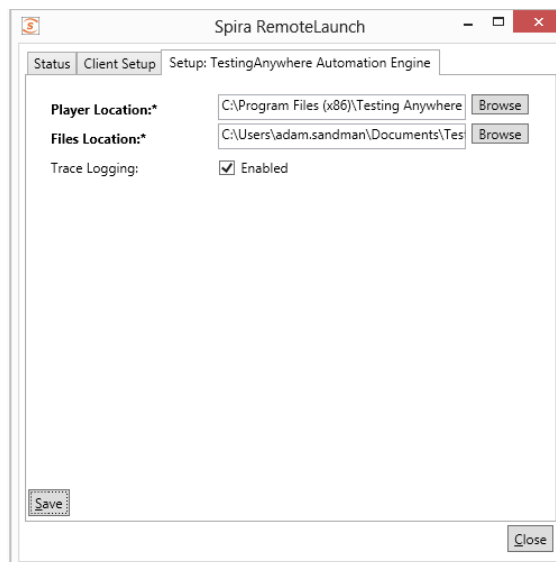
Name	Token	Description	Active	Operations
Bad Boy	BadBoy2	Engine that integrates with Bad Boy	No	> Edit Delete
Command-Line	CommandLine	Engine that will execute a generic command-line executable	No	> Edit Delete
FitNesse	FitNesse	Engine that integrates with FitNesse framework	No	> Edit Delete
TestingAnywhere	TestingAnywhere	Plugin that integrates with TestingAnywhere	Yes	> Edit Delete
TestPartner	TestPartner	Engine that integrates with MicroFocus TestPartner	Yes	> Edit Delete

[Add](#)

17.1.1. Advanced Settings

You can modify the TestingAnywhere configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The TestingAnywhere engine adds its own tab to this page which allows you to configure how TestingAnywhere operates:



The following fields can be specified on this screen:

- **Player Location** – this is the folder where the TestingAnywhere player (TAPlayer.exe) can be found. Typically it's C:\Program Files (x86)\Testing Anywhere 9.0\Testing Anywhere
- **Files Location** – This is the folder where the TestingAnywhere test scripts and generated log files will be stored. The currently logged-in user needs to have Read/Write permissions over this folder. Typically it's:
 - ▷ C:\Documents And Settings\[UserName]\My Documents\Testing Anywhere Files on a Windows XP workstation or Windows 2003 server.

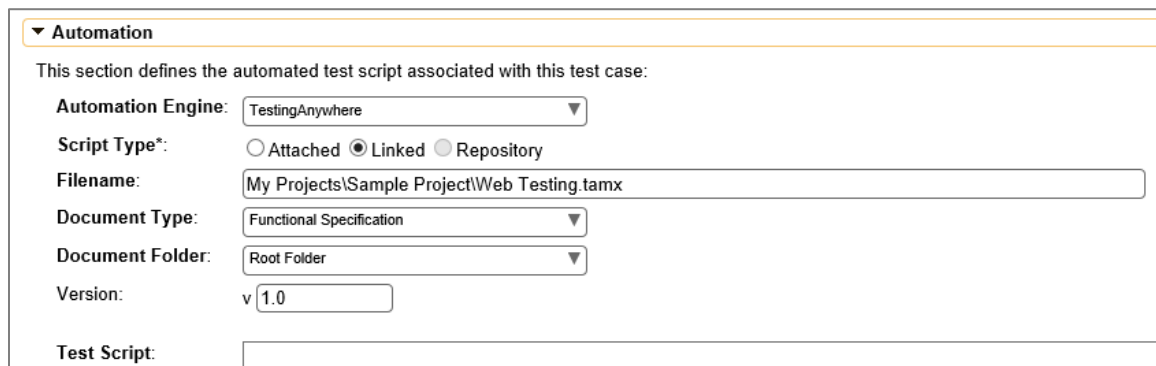
▷ C:\Users\[UserName]\Documents\Testing Anywhere Files on a Windows Vista, 7, 2008 or 2008 R2 computer.

- ▶ **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.

17.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated TestingAnywhere test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Overview” tab, and scroll down to the “Automation” section:



You need to enter the following fields:

- ▶ **Automation Engine** - Choose the TestingAnywhere Automation Engine that you created in the previous section from the drop-down list.
- ▶ **Script Type** – This should be set to Linked as the integration with TestingAnywhere only supports referencing TestingAnywhere test script files and not physically uploading the test scripts into SpiraTeam.
- ▶ **Filename** – This needs to consist of the relative location of the TestingAnywhere test script to the test script root folder.
- ▶ **Document Type** – This allows you to choose which document type the automated test script will be categorized under.
- ▶ **Document Folder** – This allows you to choose which document folder the automated test script will be stored in.
- ▶ **Version** – The version of the test script (1.0 is used if no value specified)
- ▶ **Test Script** – *This is not used with the TestingAnywhere Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

17.3. Executing the TestingAnywhere Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future

2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

17.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 7	Edit
TARDIS	TARDIS	Yes	14-Nov-2017	AH5			Edit

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Name	Execution Status	Planned Date	Last Executed	Owner	Status	ID
Functional Test Sets		4-Feb-2007			In Progress	TX000008
JA Test Set (1)		2-Apr-2014	2-Apr-2014		Completed	TX000010
Testing Cycle for Release 1.0 (7)		4-Feb-2007	1-Dec-2003	Joe P Smith	In Progress	TX000001
Testing Cycle for Release 1.1 (9)		6-Feb-2007	1-Dec-2003	Joe P Smith	Not Started	TX000002
Testing New Functionality (4)		9-Feb-2007	27-Mar-2014	Fred Bloggs	In Progress	TX000005
Exploratory Testing (2)				Fred Bloggs	Deferred	TX000006

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the TestingAnywhere automated test cases and click on its hyperlink to display the test set details page:

Test Set: TA Test Set [TX.000010]

Name: TA Test Set

Overview | Test Runs | Attachments | Incidents | History

Details

Owner: -- None -- Creator*: System Administrator

Release: 1.0.0.0 - Library System Release 1 Type: Automated

Automation Host: Windows 7 Host Creation Date: 4/2/2014 1:03:27 PM

Status*: Not Started Last Executed: 4/2/2014 1:21:09 PM

Planned Date: 04/02/2014 2:00 pm -- One Time -- Last Updated: 4/2/2014 1:33:11 PM

Notes: [Rich text editor]

Operating System: -- Please Select --

Description [Text area]

Comments [Text area]

Test Cases

> Add Tests | Remove Tests | Refresh | Edit Parameters | Execute Tests Est. Dur.: 0.00 / Act. Dur.: 0.00

Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
Web Application Test				0.0h	2-Apr-2014	Blocked	TC001029	Edit

Show 15 rows per page

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

17.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be executed. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the TestingAnywhere test:

- **Passed** – The TestingAnywhere automated test ran successfully and all the test steps in the test script passed and no assertions were thrown.
- **Failed** – The TestingAnywhere automated test ran successfully, but at least one test step failed or at least one assertion failed.
- **Caution** – The TestingAnywhere automated test run successfully, but at least one warning was logged in one of the test steps.
- **Blocked** – The TestingAnywhere automated test did not run successfully.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see browser windows launch as TestingAnywhere executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by TestingAnywhere, you will see the following high-level test information:

Test Run: Web Application Test [TR:000036]

Overview # Attachments Incidents

Details

Release #: -- None -- Estimated Duration: hours

Tester Name:* System Administrator Actual Duration: 0.00 hours

Test Set: TA Test Set Execution Date: 4/2/2014 1:33:30 PM

Test Case #:* TC001029 Execution Status:* **Failed**

Build: Test Run Type:* Automated

Web Browser: -- Please Select -- Operating System: -- Please Select --

Notes:
 -- Font -- -- Size -- **B** **I** **U** |

This screen indicates the status of the test run that was reported back from TestingAnywhere together with the execution date/time.

If you scroll down to the 'Console Output' section, you will see:

Console Output

Runner Name: TestingAnywhere Auto Assert Count: 0

Automation Host: Windows 7 Host Test Name: Web Testing

Message: Total: 4, Passed: 2, Failed: 2

Details: Executing: 'C:\Program Files (x86)\Testing Anywhere 9.0\Testing Anywhere\TAPlayer.exe "/FC:\Users\stadam.sandman\Documents\Testing Anywhere Fi
 Project\Web Testing tamx/e"
 1: Comment = Passed
 2: Comment = Passed
 3: OpenBrowser = Passed
 4: Comment = Passed
 5: ExtractData = Passed
 6: variable = Passed
 7: variable = Failed
 8: ManageWebCtrl = Passed
 9: Delay = Passed

Finally, to see the detailed test steps, look under the 'Test Steps' section:

ID	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
RS000038	Comment		Comment: === Please make sure your default browser is Internet Explorer before running this test ===	/		Passed
RS000039	Comment		Comment: === Go to Tethys Solutions Forums page ===	/		Passed
RS000040	OpenBrowser		Open "http://www.tethyssolutions.com/forum"	/		Passed
RS000041	Comment		Comment: === Extract the link caption and the URL in \$Forum-Link\$ and \$Forum-URL\$ variables and click the "Check Point" checkboxes to insert the check points on the values: ===	/		Passed
RS000042	ExtractData		Extract data from Object type: Hyperlink, Webpage: http://www.tethyssolutions.com/forum/ to Forum-Link	/		Passed
RS000043	variable		CheckPoint: Testing Anywhere Equal To (=) "Testing Anywhere"	/		Passed
RS000044	variable		CheckPoint: http://www.automationanywhere.com/forum/forumdisplay.php?25-Testing-Anywhere Equal To (=) "http://www.tethyssolutions.com/forum/testing-anywhere/"	/	Specified criteria ('http://www.automationanywhere.com/forum/forumdisplay.php?25-testing-anywhere' Equal To 'http://www.tethyssolutions.com/forum/testing-anywhere/') did not match. > View Incidents	Failed

Congratulations... You are now able to run TestingAnywhere automated functional tests and have the results be recorded within SpiraTest / SpiraTeam.

18. Worksoft Certify

Worksoft Certify is a test automation solution for enterprise applications including SAP, Workday, Salesforce.com, Oracle, and Web Apps. Built to test complex business processes that span multiple applications, Certify ensures that your software and business work exactly as planned.

This section describes how you can use SpiraTest, SpiraTeam, or SpiraPlan (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of Worksoft Certify on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated Worksoft Certify tests.

Note: This integration requires at least version 5.0 of SpiraTeam and version 10.0 of Worksoft Certify.

18.1. Installing the Worksoft Certify Engine

This section assumes that you already have a working installation of SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [WorksoftCertifyEngine.zip](#) file from the Inflectra website and locate the appropriate WorksoftCertifyEngine.dll for the version of Worksoft Certify that you are using.
- Copy the file "WorksoftCertifyEngine.dll" into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

The screenshot shows a web form titled "Worksoft Certify Edit Automation Engine". At the top left is a button labeled "← Back to Test Automation Engine Home". Below this is the instruction: "Please enter/edit the following information for the test automation engine." The form contains four main fields: "Name:" with a text input containing "Worksoft Certify"; "Description:" with a larger text area containing "Plugin that integrates with Worksoft Certify"; "Token:" with a text input containing "WorksoftCertify"; and "Active:" with a toggle switch currently set to "Yes". At the bottom of the form are two buttons: "Save" (orange) and "Cancel" (white).

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For Worksoft Certify this should always be **WorksoftCertify**.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with Worksoft Certify listed as an available automation engine:

Test Automation Engines

SpiraPlan is able to integrate with a variety of external test automation systems using its flexible, open architecture and library of available test automation engines.

This page allows you to view, add and modify the list of test automation engines and make changes to their configuration:

Name	Token	Description	Active	Operations
Bad Boy	BadBoy2	Engine that integrates with Bad Boy	No	Edit Delete
Command-Line	CommandLine	Engine that will execute a generic command-line executable	No	Edit Delete
FitNesse	FitNesse	Engine that integrates with FitNesse framework	No	Edit Delete
IBM RFT	RTFAutomationEngine	Engine that integrates with IBM Rational Functional Tester	No	Edit Delete
JMeter	JMeter2	Engine that integrates with Apache JMeter	No	Edit Delete
TestComplete	TestComplete11	Engine that integrates with SmartBear TestComplete 11.0+	No	Edit Delete
TestPartner	TestPartner	Engine that integrates with MicroFocus TestPartner	No	Edit Delete
Worksoft Certify	WorksoftCertify	Plugin that integrates with Worksoft Certify	Yes	Edit Delete

[Add](#)

18.1.1. Advanced Settings

You can modify the Worksoft Certify configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The Worksoft Certify engine adds its own tab to this page which allows you to configure how Worksoft Certify operates:

The following fields can be specified on this screen:

- **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.

- **Certify User** – This should be populated with a valid username that can login to the Worksoft Certify database that you are using
- **Certify Password** – This should be populated with a valid password for the user specified in the previous field that can login to the Worksoft Certify database that you are using
- **Test Timeout** – This tells RemoteLaunch how long to let the Worksoft Certify tests run (in seconds) in the event one of the tests were to not finish properly. Once the test report is generated, RemoteLaunch will stop execution, regardless of this setting.
- **Report Generation Time** – This is how long (in seconds) RemoteLaunch should wait after Worksoft Certify has finished before reading the test report (for sending to SpiraTeam). It ensures that Worksoft Certify has enough time to finish writing the report.

18.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated Worksoft Certify test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” section of the main “Overview” tab:

You need to enter the following fields:

- **Automation Engine** - Choose the **Worksoft Certify** Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to Linked as the integration with Worksoft Certify only supports referencing Worksoft Certify projects and not physically uploading the tests into SpiraTeam.
- **Filename** – This needs to contain the following elements at the very least:
 - ▷ **/Process="xxxxx"** needs to specify the name of the Worksoft Certify process
 - ▷ **/Project="xxxxx"** needs to specify the name of the Worksoft Certify project
 - ▷ You can also add other Worksoft Certify command line options - <http://community.worksoft.com/Knowledge-Base/Worksoft-Products/Worksoft-Certify/certify-command-line-options.html>
- **Document Type** – This allows you to choose which document type the automated test script will be categorized under.
- **Document Folder** – This allows you to choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the Worksoft Certify Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

18.3. Executing the Worksoft Certify Test Sets from SpiraTeam

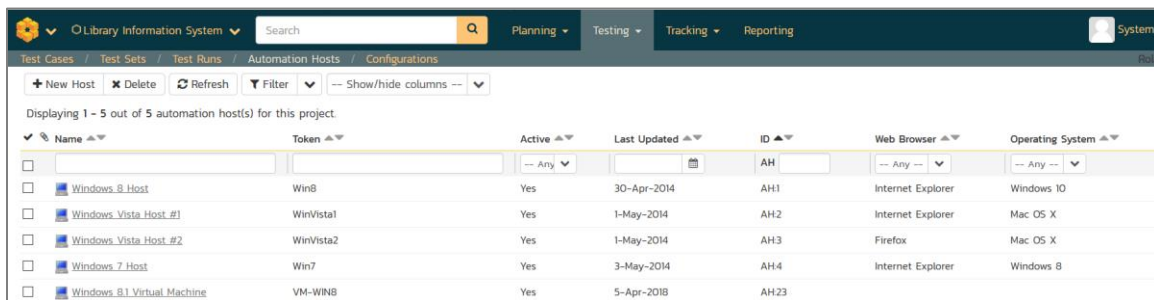
There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

18.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

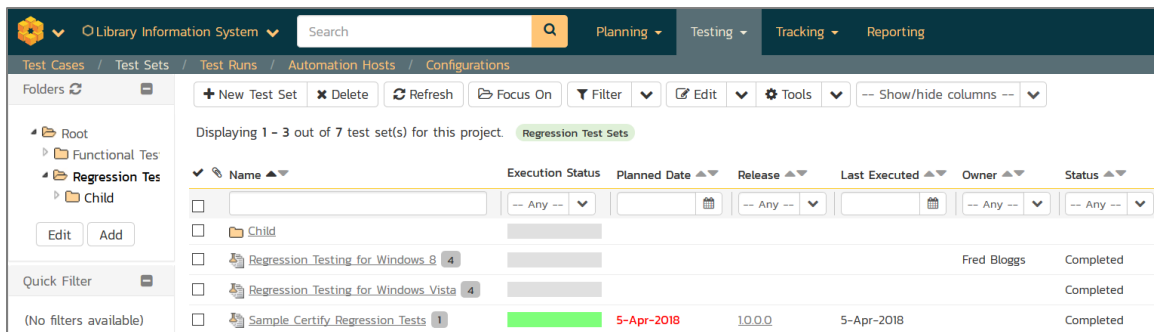


The screenshot shows the SpiraTeam interface for configuring automation hosts. The breadcrumb trail is 'Test Cases / Test Sets / Test Runs / Automation Hosts / Configurations'. The page title is 'Automation Hosts'. There are buttons for '+ New Host', 'Delete', 'Refresh', and a 'Filter' dropdown. Below the buttons, it says 'Displaying 1 - 5 out of 5 automation hosts(s) for this project.' The table below lists the automation hosts with columns for Name, Token, Active, Last Updated, ID, Web Browser, and Operating System.

Name	Token	Active	Last Updated	ID	Web Browser	Operating System
Windows 8 Host	Win8	Yes	30-Apr-2014	AH1	Internet Explorer	Windows 10
Windows Vista Host #1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Mac OS X
Windows Vista Host #2	WinVista2	Yes	1-May-2014	AH3	Firefox	Mac OS X
Windows 7 Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 8
Windows 8.1 Virtual Machine	VM-WIN8	Yes	5-Apr-2018	AH23		

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:



The screenshot shows the SpiraTeam interface for configuring test sets. The breadcrumb trail is 'Test Cases / Test Sets / Test Runs / Automation Hosts / Configurations'. The page title is 'Test Sets'. There are buttons for '+ New Test Set', 'Delete', 'Refresh', 'Focus On', 'Filter', 'Edit', 'Tools', and 'Show/hide columns'. Below the buttons, it says 'Displaying 1 - 3 out of 7 test set(s) for this project.' The table below lists the test sets with columns for Name, Execution Status, Planned Date, Release, Last Executed, Owner, and Status.

Name	Execution Status	Planned Date	Release	Last Executed	Owner	Status
Child						
Regression Testing for Windows 8					Fred Bloggs	Completed
Regression Testing for Windows Vista						Completed
Sample Certify Regression Tests		5-Apr-2018	1.0.0.0	5-Apr-2018		Completed

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the Worksoft Certify automated test cases and click on its hyperlink to display the test set details page:

Sample Certify Regression Tests

Test Set [TX:32] Status: * Completed Execution Status:

Overview Test Runs * Attachments Incidents History *

People	Properties	Dates and Times
Owner: <input type="radio"/> -- None --	Release: 1.0.0.0 - Library System...	Creation Date: 4/5/2018 10:54:29 AM
Creator: * <input type="radio"/> System Administrator	Type: * Automated	Last Executed: 4/5/2018 12:40 PM
	Automation Host: Windows 8.1 Virtual Machine	Last Updated: 4/5/2018 12:41 PM
	Configuration: -- None --	Planned Date: 04/05/2018 10:54 ...
	Operating System: --- Please Select ---	Recurrence: -- One Time --
		Schedule on Build: <input type="button" value="No"/>
		Post-Build Delay (s):

Lower down, the list of test cases in the test set are displayed:

Test Cases

-- Show/Hide columns -- Est. Dur. 0.0h / Act. Dur. 0.0h

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	ID	<input type="button" value="Edit"/>
<input type="checkbox"/>	Google Test			0.0h		5-Apr-2018	Regression	Passed	TC:62	<input type="button" value="Edit"/>

Show 15 rows per page Displaying page 1 of 1

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

18.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the Worksoft Certify test:

- **Passed** – The Worksoft Certify automated test ran successfully and all the test steps in the test script passed and no assertions were thrown.
- **Failed** – The Worksoft Certify automated test ran successfully, but at least one test step failed or at least one assertion failed.

- **Blocked** – The Worksoft Certify automated test did not run successfully and reported back the **aborted** test status to RemoteLaunch.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see browser windows launch as Worksoft Certify executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by Worksoft Certify, you will see the following information:

This screen indicates the status of the test run that was reported back from Worksoft Certify together with any messages or other information. The Test Name indicates the name of the test inside Worksoft Certify and there will be detailed steps displayed that match the Worksoft Certify execution steps:


Step	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
1	Load Browser	Browser	Load "https://www.google.com" into "Internet Explorer".	/	ContinueExecution	Passed
2	Set Window State	System	"Maximize" window with caption "Contains" "Google"	/	ContinueExecution	Passed
3	Wait	Execution	Wait "5" seconds	/	ContinueExecution	Passed
4	Input	txt_SearchBox	Input "Inflectra" into the txt_SearchBox EditBox.	/	ContinueExecution	Passed
5	Send Click	btn_Search	"LeftClick" the btn_Search PushButton.	/	ContinueExecution	N/A
6	Key Press	txt_SearchBox	Press "(ENTER)" time(s) in the txt_SearchBox EditBox.	/	ContinueExecution	Passed
7	Wait	Execution	Wait "3" seconds	/	ContinueExecution	Passed
8	Send Click	link_Result	"LeftClick" the link_Result Link.	/	ContinueExecution	Passed
9	Wait	Execution	Wait "3" seconds	/	ContinueExecution	Passed
10	Close Browser	Browser	Close the most recently used browser window.	/	ContinueExecution	Passed

Each of the SpiraTeam execution status values corresponds the matching status inside Worksoft Certify as illustrated below:

Worksoft Certify Status	SpiraTeam Status
Passed	Passed
Failed	Failed
Aborted	Blocked
Skipped	N/A

In addition, the detailed test report from Worksoft Certify is available in the large text-box below. It will contain messages such as:

Console Output

Runner Name:	Worksoft Certify Auto	Assert Count:	0
Test Name:	01_GoogleSearch - April 5, 2018 13:26:40	Automation Host:	 Windows 8.1 Virtual Machine

Message: 9 passed, 0 failed, 1 skipped, 0 aborted, out of 10 total steps

Details:

```
Load "https://www.google.com" into "Internet Explorer". [passed]
"Maximize" window with caption "Contains" "Google" [passed]
Wait "5" seconds [passed]
Input "Inflectra" into the txt_SearchBox EditBox. [passed]
"LeftClick" the btn_Search PushButton. [skipped]
Press "(ENTER)" time(s) in the txt_SearchBox EditBox. [passed]
Wait "3" seconds [passed]
"LeftClick" the link_Result Link. [passed]
Wait "3" seconds [passed]
Close the most recently used browser window. [passed]
```

Congratulations... You are now able to run Worksoft Certify automated functional tests and have the results be recorded within SpiraTest, SpiraTeam, or SpiraPlan.

19. ZAPTEST

ZAPTEST is a cross-platform and cross-browser software testing solution. Using OCR and Image Based recognition, it's able to automate the testing process without any API, Framework or environment dependencies and work only with a Graphic User Interface

This section describes how you can use SpiraTest, SpiraTeam, or SpiraPlan (hereafter SpiraTeam) together with RemoteLaunch to schedule and remotely launch instances of ZAPTEST on different computers and have the testing results be transmitted back to SpiraTeam. This allows you to extend your SpiraTeam's test management capabilities to include automated ZAPTEST tests.

Note: This integration requires at least version 5.0 of SpiraTeam and version 11.0 of ZAPTEST.

19.1. Installing the ZAPTEST Engine

This section assumes that you already have a working installation of SpiraTeam and have installed RemoteLaunch on the various test automation hosts following the instructions in Section 1 (above). Once those prerequisites are in place, please follow these steps:

- Download and extract the [ZapTestEngine.zip](#) file from the Inflectra website and locate the appropriate [ZapTestEngine.dll](#) for the version of ZAPTEST that you are using.
- Copy the file "ZapTestEngine.dll" into the "extensions" sub-folder of the RemoteLaunch installation.
- Log in to SpiraTeam as a system administrator and go into SpiraTeam main Administration page and click on the "Test Automation" link under **Integration**.
- Click the "Add" button to enter the new test automation engine details page. The fields required are as follows:

ZapTest Edit Automation Engine

[← Back to Test Automation Engine Home](#)

Please enter/edit the following information for the test automation engine.

Name:

Description:

Token:

Active

- **Name:** This is the short display name of the automation engine. It can be anything that is meaningful to your users.
- **Description:** This is the long description of the automation engine. It can be anything that is meaningful to your users. (Optional)
- **Active:** If checked, the engine is active and able to be used for any project.
- **Token:** This needs to be the assigned unique token for the automation engine and is used to tell RemoteLaunch which engine to actually use for a given test case. For ZAPTEST this should always be **ZapTest**.

- Once you have finished, click the “Insert & Close” button and you will be taken back to the Test Automation list page, with ZAPTEST listed as an available automation engine:

SOAP-UI	SoapUI	Engine that integrates with SmartBear SOAP-UI	No	Edit Delete
TestComplete	TestCompleteII	Engine that integrates with SmartBear TestComplete 11.0+	No	Edit Delete
TestPartner	TestPartner	Engine that integrates with MicroFocus TestPartner	No	Edit Delete
ZapTest	ZapTest		Yes	Edit Delete

19.1.1. Advanced Settings

You can modify the ZAPTEST configuration for each of the specific automation hosts, by right-clicking on the RemoteLaunch icon in the system tray and choosing “Configuration”. That will bring up the RemoteLaunch configuration page.

The ZAPTEST engine adds its own tab to this page which allows you to configure how ZAPTEST operates:

The screenshot shows the 'Spira RemoteLaunch' application window. The active tab is 'Setup: ZAPTEST Automation Engine'. The configuration fields are as follows:

- ZAPTEST Location:** C:\Program Files (x86)\ZAP-fix\Standalone (with a Browse button)
- Trace Logging:**
- Report Generation Time:** 2 seconds

Buttons for 'Save' and 'Close' are located at the bottom of the configuration area.

The following fields can be specified on this screen:

- **ZAPTEST Location** – You need to browse to the location of the Standalone ZAPTEST.EXE program (e.g. C:\Program Files (x86)\ZAP-fix\Standalone)
- **Trace Logging** – When selected, this will log additional trace and debugging information to the Windows Event Log. This should not be selected in a production environment.
- **Report Generation Time** – This is how long (in seconds) RemoteLaunch should wait after ZAPTEST has finished before reading the test report (for sending to SpiraTeam). It ensures that ZAPTEST has enough time to finish writing the report.

19.2. Setting up the Automated Test Cases

This section describes the process for setting up a test case in SpiraTeam for automation and linking it to an automated ZAPTEST test script.

First you need to display the list of test cases in SpiraTeam (by clicking Testing > Test Cases) and then add a new test case. Once you have added the new test case, click on it and select the “Automation” section of the main “Overview” tab:



You need to enter the following fields:

- **Automation Engine** - Choose the **ZapTest** Automation Engine that you created in the previous section from the drop-down list.
- **Script Type** – This should be set to **Linked** as the integration with ZAPTEST only supports referencing ZAPTEST script files and not physically uploading the tests into SpiraTeam.
- **Filename** – This needs to contain the full path to a location on the computer running ZAPTEST where the test script can be found.
- **Document Type** – This allows you to choose which document type the automated test script will be categorized under.
- **Document Folder** – This allows you to choose which document folder the automated test script will be stored in.
- **Version** – The version of the test script (1.0 is used if no value specified)
- **Test Script** – *This is not used with the ZapTest Engine since it only supports linked test scripts.*

Once you are happy with the values, click [Save] to update the test case. Now you are ready to schedule the automated test case for execution.

19.3. Executing the ZAPTEST Test Sets from SpiraTeam

There are two ways to execute automated test cases in SpiraTeam:

1. Schedule the test cases to be executed on a specific computer (local or remote) at a date/time in the future
2. Execute the test cases right now on the local computer.

We shall outline both of these two scenarios in this section. However first we need to setup the appropriate automation hosts and test sets in SpiraTeam:

19.3.1. Configuring the Automation Hosts and Test Sets

Go to Testing > Automation Hosts in SpiraTeam to display the list of automation hosts:

Name	Token	Active	Last Updated	ID	Web Browser	Operating System
VM-WIN8	VM-WIN8	Yes	1-May-2018	AH1	Internet Explorer	Windows 10
Windows_Vista_Host_#1	WinVista1	Yes	1-May-2014	AH2	Internet Explorer	Mac OS X
Windows_Vista_Host_#2	WinVista2	Yes	1-May-2014	AH3	Firefox	Mac OS X
Windows_7_Host	Win7	Yes	3-May-2014	AH4	Internet Explorer	Windows 8
TARDIS	TARDIS	Yes	22-Feb-2018	AH6		

Make sure that you have created an Automation Host for each computer that is going to run an automated test case. The name and description can be set to anything meaningful, but the Token field **must be set to the same token that is specified in the RemoteLaunch application** on that specific machine.

Once you have at least one Automation Host configured, go to Testing > Test Sets to create the test sets that will contain the automated test case:

Name	Execution Status	Planned Date	Release	Last Executed	Owner	Status
Child						
Regression_Testing_for_Windows_8					Fred Bloggs	Completed
Regression_Testing_for_Windows_Vista						Completed
Sample_Certify_Regression_Tests		5-Apr-2018	10.0.0	5-Apr-2018		Completed

Note: Unlike manual test cases, automated test cases *must be executed within a test set* – they cannot be executed directly from the test case.

Create a new Test Set to hold the ZAPTEST automated test cases and click on its hyperlink to display the test set details page:

ZapTest Set Status: Completed

People	Properties	Dates and Times
Owner: -- None --	Release: 10.0.0 - Library System...	Creation Date: 5/1/2018 12:27:42 PM
Creator: System Administrator	Type: Automated	Last Executed: 5/1/2018 15:30 PM
	Automation Host: VM-WIN8	Last Updated: 5/1/2018 15:31 PM
	Configuration: -- None --	Planned Date: 05/01/2018 12:00 ...
	Operating System: --- Please Select ---	Recurrence: -- One Time --
		Schedule on Build: <input type="checkbox"/> No
		Post-Build Delay (s):

Lower down, the list of test cases in the test set are displayed:

<input type="checkbox"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Type	Execution Status	Parameters	ID	Edit
<input type="checkbox"/>	Sample ZapTest_Test			0.0h		1-May-2018	Functional	Failed		TC68	Edit

You need to add at least one automated test case to the test set and then configure the following fields:

- **Automation Host** – This needs to be set to the name of the automation host that will be running the automated test set.
- **Planned Date** – The date and time that you want the scenario to begin. (Note that multiple test sets scheduled at the exact same time will be scheduled by Test Set ID order.)
- **Status** – This needs to be set to “Not Started” for RemoteLaunch to pick up the scheduled test set. When you change the Planned Date, the status automatically switches back to “Not Started”
- **Type** – This needs to be set to “Automated” for automated testing

19.3.2. Executing the Test Sets

Once you have set the various test set fields (as described above), the Remote Launch instances will periodically poll SpiraTeam for new test sets. Once they retrieve the new test set, they will add it to their list of test sets to be execute. Once execution begins they will change the status of the test set to “In Progress”, and once test execution is done, the status of the test set will change to either “Completed” – the automation engine could be launched and the test has completed – or “Blocked” – RemoteLaunch was not able to start the automation engine.

If you want to immediately execute the test case on your local computer, instead of setting the “Automation Host”, “Status” and “Planned Date” fields, you can instead click the [Execute] icon on the test set itself. This will cause RemoteLaunch on the local computer to immediately start executing the current test set.

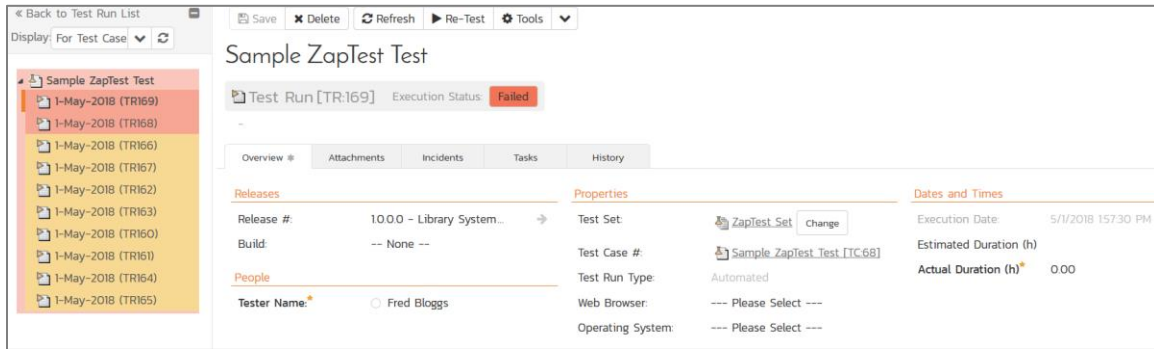
In either case, once all the test cases in the test set have been completed, the status of the test set will switch to “Completed” and the individual test cases in the set will display a status based on the results of the ZAPTEST:

- **Passed** – The ZAPTEST automated test ran completed execution and all the test steps in the test script passed and no failures or warnings were logged.
- **Failed** – The ZAPTEST automated test ran completed execution, but at least one test step failed.
- **Warning** – The ZAPTEST automated test ran completed execution, but at least one test step reported a warning, but no steps completely failed.
- **Blocked** – The ZAPTEST automated test failed to execute because of some configuration error.

If you receive the “Blocked” status for either the test set or the test cases you should open up the Windows Application Event Log on the computer running RemoteLaunch and look in the event log for error messages.

Note: While the tests are executing you will see browser windows launch as ZAPTEST executes the appropriate tests.

Once the tests have completed, you can log back into SpiraTeam and see the execution status of your test cases. If you click on a Test Run that was generated by ZAPTEST, you will see the following information:



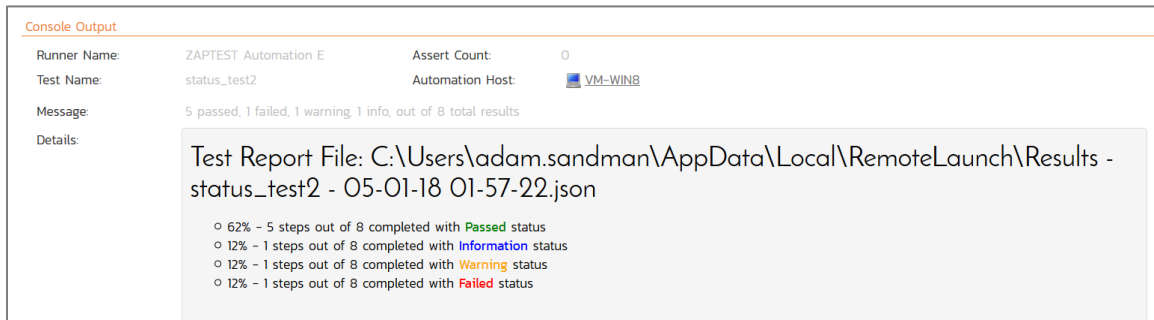
This screen indicates the status of the test run that was reported back from ZAPTEST together with any messages or other information. The Test Name indicates the name of the test script that was executed and there will be detailed steps displayed that match the ZAPTEST execution steps:

Step	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
1	Test Container	status_test2.zap		/		N/A
2	Start Transaction	Status_Test		/		Passed
3	Status Results	Application	Status Results	/	Status is Passed.	Passed
4	Status Results	Application	✘ Status Results	/	Status is Failed.	Failed
5	Status Results	Application	Status Results	/	Status is Warning	Caution
6	Status Results	Application	Status Results	/	Status is Information.	N/A
7	End Transaction	Status_Test		/	6.3 secs	Passed

Each of the SpiraTeam execution status values corresponds the matching status inside Worksoft Certify as illustrated below:

ZAPTEST Status	SpiraTeam Status
Passed	Passed
Failed	Failed
Warning	Caution
Information	N/A

In addition, the detailed test report from ZAPTEST is available in the large text-box below. It will contain messages such as:



Congratulations... You are now able to run ZAPTEST automated functional tests and have the results be recorded within SpiraTest, SpiraTeam, or SpiraPlan.

Legal Notices

This publication is provided as is without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors. Changes are periodically added to the information contained herein; these changes will be incorporated in new editions of the publication. Inflectra Corporation may make improvements and/or changes in the product(s) and/or program(s) and/or service(s) described in this publication at any time.

The sections in this guide that discuss internet web security are provided as suggestions and guidelines. Internet security is constantly evolving field, and our suggestions are no substitute for an up-to-date understanding of the vulnerabilities inherent in deploying internet or web applications, and Inflectra cannot be held liable for any losses due to breaches of security, compromise of data or other cyber-attacks that may result from following our recommendations.

SpiraTest®, SpiraPlan®, SpiraTeam®, RemoteLaunch®, RemoteLaunchX™ and Inflectra® are either trademarks or registered trademarks of Inflectra Corporation in the United States of America and other countries. Microsoft®, Windows®, Explorer® and Microsoft Project® are registered trademarks of Microsoft Corporation. QuickTest Pro® is a registered trademark of Hewlett-Packard Development Company, L.P. All other trademarks and product names are property of their respective holders.

Please send comments and questions to:

Technical Publications

Inflectra Corporation

8121 Georgia Ave, Suite 504

Silver Spring, MD 20910-4957

U.S.A.

support@inflectra.com