



Using Rapise® with SpiraTest®

User Guide

Inflectra Corporation

Date: April 18th, 2015



Contents

Introduction	1
1. Overview	2
2. Configuring SpiraTest	2
3. Connecting Rapise to SpiraTest ...	4
4. Scheduling the Tests	17
5. Configuring RapiseLauncher	23

Introduction

Rapise® is a next generation software test automation tool that leverages the power of open architecture to improve application quality and reduce time to market.

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

This guide outlines how to use Rapise® with SpiraTest® (or SpiraTeam®) to centrally manage your automated tests and remotely schedule and launch them in your test lab.

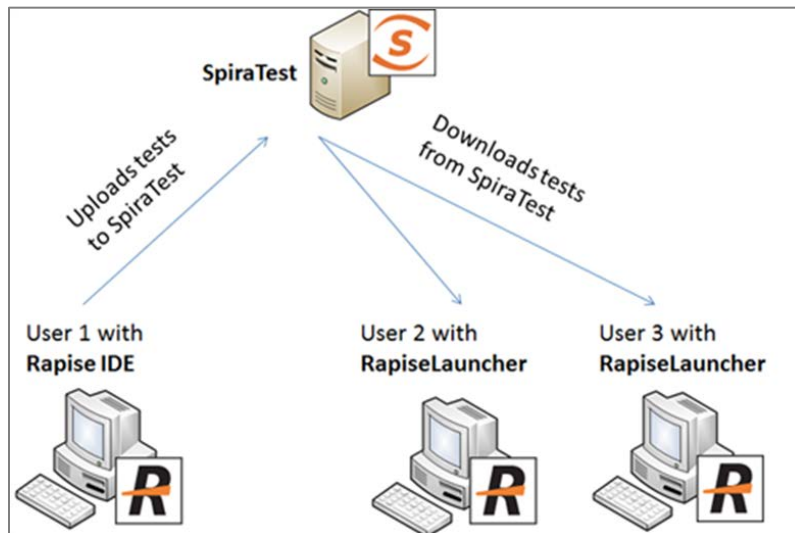
In addition, Rapise includes powerful tools for streamlining the creation, editing and execution of manual test cases that can be stored in SpiraTest, this manual also explains these features.

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

1. Overview

SpiraTest is a web-based quality assurance and test management system with integrated release scheduling and defect tracking. SpiraTest includes the ability to execute manual tests, record the results and log any associated defects.

When you use SpiraTest with Rapise you get the ability to store your Rapise automated tests inside the central SpiraTest repository with full version control and test scheduling capabilities:



You can record and create your test cases using Rapise, upload them to SpiraTest and then schedule the tests to be executed on multiple remote computers to execute the tests immediately or according to a predefined schedule. The results are then reported back to SpiraTest where they are archived as part of the project. Also the test results can be used to update requirements' test coverage and other key metrics in real-time.

In addition, since version 3.0 of Rapise, you can also create exploratory manual tests, save them to SpiraTest and then execute them from within Rapise. This gives you access to more powerful screen capture and annotation functionality than is possible solely using SpiraTest.

Note: SpiraTeam is an integrated ALM Suite that includes SpiraTest as part of its functionality, so wherever you see references to SpiraTest in this section, it applies equally to SpiraTeam.

2. Configuring SpiraTest

Before you can use SpiraTest to manage your Rapise automated tests you need to perform some initial configuration. This section assumes that you already have a working installation of SpiraTest or SpiraTeam v4.2 or later. If not, please refer to the *SpiraTest Installation and Administration Guide* for details on how to install SpiraTest.

2.1. Configuring the Rapise Automation Engine

Log in to SpiraTest as a system administrator and go into SpiraTest 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 | Rapise
[<< 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 SpiraTest which engine to actually use for a given test case. For Rapise this should always be set to “**Rapise**”.

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

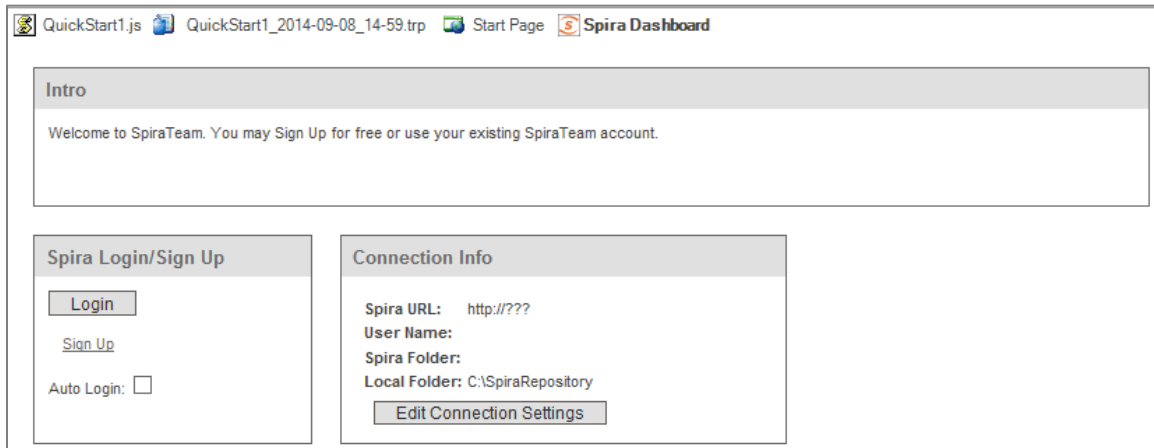
Test Automation Engines				
SpiraTeam is able to integrate with a variety of external test automation systems using its flexible, open architecture and library of available test automation engines. This page allows you to view, add and modify the list of test automation engines and make changes to their configuration:				
Engine Name	Token	Description	Active	Operations
Apache JMeter	JMeter2		Yes	> Edit Delete
Bad Boy	BadBoy2		Yes	> Edit Delete
Command Line	CommandLine		Yes	> Edit Delete
Rapise 1.3	Rapise	Automation engine for Inflectra Rapise	Yes	> Edit Delete
Selenium	Selenium	Engine that integrates with the open-source Selenium RemoteControl (RC)	Yes	> Edit Delete
				<input type="button" value="Add"/>

Depending on when you first installed SpiraTest, you may already have an entry for Rapise, since versions 3.2 and later of SpiraTest automatically ship with a default entry for Rapise.

3. Connecting Rapise to SpiraTest

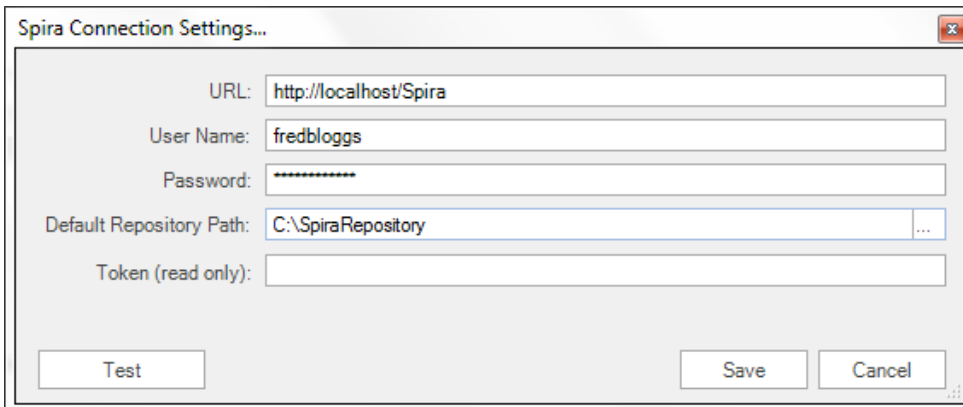
3.1. Configuring the SpiraTest Connection

The first step is to open Rapise on the workstation that will be used to record/create the automated test. Once you have Rapise open, click on the 'Spira Dashboard' icon. This will display the SpiraTest dashboard:



The screenshot shows the Spira Dashboard interface. At the top, there's a navigation bar with icons for 'QuickStart1.js', 'QuickStart1_2014-09-08_14-59.trp', 'Start Page', and 'Spira Dashboard'. Below this is an 'Intro' section with a welcome message: 'Welcome to SpiraTeam. You may Sign Up for free or use your existing SpiraTeam account.' The main area is divided into two panels. The left panel, 'Spira Login/Sign Up', contains 'Login' and 'Sign Up' buttons, and an 'Auto Login' checkbox. The right panel, 'Connection Info', displays the current connection details: 'Spira URL: http://???' and 'User Name:'. Below these, it shows 'Spira Folder:' and 'Local Folder: C:\SpiraRepository'. An 'Edit Connection Settings' button is located at the bottom of this panel.

The first thing we need to do is establish the link to the SpiraTest server. To do this, click on the **[Edit Connection Settings]** button under the 'Connection Info' section. This will bring up the dialog box that lets you configure the connection to SpiraTest:



The screenshot shows the 'Spira Connection Settings...' dialog box. It contains several input fields: 'URL' (http://localhost/Spira), 'User Name' (fredbloggs), 'Password' (masked with asterisks), 'Default Repository Path' (C:\SpiraRepository), and 'Token (read only)'. At the bottom, there are three buttons: 'Test', 'Save', and 'Cancel'.

Enter the following information and then click the [Test] button to verify that the connection information is correct:

- **URL** – This is the URL that you use to connect to SpiraTest. It needs to include either http:// or https://
- **User Name** – this is the login that you use to access SpiraTest
- **Password** – this is the password associated with the user name
- **Default Repository Path** – this is the Windows folder that local copies of the tests opened from SpiraTest will be stored.

Assuming that the test passes, click on the [Save] button to save the settings.

The Spira Dashboard will now display the entered connection information:

Intro

Welcome to SpiraTeam. You may Sign Up for free or use your existing SpiraTeam account.

Spira Login/Sign Up

Login

Sign Up

Auto Login: ☐

Connection Info

Spira URL: http://localhost/Spira

User Name: fredbloggs

Spira Folder:

Local Folder: C:\Temp\RapiseTests

Edit Connection Settings

Now you can login to Spira by clicking on the [Login] button. The system will then log you into SpiraTest (you can choose the 'Auto Login' option if you want to skip this step in the future):

Spira Login/Sign Up

Welcome, fredbloggs!

Logout

Auto Login: ☐

Connection Info

Spira URL: http://localhost/Spira

User Name: fredbloggs

Spira Folder:

Local Folder: C:\Temp\RapiseTests

Edit Connection Settings

Automation Hosts

Windows 7 Host

Token: Win7

Description: Windows 7 with IE9 Beta 2, Firefox 3, Chrome and Safari 4

Create Host for this Machine

Test Cases and Test Sets

Project

Description

Library Information System

Sample application that allows users to manage books, authors and lending records for a typical branch library

Create From Spira Manual Test

Test Cases

My Assigned

✓ Id	Name	Description	Action
TC000001	Functional Tests		
<input type="checkbox"/> TC000002	Ability to create new book	Tests that the user can create a new book in the system	Open

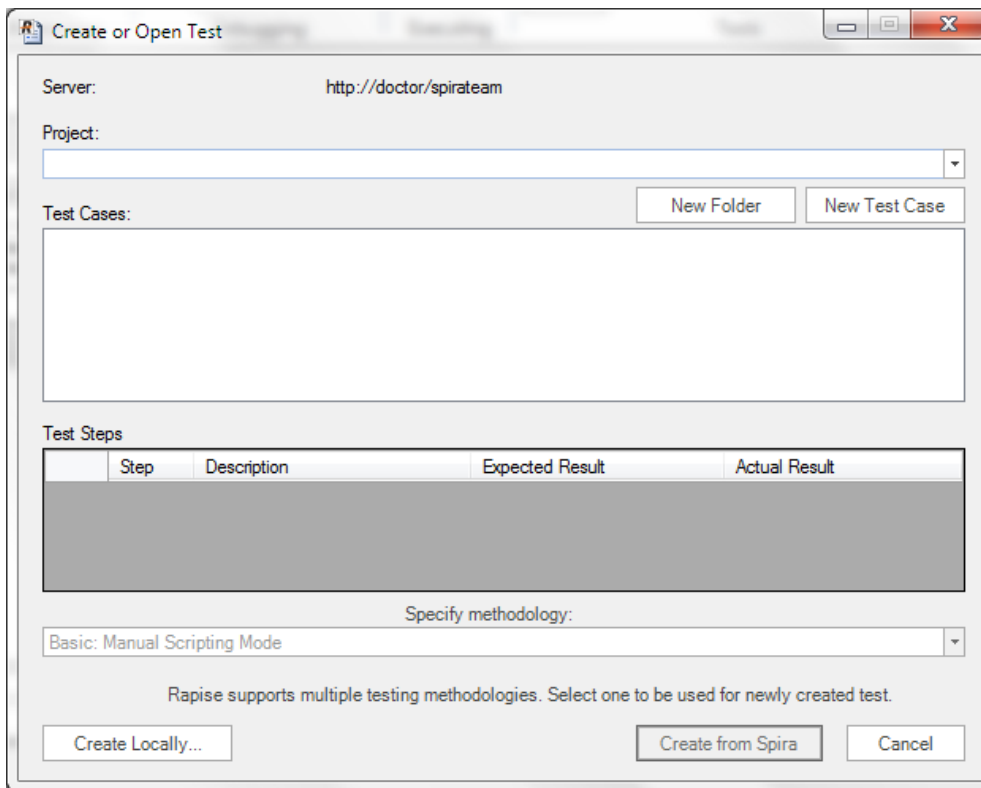
Note: If you don't have an instance of SpiraTest to connect to, click on the 'Sign Up' hyperlink and you will be taken to the SpiraTest signup page on the Inflectra website.

3.2. Creating the Rapise Test from SpiraTest

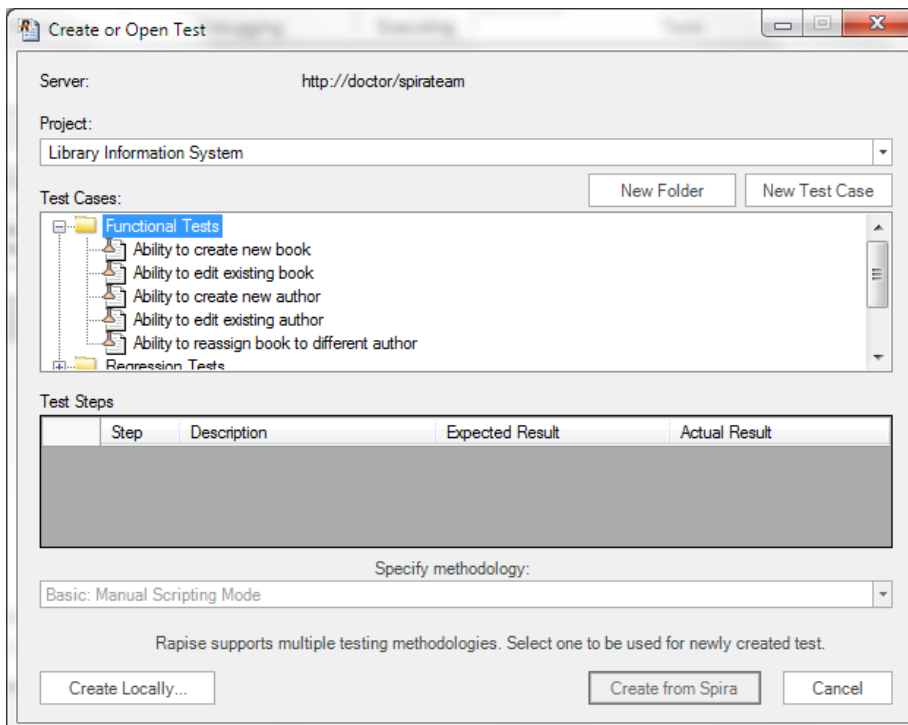
Now that we have established the connection to SpiraTest, the next step is to create our new Rapise test based on the manual test steps already defined in SpiraTest. This will ensure that we test all the required functionality and also enable Rapise to report back results against the individual test steps in SpiraTest.

You can either create a new manual test case (with test steps) that will be used to store the Rapise automated test or simply use an existing test case. In this example we shall create a new test case called "Create New Book -Automated".

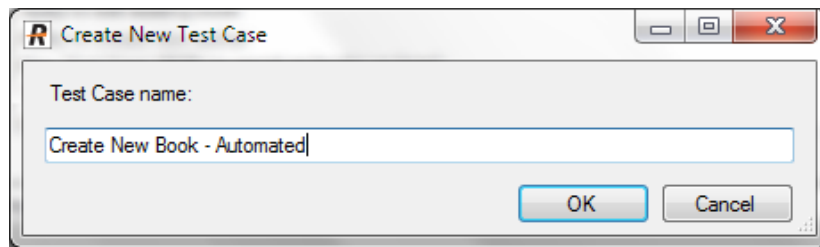
Click on the **File** tab in the top left of the application and from the File menu, choose the option **New Test - Create a New Test**. This will bring up the following dialog box:



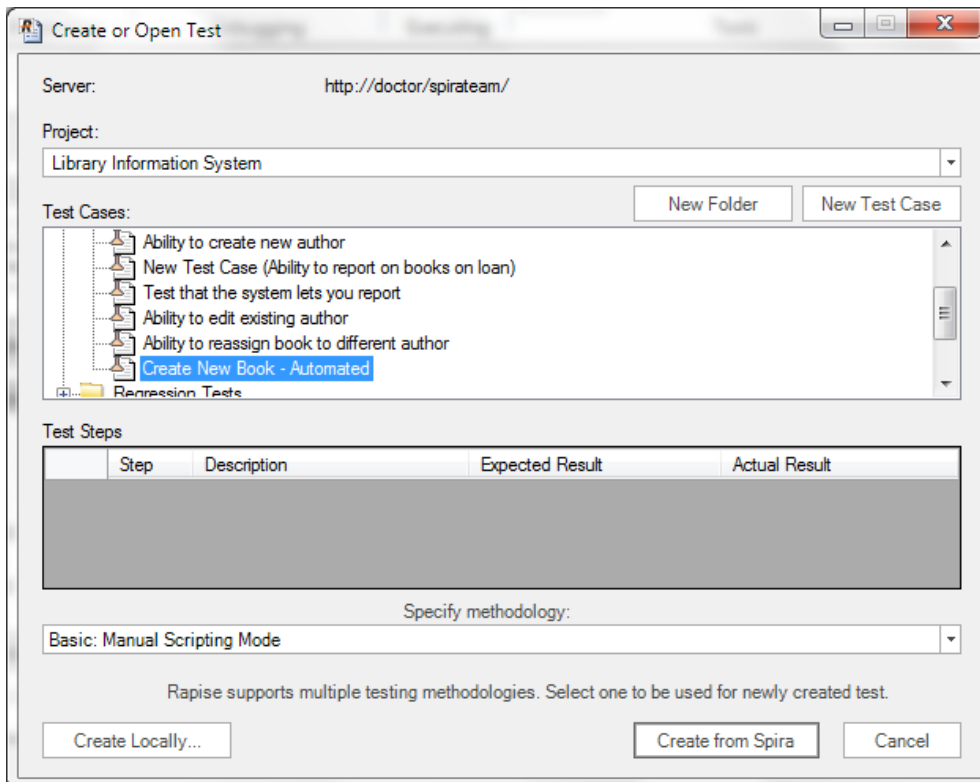
Now, in the “Project” dropdown list, select the project that you want to create the new test case in. The list of test case folders will be displayed. Expand the folders until you can see the location that you’d like to add the new test case (you can click the **[New Folder]** button to create a new folder if desired):



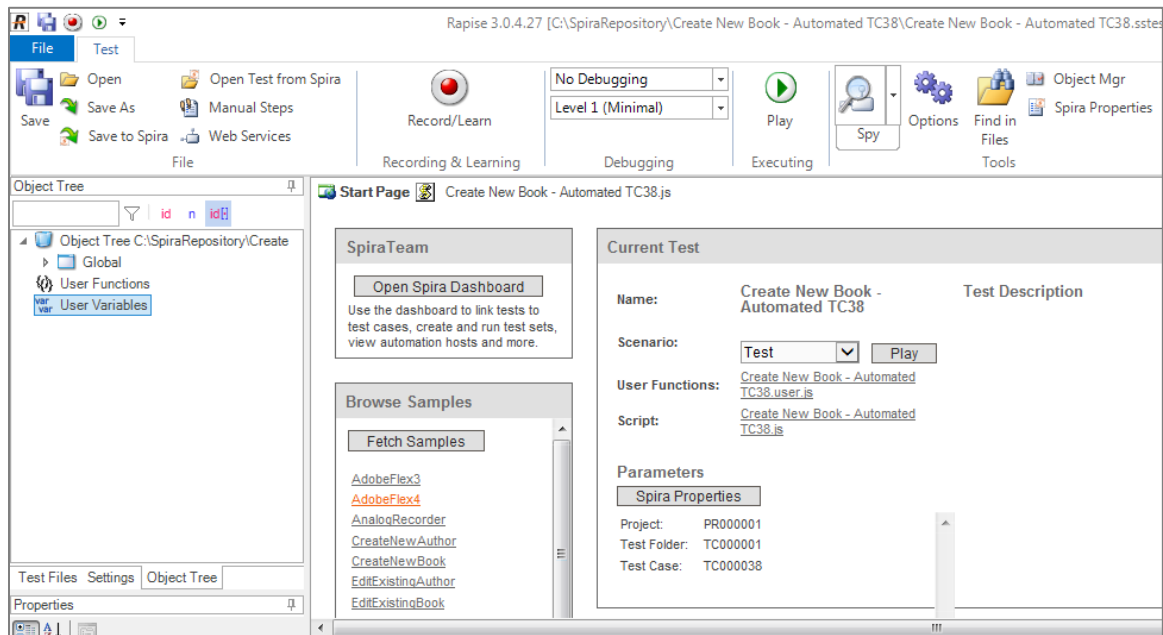
Now select the desired folder and create a new test case by using the **New Test Case** button. This will display the following:



Click on **[OK]** and the new test case will be created. When you click on the test case we just created, you will see its test steps displayed:



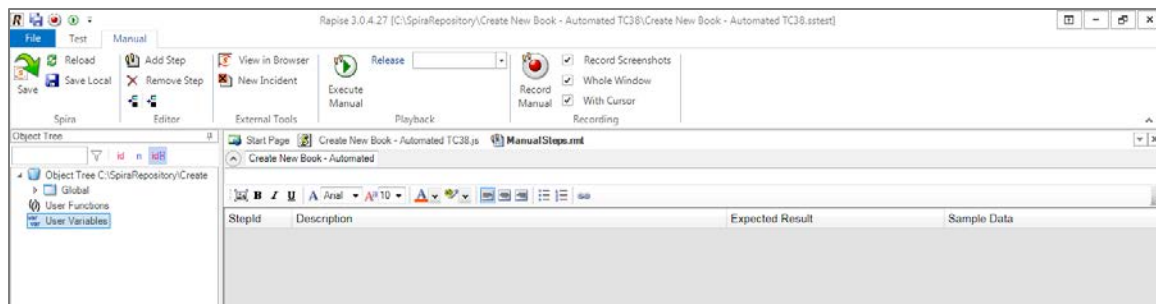
Once you are satisfied that this is the correct test case, click the **[Create from Spira]** button and Rapise will open this new test inside the main editor:



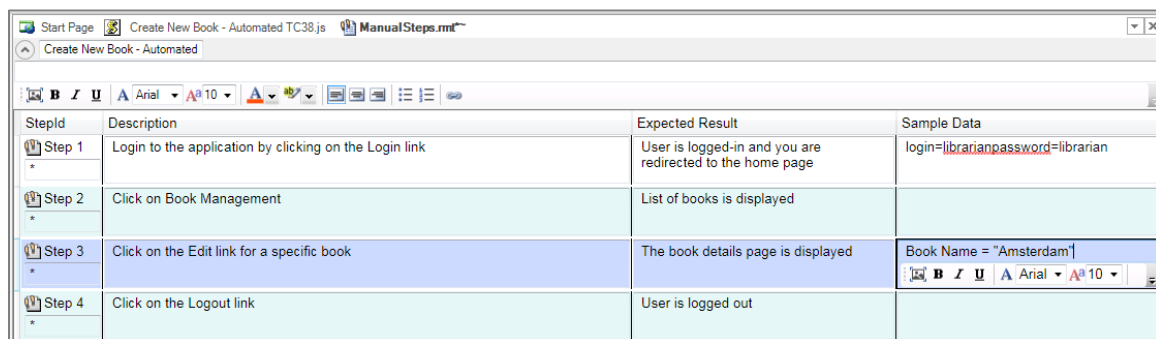
3.3. Creating the Manual Test Steps

If you used an existing SpiraTest manual test case then most likely it already has test steps defined, in which case you can skip this section and move to section 3.4.

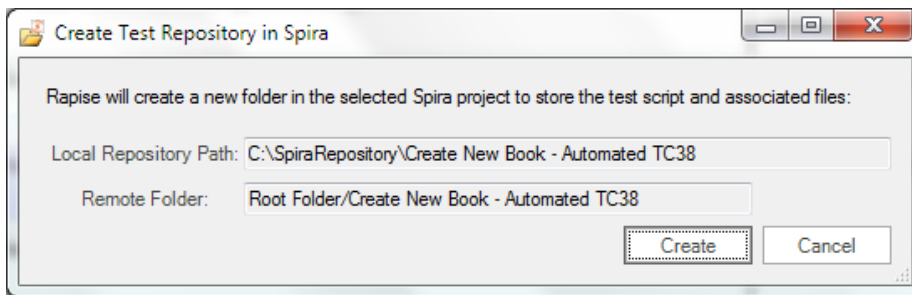
Otherwise, if you created a new test case, there will be no manual test steps and no automated script. In which case we need to first add the **manual test steps** to the test case. Click on the **[Manual Steps]** icon in the Rapise ribbon; this will display the manual test editor:



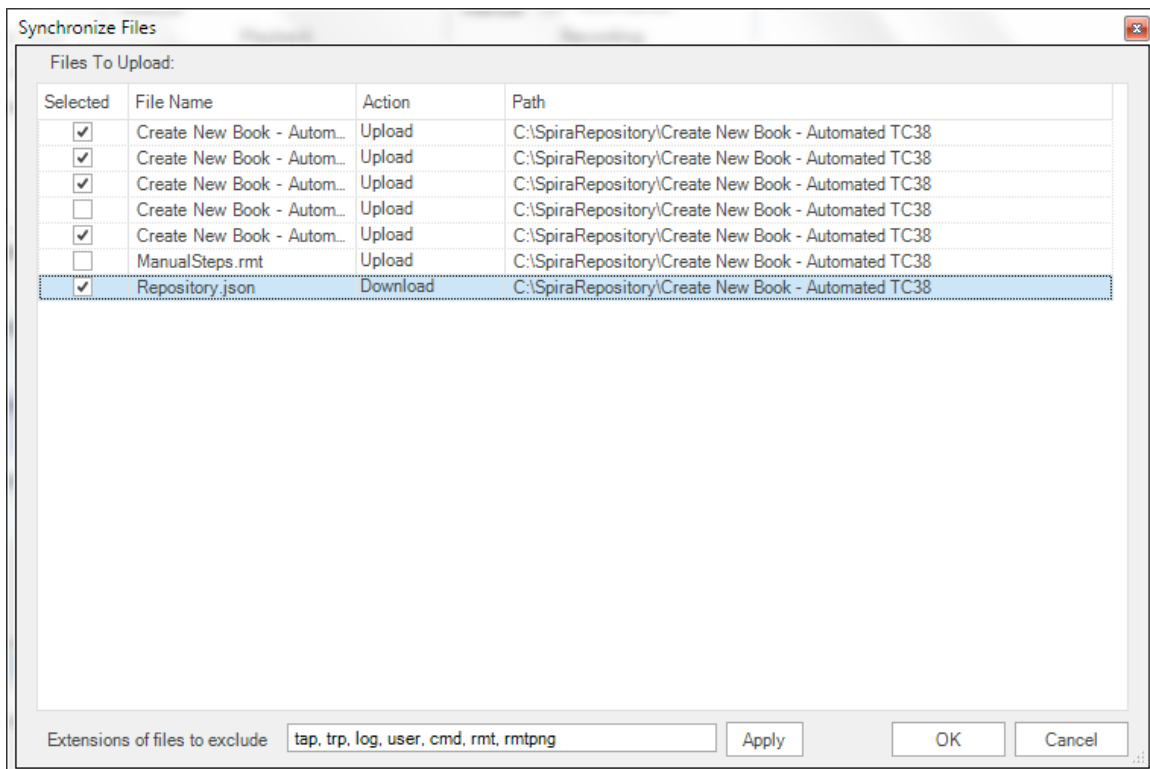
Now use the **'Add Step'** icon in the ribbon to add manual steps to this test case. In the example below we have added four manual steps:



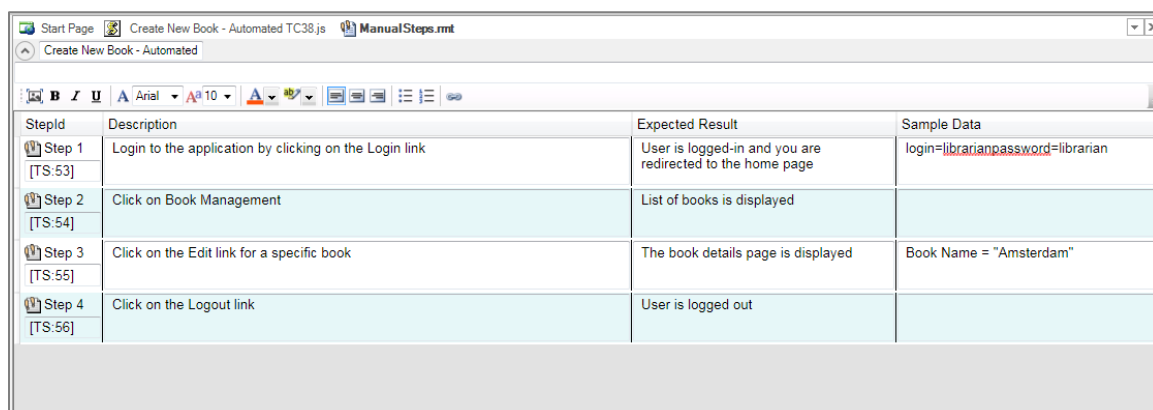
Now click on the **[Save]** icon and the system will display the following dialog box:



This lets you know that Rapise will create a new document repository in SpiraTest to contain the test folder and associated test script files. Click the [Create] button to confirm and then Rapise will display the list of files that are to be uploaded to SpiraTest:



Once you click [OK], the manual steps will be saved to SpiraTest and the test case manual step editor will display the test steps with their IDs in SpiraTest now visible:



If you click on the **'View in Browser'** icon in the ribbon, you can see the exact same manual test steps as they appear in SpiraTest:

▼ Test Steps							
> Insert Step Insert Link Delete Copy Refresh -- Show/Hide columns -- Edit Parameters							
<input type="checkbox"/>	Step #	Test Step Description	Expected Result	Sample Data	Execution Status	ID	<input type="button" value="Edit"/>
<input type="checkbox"/>	Step 1	Login to the application by clicking on the Login link	User is logged-in and you are redirected to the home page	login=librarian password=librarian	Not Run	TS000039	<input type="button" value="Edit"/>
<input type="checkbox"/>	Step 2	Click on Book Management	List of books is displayed		Not Run	TS000040	<input type="button" value="Edit"/>
<input type="checkbox"/>	Step 3	Click on the Edit link for a specific book	The book details page is displayed	Book Name = 'Amsterdam'	Not Run	TS000041	<input type="button" value="Edit"/>
<input type="checkbox"/>	Step 4	Click on the Logout link	User is logged-out		Not Run	TS000042	<input type="button" value="Edit"/>
Show 15 rows per page ◀◀ Displaying page 1 of 1 ▶▶							

Now that we have created the test case, the next step is to create the new Rapise automated test script from this manual test case and upload it into SpiraTest, linked to this particular test case.

3.4. Recording the Automated Test

Now that the new test case “shell” has been created by Rapise, you should use the standard Record/Learn button in the Rapise Test ribbon to actually record the operations against the test application. You should follow the steps you recorded manually to create the same test script. If we follow these steps using the Internet Explorer web browser you will get the following Rapise script:

```

##### Script Steps #####

function Test()
{
    //Click on Log In
    SeS('Log_In').DoClick();
    //Set Text librarian in Username:
    SeS('Username_').DoSetText("librarian");
    //Set Text librarian in Password:
    SeS('Password_').DoSetText("librarian");
    //Click on ct100$MainContent$LoginUser$LoginButton
    SeS('ctl00$MainContent$LoginUser$Logi').DoClick();
    //Click on Book Management
    SeS('Book_Management').DoClick();
    //Click on Edit
    SeS('Edit').DoClick();
    //Click on Log Out
    SeS('Log_Out').DoClick();
}

```

If you now try and execute this test in Rapise it will generate the following results:

Drag a column header here to group by that column.						
#	Type	Start	Name	Status	Comment	Iteration
1	Message	22:23:02.840	Starting scenario: Test	Info		
2	Assert	22:23:04.697	Log In.DoClick([])	Pass	Returned Value: true	0
3	Assert	22:23:06.352	Username:.DoSetText(["librarian"])	Pass	Returned Value: true	0
4	Assert	22:23:07.647	Password:.DoSetText(["librarian"])	Pass	Returned Value: true	0
5	Assert	22:23:08.923	ctl00\$MainContent\$LoginUser\$LoginButton.DoClick([])	Pass	Returned Value: true	0
6	Assert	22:23:10.643	Book Management.DoClick([])	Pass	Returned Value: true	0
7	Assert	22:23:12.309	Edit.DoClick([])	Pass	Returned Value: true	0
8	Assert	22:23:13.745	Log Out.DoClick([])	Pass	Returned Value: true	0
9	Test	22:23:13.750	Create New Book - Automated TC38	Pass	Passed:7 Failed:0	

Test Pass
 Total: 9 Pass: 8 Fail: 0 Info: 1

This test case now tests the steps outlined in the manual test case, however if we execute this from SpiraTest it will only report back the overall result for the test case (pass, fail, etc.) since we have not associated the test steps with specific parts of the automated test script. To do that we need to include a special **test case** or **test step** token (**TC** or **TS**) in the message text of the report.

This is achieved by adding verification **Tester.Assert(...)** code into the test script that checks that the test operation has performed correctly. Each of the statements in the test script should return TRUE if it operated successfully. This would change the test script to look like the following:

```
##### Script Steps #####

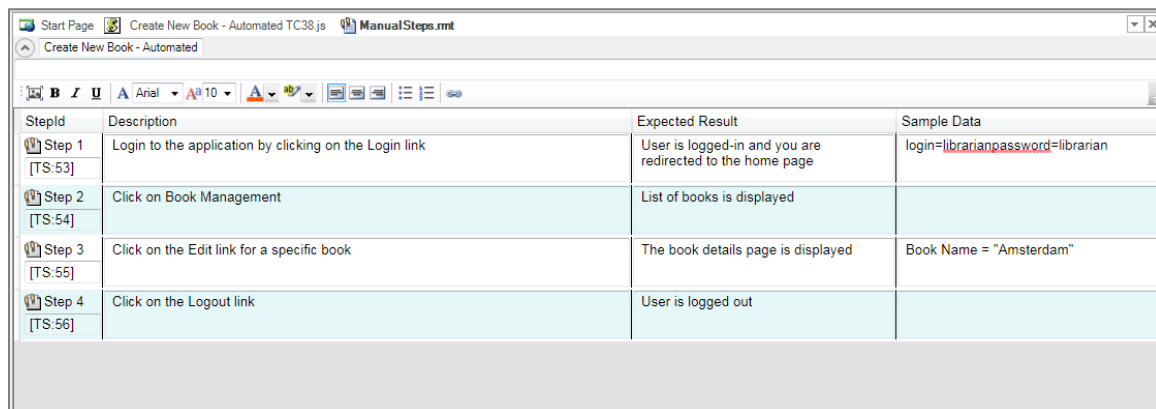
function Test()
{
    //Click on Log In
    SeS('Log_In').DoClick();
    //Set Text librarian in Username:
    SeS('Username_').DoSetText("librarian");
    //Set Text librarian in Password:
    SeS('Password_').DoSetText("librarian");
    //Click on ct100$MainContent$LoginUser$LoginButton
    Tester.Assert(' ', SeS('ct100$MainContent$LoginUser$Logi').DoClick());

    //Click on Book Management
    Tester.Assert(' ', SeS('Book_Management').DoClick());

    //Click on Edit
    Tester.Assert(' ', SeS('Edit').DoClick());

    //Click on Log Out
    Tester.Assert(' ', SeS('Log_Out').DoClick());
}
```

Now to add in the appropriate test step tokens, simply click on 'Manual Steps' to go back to the manual editor:



StepId	Description	Expected Result	Sample Data
Step 1 [TS:53]	Login to the application by clicking on the Login link	User is logged-in and you are redirected to the home page	login=librarianpassword=librarian
Step 2 [TS:54]	Click on Book Management	List of books is displayed	
Step 3 [TS:55]	Click on the Edit link for a specific book	The book details page is displayed	Book Name = "Amsterdam"
Step 4 [TS:56]	Click on the Logout link	User is logged out	

Now if you click on the test step **[TS:xx]** token in the first column (under the step number), it will put that special test step token into the Windows clipboard. You can now add this test step token to your automated script in each of the desired places:

```
##### Script Steps #####

function Test()
{
    //Click on Log In
    SeS('Log_In').DoClick();
    //Set Text librarian in Username:
    SeS('Username_').DoSetText("librarian");
```

```

//Set Text librarian in Password:
SeS('Password_').DoSetText("librarian");
//Click on ctl00$MainContent$LoginUser$LoginButton
Tester.Assert(['TS:53'], SeS('ctl00$MainContent$LoginUser$Logi').DoClick());

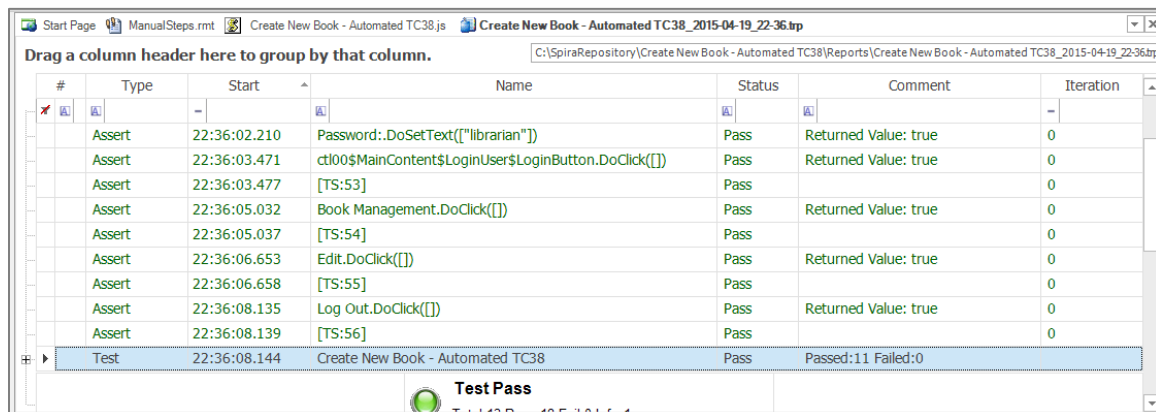
//Click on Book Management
Tester.Assert(['TS:54'], SeS('Book_Management').DoClick());

//Click on Edit
Tester.Assert(['TS:55'], SeS('Edit').DoClick());

//Click on Log Out
Tester.Assert(['TS:56'], SeS('Log_Out').DoClick());
}

```

If you now try and execute this test in Rapise it will generate the following results:



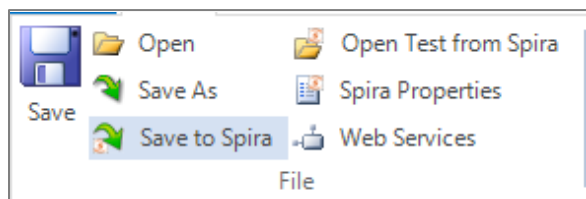
#	Type	Start	Name	Status	Comment	Iteration
	Assert	22:36:02.210	Password:.DoSetText(["librarian"])	Pass	Returned Value: true	0
	Assert	22:36:03.471	ctl00\$MainContent\$LoginUser\$LoginButton.DoClick([])	Pass	Returned Value: true	0
	Assert	22:36:03.477	[TS:53]	Pass		0
	Assert	22:36:05.032	Book Management.DoClick([])	Pass	Returned Value: true	0
	Assert	22:36:05.037	[TS:54]	Pass		0
	Assert	22:36:06.653	Edit.DoClick([])	Pass	Returned Value: true	0
	Assert	22:36:06.658	[TS:55]	Pass		0
	Assert	22:36:08.135	Log Out.DoClick([])	Pass	Returned Value: true	0
	Assert	22:36:08.139	[TS:56]	Pass		0
	Test	22:36:08.144	Create New Book - Automated TC38	Pass	Passed:11 Failed:0	

Test Pass

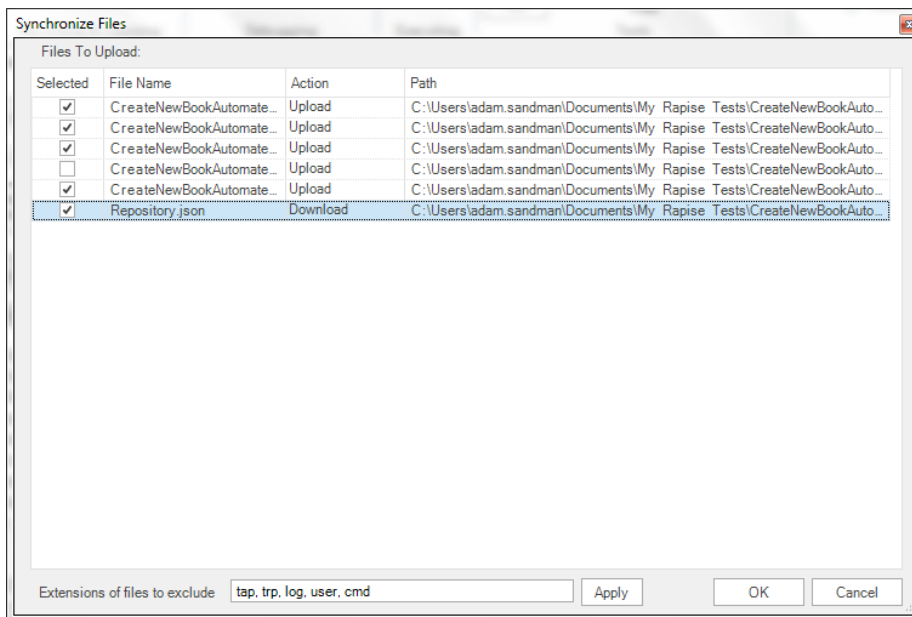
As you can see, we now have the actual test verification points reporting back as Pass or Fail against the SpiraTest test steps (with the TS:XX tokens in the Name or Comment fields of the report). That way if a specific operation fails, it will update the status of the corresponding test case and test step in SpiraTest.

3.5. Saving the Rapise Test to SpiraTest

Now that we have created our test, the next step is to save this automated test script to SpiraTest. To do this, click on the **Save to Spira** icon in the File section of the Rapise Test ribbon:



That will display the list of files that are to be uploaded to SpiraTest:



A dialog box will be displayed that lists all the files in the local working directory and shows which ones will be checked-in to SpiraTest. The system will filter out result and report files that shouldn't be uploaded. You can change which files are filtered out and also selectively include/exclude files. Once you are happy with the list of files being checked-in, click the [OK] button.

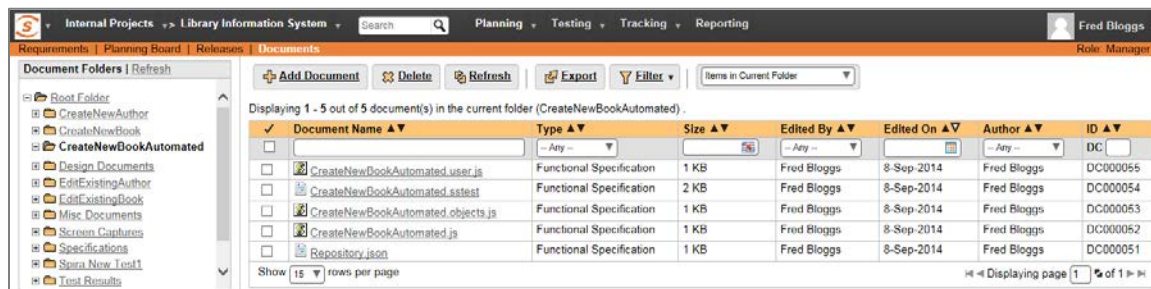
If an error occurs during the save, a message box will be displayed, otherwise the dialog box will simply close. You now have successfully uploaded the Rapise test to SpiraTest. If you make changes, you can synchronize those changes with SpiraTest. SpiraTest has a built-in version control system and will keep a list of all the revisions made to the test.

3.5.1. Synchronizing

Note: When you have a Rapise test previously loaded from SpiraTest or saved to SpiraTest then the standard "Open" and "Save" buttons should not be used. Suppose for example, you have some files modified locally and want to update them in the SpiraTest repository. When you click on the Save/Synchronize button, Rapise will show you which files have been modified locally and which have been modified in the central repository. The synchronization dialog either allows you to upload/download the revisions or keep intact the locally modified files depending on your choice.

3.6. Viewing the SpiraTest Repository

If you open up the project in SpiraTest and click on the Planning > Documents link, you will be taken to the central document repository that now includes your new Rapise test folders:



To see the different revisions of a file, simply click on the hyperlink for a repository item and the list of previous revisions will be displayed:

Document: CreateNewBookAutomated.user.js (DC000055)

Filename*: CreateNewBookAutomated.user.js

Description:

Document Folder*: CreateNewBookAutomated

Document Type*: Functional Specification

Tags:

File Type (Size): JavaScript (1 KB)

Creator*: Fred Bloggs

Edited By*: Fred Bloggs

Creation Date: 9/8/2014 5:53:16 PM

Last Edited: 9/8/2014 5:55:15 PM

Current Version: 2

Active	Version	Version Comments	Size	Author	Upload Date	Operations
✓	Version 2		1 KB	Fred Bloggs	8-Sep-2014	
	Version 1.0		1 KB	Fred Bloggs	8-Sep-2014	> Make Active Delete

> Upload New Version

If you click on the Testing > Test Cases tab and then click on the test case that we previously created you will see that it now has its Automation information populated:

Automation

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

Automation Engine: Rapise

Script Type*: ☐ Attached ☐ Linked ☒ Repository

Filename: CreateNewBookAutomated.sstest

Document Type: Functional Specification

Document Folder: CreateNewBookAutomated

Version: v 1.0

Test Script:

Project File: CreateNewBookAutomated.sstest
 Script Path: CreateNewBookAutomated.js
 User Functions Path: CreateNewBookAutomated.user.js
 Objects Path: CreateNewBookAutomated.objects.js

> Edit Parameters

This shows you that the “Create New Book – Automated” test case is now linked to the corresponding Rapise test stored in the SpiraTest document repository (in the CreateNewBookAutomated folder). You can click on the ‘CreateNewBookAutomated.sstest’ hyperlink and it will automatically take you to the corresponding sstest file in the SpiraTest document repository:

Document: CreateNewBookAutomated.sstest (DC000054)

Filename*: CreateNewBookAutomated.sstest

Description:

Document Folder*: CreateNewBookAutomated

Document Type*: Functional Specification

Tags:

File Type (Size): Unknown (2 KB)

Creator*: Fred Bloggs

Edited By*: Fred Bloggs

Creation Date: 9/8/2014 5:53:16 PM

Last Edited: 9/8/2014 5:53:16 PM

Current Version: 1.0

Active	Version	Version Comments	Size	Author	Upload Date	Operations
✓	Version 1.0		2 KB	Fred Bloggs	8-Sep-2014	

> Upload New Version

3.7. Using Parameterized Test Cases

Often you will have an automated test script that you want to run several times using:

- Different browsers (e.g. Firefox, Chrome and Internet Explorer)
- Different test data

You can define the various test parameters for both these cases and have SpiraTest pass the values through to the Rapise automated test. For example, in the Automation (or Test Steps) section of the new test case, click the “Edit Parameters” link and enter the following information:

Edit Test Case Parameters

The following parameters have been defined for this test case:

Name	Default Value	Operations
{g_book_author}	undefined	Copy To Clipboard Delete
{g_book_genre}	undefined	Copy To Clipboard Delete
{g_book_name}	undefined	Copy To Clipboard Delete

Add a new parameter to this test case:

Name*:

Default Value*:

We have defined four input parameters for this test case:

- **g_book_author** – This contains the name of the author for the new book being created
- **g_book_genre** – This contains the name of the genre for the new book being created
- **g_book_name** – This contains the title/name of the new book being created
- **g_browser_library** – This contains the name of the browser we should use to run the test

Now for these parameters to actually affect the Rapise test, you need to make sure that the Rapise test is expecting these variables and knows how to handle them. Conventionally in Rapise, all **global variables** are prefixed with **g_** which is why we have similarly named the SpiraTest parameter names. In the sample Library Information System tests that come with SpiraTest, we have the following code:

```
function TestInit()
{
    //Input variables - allows SpiraTest to configure as parameters
    //Provide backup values if not defined
    if ('undefined' == typeof(g_book_name))
    {
        g_book_name = 'The Restaurant at the end of the Universe';
    }
    if ('undefined' == typeof(g_book_author))
    {
        g_book_author = 'Agatha Christie';
    }
    if ('undefined' == typeof(g_book_genre))
    {
        g_book_genre = 'Science Fiction';
    }
    if ('undefined' == typeof(g_browser_library))
    {
        g_browser_library = "Internet Explorer HTML";
    }

    Tester.SetReportAttribute("Browser", g_browser_library);
}
```



```
} KillBrowser();
```

This code will check to see if the variables are provided by SpiraTest and if not, it will use some defaults. This is useful when you want to be able to run the test directly from Rapise and from SpiraTest without having to make changes to the test script.

These global variables (`g_book_name`, `g_book_author`, `g_book_genre`) can now be used in the Rapise test script at the appropriate points in the playback.

The `g_browser_library` variable is used to specify which browser should run the test. This is done with the following command, located in the `CreateNewBook.js` file:

```
g_load_libraries=["%g_browser_library:Internet Explorer HTML%"];
```

This tells Rapise to use either the `g_browser_library` variable (if defined) or fallback to using Internet Explorer if not.

Note: Parameters in Rapise and SpiraTest are case-sensitive, so make sure that your parameter names in SpiraTest match those in Rapise exactly, including the specific case.

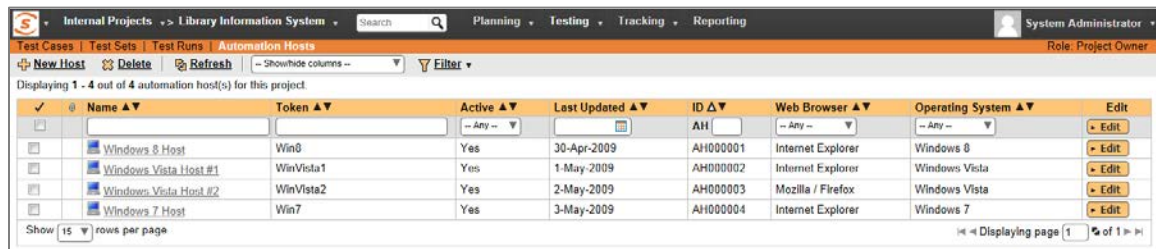
4. Scheduling the Tests

Now that we have our Rapise automated test uploaded to SpiraTest and associated with a test case, we can now schedule the test to be executed. The first thing we need to do is setup the list of automation hosts.

4.1. Configuring the Automation Hosts

When you execute Rapise automated tests from SpiraTest, you have the ability to specify which computer(s) it will be executed on. We call those different computers, the “automated hosts”. Each automation host needs to have a copy of Rapise installed on it.

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



The screenshot shows the 'Automation Hosts' table in SpiraTest. The table has columns: Name, Token, Active, Last Updated, ID, Web Browser, Operating System, and Edit. There are four rows of automation hosts listed.

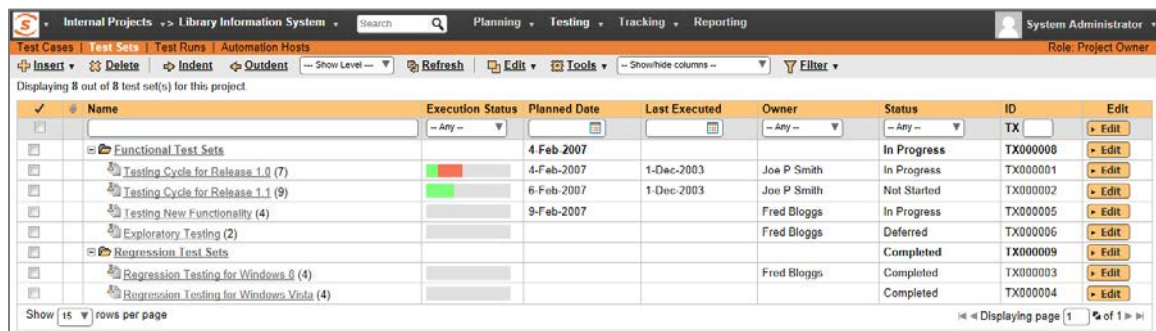
Name	Token	Active	Last Updated	ID	Web Browser	Operating System	Edit
Windows 8 Host	Win8	Yes	30-Apr-2009	AH000001	Internet Explorer	Windows 8	Edit
Windows Vista Host #1	WinVista1	Yes	1-May-2009	AH000002	Internet Explorer	Windows Vista	Edit
Windows Vista Host #2	WinVista2	Yes	2-May-2009	AH000003	Mozilla / Firefox	Windows Vista	Edit
Windows 7 Host	Win7	Yes	3-May-2009	AH000004	Internet Explorer	Windows 7	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 you like, but the Token field must be set to a **unique name for each computer**.

Once you have at least one Automation Host configured, we need to next create the test sets that will be scheduled to execute on these hosts.

4.2. Creating and Scheduling the Test Sets

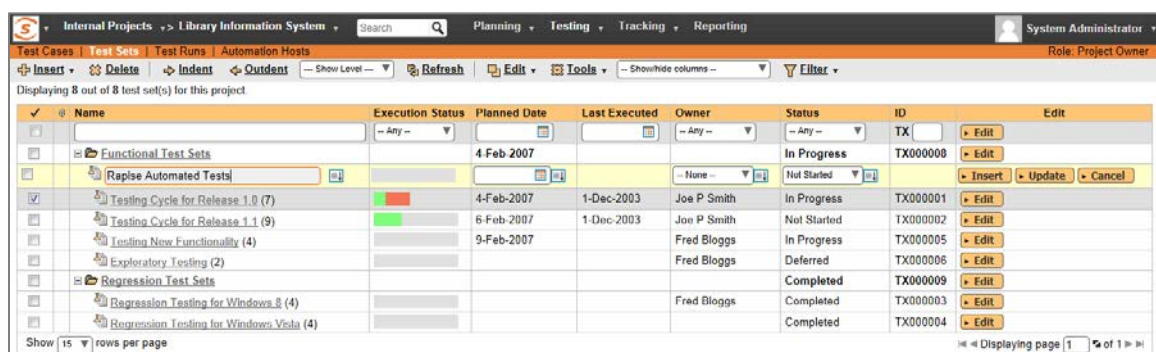
Inside SpiraTest, click on Testing > Test Sets to display the list of existing test sets:



The screenshot shows the 'Test Sets' table in SpiraTest. The table has columns: Name, Execution Status, Planned Date, Last Executed, Owner, Status, ID, and Edit. There are several test sets listed, including Functional Test Sets and Regression Test Sets.

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

Now click on **Insert** to create a new test set that will contain our automated test case:



The screenshot shows the 'Test Sets' table in SpiraTest after a new test set has been inserted. The new test set, 'Rapise Automated Tests', is highlighted in yellow and has an 'Insert' button next to it.

Name	Execution Status	Planned Date	Last Executed	Owner	Status	ID	Edit
Functional Test Sets		4 Feb 2007			In Progress	TX000008	Edit
Rapise Automated Tests					Not Started		Insert Update Cancel
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

Test Set: Rapise Automated Tests [TX:000015]

Name: Rapise Automated Tests

Overview | Test Runs | Attachments | Incidents | History *

Details

Owner: -- None -- Creator*: System Administrator
 Release: -- None -- Type: Manual
 Automation Host: -- None -- Creation Date: 2/25/2013 10:10:14 AM
 Status*: Not Started Last Executed: -
 Planned Date: -- One Time -- Last Updated: 2/25/2013 10:23:50 AM
 Notes: -- Font -- -- Size -- B I U [text formatting 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"/>	Test Case Name	Owner	Priority	Est. Dur.	Act. Dur.	Last Executed	Execution Status	ID	Edit
<input type="checkbox"/>	Create New Book - Automated						Not Run	TC000026	Edit
<input type="checkbox"/>	Create New Book - Automated						Not Run	TC000026	Edit

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

Since in this example we have parameterized test cases inside the automated test set, we need to set their values by right-clicking on each test case in turn and choose “Edit Parameters”:

Edit Test Case Parameters

Please fill out the parameters for this test case entry:

g_book_author: Charles Dickens
 g_book_genre: Historical Fiction
 g_book_name: A Tale of Two Cities
 g_browser_library: Firefox HTML

> Update | Cancel

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 of the test case.

Now that we have the test case added to the set, we need schedule the test set for execution by filling in the following fields:

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: 2/25/2013 10:10:14 AM
 Status*: Not Started Last Executed: -
 Planned Date: 02/25/2013 11:00 am -- One Time -- Last Updated: 2/25/2013 10:23:50 AM

- **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 RapiseLauncher 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

4.3. Executing the Test Sets

Once you have set the various test set fields (as described above), the RapiseLauncher instance running on the assigned automation host will periodically poll SpiraTest for new test sets. Once it retrieves the new test set, it will add it to its list of test sets to be execute. Once execution begins it 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” – RapiseLauncher was not able to execute the automation test.

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 RapiseLauncher 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 Rapise test:

- **Passed** – The Rapise automated test ran successfully and all the test conditions in the test script passed
- **Failed** – The Rapise automated test ran successfully, but at least one test condition in the test script failed.
- **Blocked** – The Rapise 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 RapiseLauncher and look in the event log for error messages.

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

Once the tests have completed, you can log back into SpiraTest and see the execution status of your test sets:

Name	Execution Status	Planned Date	Last Executed	Owner	Status	ID	Edit
Functional Test Sets		4-Feb-2007			In Progress	TX000008	Edit
Rapise Automated Tests (2)		25-Feb-2013	25-Feb-2013		Completed	TX000015	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

If you click on a Test Run in that test set, you will see the following information:

Test Run: Create New Book - Automated [TR:000045]

Overview ▾ Attachments Incidents

▼ Details

Release #: 1.0.0.0 - Library System Release 1

Tester Name: System Administrator

Test Set: Rapise Automated Tests

Test Case #: TC000026

Build: -- None --

Web Browser: -- Please Select --

Notes:

Estimated Duration: 0.00 hours

Actual Duration: 0.00 hours

Execution Date: 2/25/2013 12:44:06 PM

Execution Status: **Failed**

Test Run Type: Automated

Operating System: -- Please Select --

▼ Test Steps

ID	Test Step Description	Expected Result	Sample Data	Test # / Step #	Actual Result	Execution Status
RS000240	TARDIS:adam.sandman	Session		/		Not Run
RS000241	CreateNewBook-Automated	Test	Test Started: C:\Temp\RapiseTests\CreateNewBook-Automated\CreateNewBook-Automated.js	/		N/A
RS000242	Starting scenario: Test	Message		/		N/A
RS000243	IE Process Created	Assert		/	Path "C:\Program Files (x86)\Internet Explorer\EXPLORE.EXE" "http://localhost/libraryinformationssystem/" PID: 8024 > View Incidents	Passed
RS000244	Log In.DoClick()	Assert		/	Returned Value: true > View Incidents	Passed
RS000245	Username: DoSetText("librarian")	Assert		/	Returned Value: true > View Incidents	Passed
RS000246	Password: DoSetText("librarian")	Assert		/	Returned Value: true > View Incidents	Passed
RS000247	ctl00 \$MainContent\$LoginUser\$LoginButton.DoClick()	Assert		/	Returned Value: true > View Incidents	Passed
RS000248	ctl00 \$MainContent\$LoginUser\$LoginButton.DoClick()	Unknown		/	Exception in EnsureVisible: Permission denied > View Incidents	Passed
RS000249	Log Out.DoEnsureVisible()	Assert		/	Returned Value: false > View Incidents	Failed
RS000250	[TC:26]/[TS:39] - Login to the application by clicking on the Login	Assert		/ TS000039	User is logged-in and you are redirected to the h > View Incidents	Failed
RS000251	Book Management.DoClick()	Assert		/	Returned Value: true > View Incidents	Passed
RS000252	MainContent_grdBooks.DoGetRect()	Assert		/	Returned Value: [object Object] > View Incidents	Passed
RS000253	MainContent_grdBooks.DoEnsureVisible()	Assert		/	Returned Value: [object Object] > View Incidents	Passed
RS000254	[TC:26]/[TS:40] - Click on Book Management	Assert		/ TS000040	List of books is displayed > View Incidents	Passed
RS000255	Edit.DoClick()	Assert		/	Returned Value: true > View Incidents	Passed
RS000256	[TC:26]/[TS:41] - Verify that Value=Amsterdam	Assert		/ TS000041	The book details page for Amsterdam is displayed > View Incidents	Passed
RS000257	Log Out.DoClick()	Assert		/	Returned Value: true > View Incidents	Passed
RS000258	Log In.DoGetRect()	Assert		/	Returned Value: [object Object] > View Incidents	Passed
RS000259	Log In.DoEnsureVisible()	Assert		/	Returned Value: [object Object] > View Incidents	Passed
RS000260	[TC:26]/[TS:42] - Click on the Logout link	Assert		/ TS000042	User is logged-out > View Incidents	Passed
RS000261	CreateNewBook-Automated	Test		/	Passed 15 Failed 2 > View Incidents	Failed

▼ Console Output

Runner Name: Rapise

Automation Host: Windows 7 Host

Message: 16 Steps Passed, 2 Steps Failed (88% Success Rate)

Assert Count: 0

Test Name: CreateNewBook-Automated

Details:

```

Command Output
=====
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.

SeS Executor Starting...
Verbose level: 1
RUNNING: C:\Temp\RapiseTests\CreateNewBook-Automated\CreateNewBook-Automated.sstest

STATUS: FAIL

Test was successfully executed!

Script Execution Report
=====

```

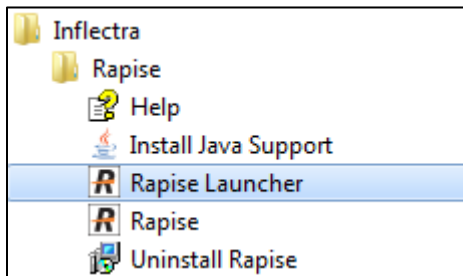
If you have configured SpiraTest to capture screenshots during execution (see section 5.2 below), there may also be screenshots visible in the Attachments tab of the test run:

Test Run: Create New Book - Automated [TR:000048]							
<div> <div>Overview *</div> <div>Attachments *</div> <div>Incidents</div> </div>							
<div> > Add New Add Existing Remove Refresh Apply Filter Clear Filter <input type="checkbox"/> Include Source Code Documents </div>							
Displaying 1 - 13 out of 13 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"/>	Step9.png	Functional Specification	157 KB	System Administrator	25-Feb-2013	System Administrator	DC000089
<input type="checkbox"/>	Step8.png	Functional Specification	157 KB	System Administrator	25-Feb-2013	System Administrator	DC000088
<input type="checkbox"/>	Step7.png	Functional Specification	152 KB	System Administrator	25-Feb-2013	System Administrator	DC000087
<input type="checkbox"/>	Step6.png	Functional Specification	153 KB	System Administrator	25-Feb-2013	System Administrator	DC000086
<input type="checkbox"/>	Step5.png	Functional Specification	153 KB	System Administrator	25-Feb-2013	System Administrator	DC000085
<input type="checkbox"/>	Step4.png	Functional Specification	153 KB	System Administrator	25-Feb-2013	System Administrator	DC000084
<input type="checkbox"/>	Step3.png	Functional Specification	130 KB	System Administrator	25-Feb-2013	System Administrator	DC000083
<input type="checkbox"/>	Step2.png	Functional Specification	130 KB	System Administrator	25-Feb-2013	System Administrator	DC000082
<input type="checkbox"/>	Step13.png	Functional Specification	154 KB	System Administrator	25-Feb-2013	System Administrator	DC000081
<input type="checkbox"/>	Step12.png	Functional Specification	154 KB	System Administrator	25-Feb-2013	System Administrator	DC000080
<input type="checkbox"/>	Step11.png	Functional Specification	129 KB	System Administrator	25-Feb-2013	System Administrator	DC000079
<input type="checkbox"/>	Step10.png	Functional Specification	129 KB	System Administrator	25-Feb-2013	System Administrator	DC000078
<input type="checkbox"/>	Step1.png	Functional Specification	151 KB	System Administrator	25-Feb-2013	System Administrator	DC000077
<div> <div>Show 15 ▼ rows per page</div> <div> <div>◀◀</div> <div>Displaying page 1</div> <div>of 1 ▶▶</div> </div> </div>							

So, you now have a complete record of the automated test execution in SpiraTest, with the execution status of the appropriate test case and test steps updated, and a complete log of the testing activities.

5. Configuring RapiseLauncher

RapiseLauncher is an add-on for Rapise that installs along with the main Rapise application. It can be found in the Start > Programs > Inflectra > Rapise program folder:



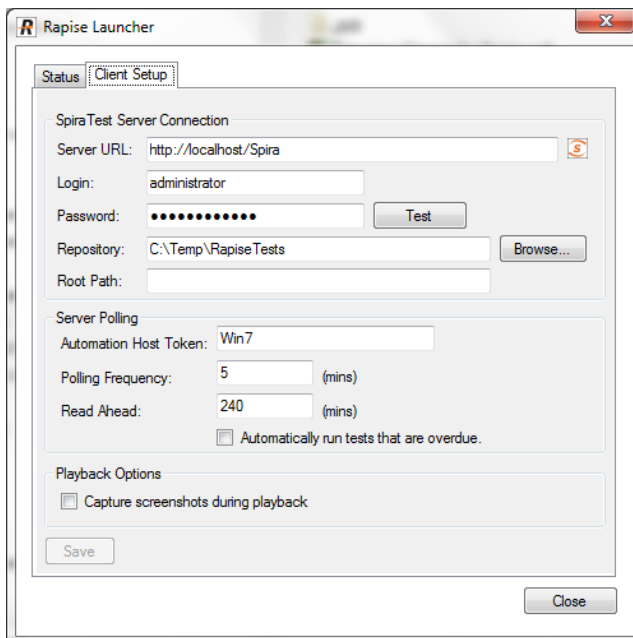
5.1 Basic Unattended Operation

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 Rapise to execute the test. Rapise will then perform the test activities and report the results back to SpiraTest.

At the end of the test, the program will go back and resume scanning for tests that need to be executed. Typically (unless there is a bug in the test or application being tested) no user input is ever needed from the application itself.


5.2. 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:

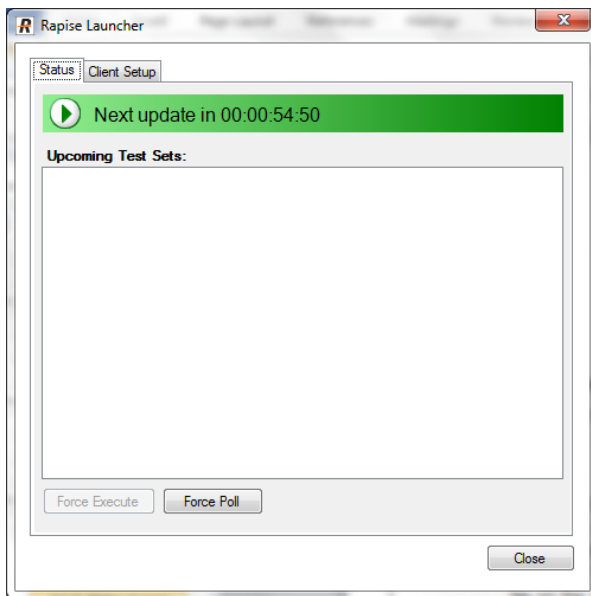
- SpiraTest Server Connection:
 - **Server URL:** This is the URL of the SpiraTest installation. Be sure to not put /Login.aspx or any other page in the string, this should be just the root URL of the application's install.

- **Login Username:** This is the 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.
 - **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.
 - **Repository** – this is the Windows folder that used to store local copies of the non-absolute repositories (downloaded from SpiraTest).
 - 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 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.
- **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.
 - **Polling Frequency:** How often in minutes the application will poll the SpiraTest 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 SpiraTest 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.
 - Playback Options
 - **Capture screenshots during playback** – selecting this option will instruct RapiselLauncher to capture screenshots of the objects being recognized during testing and upload them to SpiraTest at the end of the execution. The screenshots will then be linked to the test run inside SpiraTest.

5.3. 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:

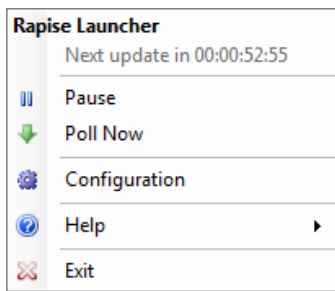
- **Green Arrow:** A green arrow indicates that the test is still running, or RapiseLauncher is waiting for a reply from Rapise.
- **Blue Checkbox:** A blue checkbox indicates that the test is completed, regardless of status of the individual test steps in the scheduled test set.
- **Red Error:** A red error indicator indicates that Rapise ran into an issue (outside of test results). In this case, any further tests will be marked as blocked, as the issue needs to be corrected within Rapise or the Rapise test script.
- **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 RapiseLauncher to initiate an immediate poll of the SpiraTest server to check for pending runs. The timers for the next server poll will be reset when the button is clicked.

5.4. 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 information about Rapise Launcher.
- **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 tray icon to bring up the main window on the Status page.

5.5. Running RapiseLauncher from a Build Script

Normally you schedule tests in SpiraTest using the Planned Date field of the test sets and let the various instances of RapiseLauncher poll SpiraTest for upcoming tests. In addition (as described in the *SpiraTest 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 Rapise test script 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 SpiraTest. You can achieve this by using the special command-line argument `-testset` which is passed to RapiseLauncher. For more details on this parameter see the next section.

5.6. 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).
-testset:[Test Set ID]	Allows you to tell RapiseLauncher 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 SpiraTest to start immediate execution on.

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®, Rapise® and Inflectra® are 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. 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