

Rapise User Manual Version 5.4 Inflectra Corporation

Monday, November 6, 2017



© 2017 Inflectra Corporation

Table of Contents

	Foreword	0
Part I	Company & Copyright	6
Part II	Rapise User's Guide	7
1	About this Guide	7
2	Getting Started	
	Overview	
	Samples Index	
	Tutorial: Web Testing	
	Using JavaScript	
	Tutorial: Windows Testing	
	Using JavaScript	
	Tutorial: REST Web Services	
	Tutorial: SOAP Web Services	
	Tutorial: Mobile Testing	
	Tutorial: Manual Testing	
	Tutorial: Java Testing	115
	Using JavaScript	
	Tutorial: Qt Framework	
	Using JavaScript	
3	Features	150
	Recording and Learning	
	Recording	
	Learning	
	Analog Recording	
	Absolute Analog Recording	
	Relative Analog Recording	
	Simulated Objects	
	Object Libraries	
	Custom Libraries	
	Actions	
	Multiple Recordings	
	Object Spy	
	Accessible (MSAA) Spy	
	Java Spy	
	Mobile Spy	
	Managed (.NET) Spy	
	UI Automation Spy	
	Web Spy	
	Object Manager	
	Playback	
	Command Line	
	Object Locator	
	Automated Reporting	
	Writing to the Report	
	Report Filtering	105

Contents	3

Scripting	201
Understanding the Script	
Naming Conventions	
Defining Functions	203
Global Variables	
Including other Files	
Regular Expressions	
Assert Statements	
Data Driven Testing	
Customizable Engine	
Scenarios	
Javascript IDE	
Internal Debugger	
Tooltips	
Control Execution	
Breakpoints	
External Debugger	
Verbosity Levels	
Syntax Highlighting	
Code Folding	
Syntax Checking	
Code Completion	
Unit Testing	
Visual Studio	
NUnit	
DLL Testing	
COM Testing Support	
Custom Strings	
TAP Results	
Web Service Testing	
Testing REST Web Services	
Testing SOAP Web Services	
Mobile Testing	254
Apple iOS	
Android	
Manual Testing	
Manual Recording	
Manual Playback	
Semi-Manual Testing	
SpiraTest Integration	
Checkpoints	
Tests and Sub-Tests	
NeoLoad Integration	
Convert Functional to Load Test	
Client Performance Monitoring	
alogs, Views, and Menus	
Add Web Service Dialog	
Browser Settings Dialog	
Create New Test Dialog	
Create Sub-Test Dialog	
Content View	328
Enter filter criteria for Dialog	329
Errors View	331
Find and Replace Dialog	

	Find Results View	333
	Find Text dialog	334
	Image Capture	334
	Incident Logging	337
	Manual Playback	340
	Manual Test Editor	342
	Mobile Settings Dialog	344
	Mobile Test Locator Dialog	348
	NameValue Collection Editor Dialog	
	NeoLoad Convertor Dialog	
	NeoLoad Settings Dialog	
	Object Tree Dialog	
	Options Dialog	
	Output View	
	Properties Dialog	
	Recording Activity Dialog	
	Replace Text Dialog	
	Report Viewer	
	REST Definition Editor	
	Ribbon: Test	
	Ribbon: Report	
	Ribbon: RVL	
	Ribbon: Spreadsheet	
	Ribbon: Edit	
	Ribbon: Debugger	
	Ribbon: Manual	
	Ribbon: REST	
	Ribbon: SOAP	
	Ribbon: Options	
	Scripting Choice Dialog	
	Select an Application to Record Dialog	
	Selenium Settings Dialog	390
	Settings Dialog	393
	SOAP Definition Editor	396
	Source Editor	400
	Spreadsheet Viewer	401
	Start Page	402
	Spira Dashboard	403
	Spira Properties Dialog	408
	Spy Dialog	408
	Test Files Dialog	417
	Variable/Call Stack View	419
	Verify Object Properties Dialog	420
	Warning View	
	Watch View	422
	File Menu	424
	Web Spy Settings	424
Ηον	vTos	
	Open a Test	
	Create a New Test	
	Learn an Object	
	Restoring the Default Layout	
	Change Test Entry Point	
	Deal with a Simulated Object	436

5

	Do Absolute Analog Recording	
	Do Relative Analog Recording	
	Changing the URL of Website being Tested	
	Extracting test data from an Excel spread sheet	
	Accessing Files and I/O Functions	
	Sending Special Keys to the Current Application	445
	Detecting the presence of an object	447
	Change the libraries being used	447
6	Technologies	449
	Web Testing	
	Cross Browser Testing	
	Setting Up Web Browsers	
	XPath	
	CSS	
	Selenium WebDriver	
	Setting Up Selenium	
	Using Native Selenium Code	
	Mobile Testing	
	Mobile Testing: iOS Setup	
	Microsoft Dynamics	
	Dynamics AX	500
	Dynamics CRM	
	Dynamics 365	516
	Windows Applications	521
	Salesforce.com	
	Qt Framework	
	Java AWT/Swing	
	Java SWT	
7	Extensibility	542
	Tutorial: Custom Library	
8	Glossary	551

Index

552

1 Company & Copyright



This documentation and the software it describes is the proprietary and copyrighted intellectual property of Inflectra Corporation,

© All Rights Reserved. Rapise®, Inflectra®, SpiraTest®, SpiraTeam® and Spira™ are either trademarks or registered trademarks of Inflectra Corporation.

2 Rapise User's Guide

2.1 About this Guide

The Rapise User's Guide is divided into four sections: Getting Started; Features; Dialogs, Views, and Menus; HowTos.

Getting Started

The Getting Started section is for new Rapise users. It has the following subsections:

- 1. An Overview of Rapise: what it's for and how to use it.
- 2. Samples Index, where the sample projects included with Rapise are described.
- 3. <u>Tutorial: Windows Testing</u>, a step-by-step tutorial for creating your first test with Rapise using a Windows desktop application.
- 4. <u>Tutorial: Web Testing</u>, a slightly more advanced tutorial in using Rapise to test a web page.
- 5. <u>Tutorial: REST Web Services</u>, a tutorial in using Rapise to test a RESTful web service API.
- 6. <u>Tutorial: SOAP Web Services</u> a tutorial in using Rapise to test a SOAP web service API.
- 7. <u>Tutorial: Mobile Testing</u> a tutorial explaining how to use Rapise to test a mobile application (in this case using Android)
- 8. Tutorial: Manual Testing a tutorial explaining how to use Rapise to do exploratory / manual testing.
- 9. <u>Tutorial: Java Testing</u> a step-by-step tutorial for creating a Rapise test for Java AWT, Swing and SWT applications

10. <u>Tutorial: Qt Framework</u> - a step-by-step tutorial for creating a Rapise test for Qt Framework desktop applications.

Features

The features of Rapise are many. The features have been designed to make all aspects of test automation as easy as possible.

Most of the features of Rapise fall into one of five categories:

- 1. Building test scripts with little or no manual scripting.
- 2. Reading and interpreting results and reports.
- 3. Additional features and capabilities for sophisticated testing.
- 4. Writing more involved or complicated tests using scripting.
- 5. Extending Rapise to learn new or extended libraries of capabilities.

Depending on the application set being tested, not all of these features are necessarily needed for every situation.

For each feature, this document attempts to present:

- 1. The reason you might use a given feature.
- 2. A summary of the basic value of the feature.

- 3. An overview of how the feature works from the perspective of using it.
- 4. At least one useful sample that demonstrates how to use the feature.

Dialogs, Views, and Menus

This section details the Rapise GUI. Each subsection describes the function of a particular Dialog, View, or Menu. The purpose and consequences of all buttons, options, lists, and check boxes are listed.

How-Tos

This section focuses on specific tasks that a Rapise user might want to accomplish.

2.2 Getting Started

The Getting Started section is for new Rapise users. It has the following subsections:

- 1. An Overview of Rapise: what it's for and how to use it.
- 2. <u>Samples Index</u>, where the sample projects included with Rapise are described.
- <u>Tutorial: Windows Testing</u>, a step-by-step tutorial for creating your first test with Rapise using a Windows desktop application.
- 4. Tutorial: Web Testing, a slightly more advanced tutorial in using Rapise to test a web page.
- 5. <u>Tutorial: Testing REST Web Services</u>, a tutorial in using Rapise to test a RESTful web service API.
- 6. Tutorial: Testing SOAP Web Services, a tutorial in using Rapise to test a SOAP web service API.
- <u>Tutorial: Mobile Testing</u> a tutorial explaining how to use Rapise to test a mobile application (in this case using Android)
- 8. <u>Tutorial: Manual Testing</u> a tutorial explaining how to use Rapise to create and execute <u>manual</u> <u>tests</u>.
- <u>Tutorial: Java Testing</u> a tutorial explaining how to use Raise to test some simple AWT, Swing and SWT applications
- 10. Tutorial: Qt Framework a tutorial explaining how to use Rapise to test a simple Qt Framework application

2.2.1 Overview

Why Use Rapise?

Rapise was created to make software testing easy and manageable without being prohibitively expensive.

Rapise was made easy for software test professionals, developers and professionals concerned with quality assurance to simply and quickly write a test to cover an application, a web page, or a single bug to prevent regression.

Make Testing Fast and Repeatable

Consider for a moment what it is you do to test your software today. Most likely it has some for of user interface (UI), probably a graphic user interface (GUI). So you will run the application, click

around, perhaps in some way that gives you complete coverage of all the features (but probably not if it's a large application or web). Then you will login, if appropriate, and you will fetch some data and modify some data, test some more controls - edit boxes, buttons, drop-down lists, links, etc. If you have just fixed a bug then you will focus on the area of the application where the bug occurred. You will enter data that causes the bug, or go through the control sequence that causes the bug.

Next time you come to fix a bug in this application, you will **do the same thing again**.. Once again, you will focus on the area where the bug was.

Rapise presents you with **two methods for capturing specific tests**, and it will keep the test as a solo test or as part of a suite of tests that help you to qualify the application for release or a more formal QA process. Rapise is designed to allow the developer or the test professional to add new tests quickly and so to build up a library of tests.

There are two methods for capturing tests:

- <u>Record</u> and <u>playback</u>. In this type of test creation, you turn on the recorder and perform the actions needed to execute the test. Each test is saved to its own directory. A test consists of a javascript <u>test script</u> (.js), a meta-data file (*.sstest), and any number of additional supplementary scripts and data files. The test script is automatically generated after recording; simple modifications are required to make the test <u>data driven</u>. <u>Checkpoints</u> can be added during recording, or manually into the script.
- 2. Object-driven learning. Rapise considers each item on the page or within the application window to be an object. Examples are buttons, edit boxes, links, etc. To create a test using this technique, you have Rapise "learn" each control, and it will keep a miniature database of all the controls you "teach" it. To create a test, you write a script to instruct Rapise to perform a particular action on each object in the prescribed order. As any point along the way, the script you write can instruct Rapise to look inside an object and read its data and compare that value or content with what you expect it to be.

There are many methodologies with their own recommended approaches for designing testing strategies to ensure that application coverage is complete and meets the business requirements specification of the work being accomplished. Inflectra in general, recommends that you create a new test for each software requirement (to track progress) and for each issue in your issue tracking system (to test for regressions).

Choice of Scripting or Scriptless Rapise Visual Language (RVL)

Rapise gives you the choice of recording/writing your tests in a full <u>scripting language</u> - <u>JavaScript</u>, or the <u>Rapise Visual Language (RVL</u>) which provides a completely scriptless approach to writing tests (based on a simple table-based format). The RVL option is simpler for users that are not programmers but it has less flexibility than using JavaScript.

You can have the best of both worlds, putting the more sophisticated steps into JavaScript <u>scenarios</u> and then including in the main RVL file.

Integration with Test Management

To help you manage the requirements and issue tracking processes and to ensure that you have adequate **test coverage**, Inflectra recommend that you use Rapise with a **test management system** such as <u>SpiraTest</u>. That way you can maintain all your requirements, test cases and defects in a single place.

Once you have created the test, you can <u>playback</u> your test from within Rapise, run it from the

<u>command-line</u> or execute it remotely using RapiseLauncher in conjunction with <u>SpiraTest</u>. A <u>report</u> detailing the outcome of each step of the test will be automatically generated.

<u>Recording</u>, <u>playback</u>, the <u>report</u>, and the Rapise <u>engine</u> are all customizable.

2.2.2 Samples Index

Rapise includes several sample tests that you are free to read, modify, copy and use. They are located in: *RapiseDataDirectory*\Samples. Unless you specified otherwise, the *RapiseDataDirectory* will be:

C:\Users\Public\Documents\Rapise.

The sample tests are described below.

ActiveX

These samples demonstrate the testing of Microsoft ActiveX / COM controls used in Visual Basic applications including the MSComCtl library. The samples include the Microsoft FlexGrid Control, MS Common Toolbar Control, Microsoft Tabbed Dialog Control, TabStrip, and Microsoft Windows Common Controls 6.0 [MSCOMCTL.OCX].

AnalogRecorder

This sample demonstrates Analog Recording.

Dynamics AX

This sample demonstrates using Rapise to test an installation of <u>Microsoft Dynamics AX</u> 2013 to perform some basic ERP functions.

Dynamics CRM

This sample demonstrates using Rapise to test an installation of <u>Microsoft Dynamics CRM</u> to perform some basic contact management functions.

FarPoint

This sample script demonstrates using the FarPoint library to test the FarPoint SpreadSheet Control.

HTML5

This sample tests a HTML5 application. This sample demonstrates the capabilities of the **HTML5** DOM browser library. The application under test contains various HTML5 specific controls, such as: color, date, datetime, email, range, progress, etc.

The sample is also available online at http://www.libraryinformationsystem.org/Html5/AUTHTML5.htm

Java

This sample tests a Java AWT/SWING application. This sample demonstrates the capabilities of the **Java** library. The application under test contains various standard GUI controls, such as: button, edit,

tree, combo box, menu, etc.

Java SWT

This sample tests a Java SWT/RCP application. This sample demonstrates the capabilities of the **SWT** and **UIAutomation** libraries. The application under test contains various standard GUI controls, such as: button, edit, tree, combo box, menu, etc.

jQuery-Ul

This sample illustrates using the jQuery HTML DOM extension library that allows you to record/ playback test scripts against web applications using widgets from the jQuery Javascript library framework.

Library Information System

These tests can be used to test the sample library information system web application hosted at http:// www.libraryinformationsystem.net. This is the same sample application used by <u>SpiraTest</u> and illustrates how you can use <u>SpiraTest</u> to manage and execute automated Rapise tests. A copy of these tests is also available in new installations of <u>SpiraTest</u> v3.2+.

Managed

This sample tests a .NET 2.0 application. This sample demonstrates the capabilities of the **Managed** library. The application under test contains various standard GUI controls, such as: button, edit, tree, combo box, grid, listbox, listview, menu, etc.

QtFramework

TThis sample tests a sample <u>QT Framework</u> cross-platform application. This sample demonstrates the capabilities of the QtFramework library. The application under test contains various standard Qt widgets, such as: button, edit, tree, combo box, etc.

Silverlight

This sample tests a Silverlight web application. This sample demonstrates the capabilities of the **UIAutomation** library. The application under test contains various standard GUI controls, such as: button, edit, tree, combo box, menu, etc.

SimulatedObject

This sample opens **MS Paint** and draws on its canvas. It uses <u>Simulated Objects</u> and several related methods: **DoMouseMove**(X,Y), **DoLButtonDown**(), **DoLButtonUp**(), and **DoSendKeys**(text).

SampleATM

This sample tests an **MFC** application. You will also learn how to organize your test script in modular form, how to launch the AUT from your test script and how to execute various actions on GUI controls.

UsingCustomStrings

This sample demonstrates how to integrate Rapise tests with other tools using <u>Custom Strings</u>. **TestFinish()** is used to analyze and save test results. For more details, see: <u>Custom Strings</u>.

UsingDatabase

This example shows how you can use a relational (SQL) database to create <u>Data-Driven</u> tests. This script reads test case data from a spreadsheet ADO datasource to test Calculator.

UsingDLLHandlerManaged

This sample shows how to <u>unit test managed DLLs</u>. You'll see how to use methods **CreateClassInstance**() and **InvokeMember**().

UsingDLLHandlerUnManaged

This sample shows how to <u>unit test unmanaged DLL code</u>. You'll learn how to register (UserWrap.Register) and execute (UserWrap.ShellExecute) a function.

UsingImageCheckPoint

This example shows how to create image checkpoints.

UsingInclude

This sample demonstrates two ways to include external files/functions:

- 1. eval(g_helper.Include(...)): include a file with utility functions.
- 2. SeSRunJSScript(...): include and execute external function with its own object map.

UsingMSAccess, UsingMSExcel, UsingMSWord

These samples demonstrate how you can work with **Microsoft Word**, **Excel**, and **Access** from scripts. You'll learn how to test applications that expose a <u>COM interface</u>.

UsingMobile

These samples demonstrate how to do the testing of mobile devices running either Apple iOS or Android.

UsingOCR

This sample demonstrates usage of the Optical Character Recognition (OCR) functionality.

UsingRegistry

This sample demonstrates usage of the windows registry. The registry is queried to determine the OS (XP/2003/Vista, etc) and owner.

UsingReporting

This sample illustrates various reporting features:

- 1. Regular reporting (Tester.Assert, Tester.Message)
- 2. Custom attributes (Tester.SetReportAttribute, Tester.ResetReportAttribute)
- 3. Stacked attributes (Tester.PushReportAttribute, Tester.PopReportAttribute)
- 4. Nested Tests (Tester.BeginTest, Tester.EndTest)
- Inserting Links, Text and Images into the report (tags parameter, SeSReportText, SeSReportLink, SeSReportImage)

UsingSelenium

This sample how you can use Rapise to write <u>Selenium WebDriver</u> based web application tests using the raw Selenium WebDriver API.

UsingSpreadSheet

This example shows how you can use **Excel** spreadsheets to create <u>Data-Driven</u> tests. This script reads test case data from an XLS spreadsheet to test **Calculator**.

UsingXML

This sample demonstrates how to read, modify and write XML files.

WebServicesREST

This sample demonstrates how you can use the Rapise <u>Web-Services</u> module to write and execute automated web service tests against an HTTP <u>RESTful web service</u>.

WebServicesSOAP

This sample demonstrates how you can use the Rapise <u>Web-Services</u> module to write and execute automated web service tests against an <u>SOAP web service</u>.

2.2.3 Tutorial: Web Testing

In this section, you will learn how to record and execute a Rapise script against a web application. We will be using a demo application called **Library Information System**. Our test will be simple. It will log on to the library catalog, navigate to the main menu, and click on all of the menu options to make sure the links are working.

1. Open Rapise and Create New Test

Go to Start > All Programs > Inflectra > Rapise. The following window should appear.

File Test Oddions REST	Page 🗵 Spira Dashboard 🛛 🧕	💽 . 🕟 No Debugging - 🕫 About 🖌 Convert to NeoLoad 🙌 🛷	
Save to Spira 🕅 Test	Manual Steps Reco		
Gopen Test from Spira	tions 👍 Web Services	Spy La Object Mor Files	
Eté	Shortcuts Rec	old and Execute Debugging Tools Help.	
Objects	Start Page 31 LIS-E11_2016-12	-21_16-45 try 🗿 US mot	+ x
V * * R			~
Object Tree C-[Ubers],edem sendman(Docu J Inflectre Library Information System - Int	SpiraTeam	Current Test	Documentation 2
Author_{Author} Author_Management (Author Manage	Open Spira Dashboard	Name: LIS/E11 Test Description	+ Rapise Quick Start Guide
A Book, Management (Book Manageme ColobMainContexctbinSubmet (ct008 A Home (Home) D Ibbraine (Ibbrainet)	Use the dashboard to link texts to test cases, create and run test sets, vew automation hosts and more.	Scenario: Test Y Play User Functions: USER 11. John Play	Proparing Fredux for Testing Preparing Chrome for Testing Preparing for I/OS Moote Testing Preparing for Andioid Testing
A Log_in [Log in] Mane_[Narie]	Samples	Script: US-IE11.p	Using Selenium with Rapise
Library Information System Log In - Inter Library Information System Book Manag	Fetch Samples	Parameters Spira Properties	Shortcuts
IIS Global Deer Functions User Variables.	AdobeFitx3	Browser: Internet Digsner HTML V Record Tille reges. *	Create New Test Copen Existing Test Copen Start Recording Start Recording
	CreateNewAuthor		Online Resources
4 F	DynamicsAX DynamicsCRM		Rappe Home Page
Objects Files REST Settings	EdtExtingAutor	Start Here Your Rapise Tutorial snow:	Documentation
Properties	EditExistingBook		Forum Help Desk
22 24 III	EarPoint	Recent Tests	- Luch Pank
	DITTMLS	C./Usersiadam.sandman/Documents/My Rappie Tests/US-IE11/US-IE11.ssteat	
	D Java D JavaSWT	C.:Subversion/Projects/SpiraTeam.Add.One:Trunk/Version.Control Tools/TaravautProvider/Baprie_Test/ProjectLocker/ProjectLocker.sstest	~
	Response Body		
	Auto Raw XML ISON		
			î
			*
	<	Output Warrings Entry FindPassats	}
	NUM	and a second second second second	Inflectva Corporation

Click on the File tab in main menu and then click on the option to 'Create New Test':

€	
Recent Tests	
New Test Create a new Test	

If this is your first time using Rapise on this computer, you may see the following dialog box:

Connect To Spira Server	
URL:	http://localhost/SpiraTest
User Name:	
Password:	
Default Repository Path:	
Token (read only):	
Test	Save Cancel

If you see this, it means that Rapise is trying to <u>connect to a SpiraTest server</u>. **SpiraTest** is our web based <u>test management system</u>. It is a powerful tool that can store your Rapise tests and deploy them onto remote machines for automated regression testing. However, for now just click on the **[Cancel]**

button and you will see the 'new test dialog':

Create New Test	×
Please specify the name of the new test:	
Web Testing 1	
Specify the folder where this test should be created:	
C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\Web Testing 1\Web Testing 1.sstest	
Rapise supports multiple testing methodologies. Select one to be used for newly created test.	
Basic: Standard Scripting Mode	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
Cancel	

Now enter the name of your new test '**Web Testing 1**', make sure the Methodology is set to '**Web: Cross-Browser Testing Support**' and click **[Create]**. Since you chose a web test, you will now need to choose the initial <u>web browser profile</u> (don't worry you can easily change it later):

Specify Web Profile:	Internet Explorer HTML	
		OK

Choose 'Internet Explorer HTML' from the list of options.

Next you will be asked if you want to create your tests using the scriptless **Rapise Visual Language** (**RVL**) technology or using JavaScript. For this example we will use the RVL scriptless approach. If you're interested in creating the test using JavaScript instead, please refer to the section - <u>Using JavaScript Scripting</u>.

R Scripting Language	_		×
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Script	ing Language	e)	

Rapise will create the new test and you will see the empty recording grid:

RVL \ +Add New									
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue	Н	
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value		
2 🕨		×	/						
3									
4				×	/				
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									

You are now ready to record your first test.

2. Open the AUT (Application Under Test)

Open up Internet Explorer. You will find it in **Start** > **All Programs** > **Internet Explorer**. In Internet Explorer, navigate to: <u>http://www.libraryinformationsystem.org</u>:



3. The Select an Application to Record Dialog

In the Rapise window, press the Record/Learn button on the Ribbon.



The Recording Activity Dialog (RA dialog) will appear:

Rec	cording Act	ivity for "Inte	ernet Expl	orer HTML"			
#	Object	Action	Data	Comment			
	Verify (Ctrl+1)	Learn (Ctrl+2))	SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel
Read	dy					Advanced>>	Transparent

The **RA dialog** has a grid. As you interact with the sample Library Information System program, the grid will automatically populate with your actions.

4. Recording

Let's begin creating the test. On the library information system login page, click on the **Log In** link in the top-right of the screen.

LIBRA	RY INFORMAT	ON SYSTEM	ا ملوما ا
Home	Book Management	Author Management	
Accou	er your username and passw nt Information ame:	urd:	

In the Username text box, type librarian

Click on the **Password** text box next. You'll notice that the **RA dialog** has changed. Your actions, clicking Log-In and entering a username, are listed in the grid:

# 沙1	Object Log In	Action Click	Data	Comment Click on Log	In		
⊎2	Username:	Set	librarian	Set Text libra	rian in Username:		
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel

The password for user librarian is also *librarian*. Type the password in and then press the **Log-In** button.

Two more rows should appear in the **RA dialog**: one to represent the password entry, and one to represent the button click:

#	Object	Action	Data	Comment			
1	Log In	Click		Click on Log I	n		
12	Username:	Set	librarian	Set Text librar	rian in Username:		
13	Password:	Set	librarian	Set Text librar	rian in Password:		
₩4	ctl00\$Mai	Click		Click on ctl00	\$MainContent\$LoginUs	ser\$LoginButton	
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel

You should now be on the main menu of the Library Information System with the user's name listed in the top-right:

LIBRA	RY INFORMAT	ION SYSTEM		Welcome librarian! [Log Out]
Home	Book Management	Author Managemen		
WELCOM	ME TO THE LIBRARY	INFORMATION SY	STEM	
This sample	application lets you view.	create and edit books in	the libary catalog as well as view, create and edit authors.	
	library catalog or the auti d to login as a librarian.	nors list you will need to	login as a borrower and to make changes to the list of books or au	thors
	s not a real application, bu ation system. Both of these		tion used in the popular SpiraTest <u>test management system</u> and I by <u>Inflectra Corporation</u> .	tapise
spir	Tesť	Rapise	inflectra	
Test Mana	gement 1	fest Automation	Inflectra Corporation	
This sample	application has <u>SOAP</u> and	d <u>REST</u> web service APIs t	hat can be tested by Rapise.	

Hover the mouse over the "Welcome **librarian**" username label on the top-right and click CTRL+1 to bring up the Verify dialog box:

Verify	Object Propertie	25	
#	Name 🔺	Value	•
	Bitmap	librarian	
	BWBitmap	librarian	
	Class		
	Enabled	true	
	Height	19	
	ld	HeadLoginView_HeadLoginName	
✓	InnerText	librarian	
	NodeText	librarian	
	ObjectType	HTMLObject	
	PageTitle	Inflectra Library Information	
	PageURL	http://www.libraryinformationsyste	
	State	1048580	
	Style		•
	ОК	Cancel	

This box lets you add a checkpoint to verify the properties of an object on the screen.

Select the "Inner Text" option and click the checkbox. That will add the verification check to your list of recorded actions:

#	Object	Action	Data	Comment			
10	LogIn	Click		Click on Log	In		
32	Usemame:	Soffext	librarian	Set Text libra	rian in Username:		
33	Password:	Soffext	librarian	Set Text libra	rian in Password:		
<u>84</u>	ctl00\$Mai	Click		Click on cil00	\$MainContent\$Logir	iUser\$LoginBuiton	
ø <mark>5</mark>	librarian	Verify	librarian	Verify that: In	nerText=librarian		
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel
aused						Advanced>>	Transparer

Click the **Book Management** button. It is highlighted in the next screenshot:

LIBRA	RY INFORMAT	ION SYSTEM
Home	Book Management	Author Management

You should now be on the Book Management page (see the below image). Click the Home button to go back to the main menu.

	BRARY INFORM	ATION SYSTE	IM I			Welcome librarian! [Log
Но	me Book Managemer	nt Author Manag	ement			
Bo	ok Management					
	following books exist in the s			1.000		
ID	Name	Author	Genre	Edit		
1	Hound of the Baskervilles	Arthur Conan Doyle	Murder & Mystery	Edit		
2	The Scowrers	Arthur Conan Doyle	Murder & Mystery	Edit		
3	Amsterdam	Ian McEwan	Contemporary Fiction	Edit		
4	Saturday	Ian McEwan	Contemporary Fiction	Edit		
5	The Comfort of Strangers	Ian McEwan	Contemporary Fiction	Edit		
6	Chesil Beach	Ian McEwan	Contemporary Fiction	Edit		
7	Atonement	Ian McEwan	Historical Fiction	Edit		
8	Bleak House	Charles Dickens	Historical Fiction	Edit		
9	Oliver Twist	Charles Dickens	Historical Fiction	Edit		
10	Nicholas Nickleby	Charles Dickens	Historical Fiction	Edit		
11	David Copperfield	Charles Dickens	Historical Fiction	Edit		
12	The Pickwick Papers	Charles Dickens	Historical Fiction	Edit		
13	Death on the Nile	Agatha Christie	Murder & Mystery	Edit		
14	Betrams Hotel	Agatha Christie	Murder & Mystery	Edit		

Click the Create new book link:

BOOK MANAGEMENT

The following books exist in the system (Create new book)

You should now be on the Create New Book page (see image below). Click the **HOME** button to go back to the main menu.

LIBRARY INFORMATION SYSTEM	Welcome librarian! [Log Out]
Home Book Management Author Management	
CREATE NEW BOOK Please enter the book information and click Insert: Book Information Name:	
Author: [Ian McEwan V Genre: Science Fiction V	

Now, click the Author Management button:

LIBRA	RY INFORMAT	ION SYSTEM
Home	Book Management	Author Management

You should now be on the Author Management page (see image below):

LIBRARY INFORMATION SYSTEM

Book Management Author Management

Welcome librarian! [Log Out]

AUTHOR MANAGEMENT

The following authors exist in the system: (Create new author)

ID	Name	Age	Edit
1	Ian McEwan	42	Edit
2	Charles Dickens	105	Edit
3	Arthur Conan Doyle	125	Edit
4	Agatha Christie	98	Edit

Click the Create New Author link:

AUTHOR MANAGEMENT

The following authors exist in the system (Create new author)

ID	Name	Age	Edit
1	Ian McEwan	42	<u>Edit</u>
2	Charles Dickens	105	<u>Edit</u>
3	Arthur Conan Doyle	125	<u>Edit</u>
4	Agatha Christie	98	Edit

You should now be on the Create New Author page (see below). Click the **Home** button to go back to the main menu.

LIBRARY INFOR	MATION SYSTEM	Welcome librarian! (<u>log Out</u>			
Home Book Manag	ement Author Management				
CREATE NEW AUTHO Please enter the author info	Change and an annual second				
Name:					
Insert Cancel					

At this point, there should be 11 rows in the RA dialog grid.

You are now back on the Main Menu. Click Log Out (top-right).



To end the recording session, you can either press CTRL+3 or press the Stop button on the Record dialog. End the recording session now.

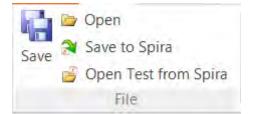
Rapise will ask you whether you want to use this recording or discard it:

🗔 Star	t Page 🗙	🔊 Spira Dashboard	× Web Testing 1.rvl.	xls ×				
	New recording is available. Please, choose how to use recorded steps:							
In	sert Here	Discard						
RVL	+Add New	\						
	Flow	Туре	Object	Action	ParamName	ParamType	Param	
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
2)		~						
3								

Click on the **Insert Here** button and Rapise will insert the recorded steps into the test grid:

RVL (+Add New						
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue H
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value
2 1	#	Click on Log I	ו				
3 🥖		Action	A Log_In	DoClick			
4 0	#	Set Text librar	ian in Username:				
5 /	,	Action	🕫 Username_	DoSetText	txt	string	librarian
6 /	#	Set Text librar	ian in Password:				
7 🧳		Action	abl Password_	DoSetText	txt	string	librarian
8 /	#	Click on ctl00	\$MainContent\$LoginUse	r\$LoginButton			
9 /	,	Action	■ ctl00\$MainConten	DoClick			
10 🥒	#	Verify that: In	nerText=librarian				
11 /		Assert			message	string	Verify that: InnerText=librarian
12 🧷	•	Action	🗆 librarian	GetInnerText			
13 🥒	,	Condition		output1 == param2			
14 🥒		Param			param2	string	librarian
15 🌶	#	Click on Book	Management				
16 🥖	,	Action	A Book_Management	DoClick			
17 🥒	#	Click on (Crea	te new book)				
18 🥖		Action	A _Create_new_boo	DoClick			
19 🌶	#	Click on Autho	or Management				
20 🧷	,	Action	Author_Managem	DoClick			
21 🥖	: #	Click on (Crea	te new author)				
22 🌶		Action	A _Create_new_aut	DoClick			
23 🌶	#	Click on Ho					
24 🧷	•	Action	A Home	DoClick			

Let's save our test. Press the Save button at the top left of the Rapise window.



5. Playback

Let's execute the test we just created. First, close Internet explorer. Rapise will open a new instance of Internet Explorer to the correct url (www.libraryinformationsystem.org) when the test begins.

To execute the script, press the Play button at the top middle of the Rapise window.



After execution, a screen like the one below will appear. Each row represents a step in the test. The rows with green text are steps which passed, whereas the rows with red text are the steps which failed.

2	Type	Start	-	Name	Status		Comment
* 10	10	-	80.		28	20	
	Message	22:49:48.222	Starting scenar	io: Test	Info		
	Assert	22:49:49.994	Log In.DoClick((1)	Pass	Returned Value:	true
	Assert	22:49:51,188	Username: DoS	letText(["librarian"])	Pass	Returned Value:	true
	Assert	22:49:52.263	Password: DoSe	etText(["librarian"])	Pass	Returned Value:	true
	Assert.	22:49:53.329	cti00\$MainCont	ent\$LoginUser\$LoginButton.DoClick([])	Pass	Returned Value:	true
	Assert	22:49:53.468	Verify that: Inn	erText=librarian	Pass		
	Assert	22:49:54.567	Book Managem	ent.DoClick([])	Pass	Returned Value:	true
	Assert	22:49:55.768	(Create new bo	ok).DoClick([])	Pass	Returned Value:	true
	Assert	22:49:57.000	Author Manage	ment.DoClick([])	Puss	Returned Value:	trué
	Assert	22:49:58.218	(Create new au	thor).DoClick([])	Pass-	Returned Value:	true
	Assert	22:49:59,472	Home.DoClick()		Pass	Returned Value:	true
	Assert	22:50:00.673	Log Out.DoClid	(1)	Pass	Returned Value:	true
	Test	22:50:00.673	Web Testing 1		Pass	Passed:11 Failed:	0 Time: 12.567s

For more information on the report, see Automated Reporting.

Playback in Other Browsers

Now that we have recorded our test in Internet Explorer, we want to play the same script back in other browsers. That is very easy to do. Simply click on the shortcut for the Rapise <u>Start Page</u>, and then change the **web browser dropdown** to a different browser (e.g. Firefox, Chrome, Selenium, etc.) and click the **Play** button.

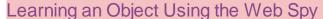
Current Test		
Name:	Scenarios Test	Test Description
Scenario:	Test 🗸 Play	
User Functions: Script:	<u>Scenarios Test.user.js</u> <u>Scenarios Test.js</u>	
Parameters Spira Properties		
Browser:	Selenium - Chrome	\checkmark
Record Title:	Chrome HTML Firefox HTML Internet Explorer HTML	
	Selenium - Chrome	
	Selenium - Firefox	
	Selenium - Internet Explorer	
🕫 🤍 🗐	Selenium - Opera Selenium - Safari	

Before the playback in other browsers will work correctly, you will need to make sure you have <u>configured the web browsers</u> appropriately. In the case of the Selenium options, you will need to make sure you have installed the <u>Selenium WebDriver libraries</u> correctly.

The next section will demonstrate how you can use Rapise to inspect the objects in a web page and

Learn them for testing. This is useful in cases where you have more complex applications to test and you need to pick specific objects. For example you may want to select one of the books in the grid based on its name rather than its row number (which may change if you add books).

Using the Spy Tools in Rapise



In the main Test ribbon of Rapise, expand the dropdown list for the **'Spy**' tool and make sure that 'Web Object' is selected:



Now, click on the main 'Spy' tool icon and Rapise will start the Web Spy:

a ≠ Sov			Rapise Spy				•	-	×
Save Track (Ctrl-T)	*	90			Test XPat	h Learn	1		
Get Snapshot Highlighting Mode	Learn	Options			Test CSS	Learn			
(Ctrl-G) Browser	Object	Tools	Locators						~
<i>₿</i> IE									
DOM Tree			DOM Element						
				Record	ding Activity for	Teflectra I I	ibeans Inf	omation	p Surta
				#	Object	Action	Data		Comm
			Test Results	#	object	Action	Dara		Comm

Go back to the web page and login to the **library information system** with the same **login/password** (**librarian/librarian**) and click on the 'Book Management' menu item so that the list of books is displayed:

Но	me Book Managemer	nt	Author Manag	ement	
	OK MANAGEMENT	system	: (Create new boo	<u>ok)</u>	
ID	Name	Auti	hor	Genre	Edi
1	Hound of the Baskervilles	Arth	ur Conan Doyle	Murder & Mystery	Edi
2	The Scowrers	Arth	ur Conan Doyle	Murder & Mystery	Edi
3	Amsterdam	Ian McEwan		Contemporary Fiction	Edi
4	Saturday	Ian McEwan		Contemporary Fiction	Edi
5	The Comfort of Strangers	Ian M	VicEwan	Contemporary Fiction	Edi
6	Chesil Beach	Ian M	VicEwan	Contemporary Fiction	Edi
7	Atonement	Ian M	VicEwan	Historical Fiction	Edi
8	Bleak House	Char	rles Dickens	Historical Fiction	Edi
9	Oliver Twist	Char	rles Dickens	Historical Fiction	Edi
10	Nicholas Nickleby	Char	rles Dickens	Historical Fiction	Edi
11	David Copperfield	Char	rles Dickens	Historical Fiction	Edi
12	The Pickwick Papers	Char	rles Dickens	Historical Fiction	Edi
13	Death on the Nile	Agat	tha Christie	Murder & Mystery	Edi
14	Betrams Hotel	Agat	tha Christie	Murder & Mystery	Ed

Now back in the <u>Web Spy</u>, click on the 'Get Snapshot' option to refresh the Web Spy and display the HTML elements (called the DOM tree) that make up this page:

	Rapise Spy				-	×
Spy Get Snapshot (Ctrl-G) Browser (Ctrl-G) Browser Ctrl-G) Browser Ctrl-G	Locators		Test XP Test CS			~
CON Tree	DOM Element					
<pre>4 <html:kml:lang="en".xmlns="http: 1999="" www.w3.org="" xhtml"=""></html:kml:lang="en".xmlns="http:></pre>	style Properties height width x	1 0 data Main all			und-color: white	в;
<pre></pre>	y Selectors css xpath Test Results	Recordir #	ng Activity for Object	"Inflectra Li	brary Informa Data	tion System Comment

Once it has loaded the DOM tree, you can expand/collapse the elements to see how the web page is constructed. This is useful when testing an application since many of the HTML elements on a page may be used for layout purposes and will not be visible in the browser. In the example page, we have expanded some of the nodes to display the main section of the page and the table that contains the list of books.

In addition, you can use the **Track (Ctrl+T)** tool to select and item in the web page and then have it be highlighted in the DOM tree. For example if we want to find the cell that contains the book title

"Amsterdam", simply click CTRL+T on the keyboard, move the mouse over the cell in the webpage, wait until the red highlighting rectangle appears and then click CTRL+T again. Rapise will now highlight that item in the DOM Tree automatically:

<u>A</u> ≠	Rapise Spy	🔁 – 🗖 🗙
Spy Track (Ctrl-T) Get Snapshot (Ctrl-G) Browser (Ctrl-G) Browser (Ctrl-G) Cbject Tools	Locators	Test XPath Learn Test CSS Learn
<i>₿</i> IE		
DOM Tree	DOM Element	
dtml xml/ang="en" xmlns="http://www.w3.org/1999/xhtml"> dead> dead>	tagName Properties height x y Selectors css have better the selectors css have better the selectors have better the sel	Imsterdam TD 12 64 52 116 tml > body > form#ctl01 > div.rith-of-type(3) > div.rith-of-type(2) html/body/form[@id='ctl01']/div[3]/div[2]/div.Aable[@id='Main

You can see all of the properties of this HTML element displayed on the right, specifically:

- The tagName is displayed as 'TD' (always upper case)
- The innerHTML of the element is displayed (Amsterdam)
- The CSS and XPath for locating this element is displayed

If you want to use this object in a Rapise test script, you can simply click the **Learn** button and the HTML element will be added to the Recording Activity Dialog:

Reco	Recording Activity for "Internet Explorer HTML"							
#	Object	Action	Data	Comment				
圆1	Amsterdam	Learn	Amsterda	Learned Amste	erdam			
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel	
Last c	aptured: HTML(Object (h	ttp://www.libr	aryinformationsy.		Advanced>>	Transparent	

Click Finish and the object will have been added to your test's Object Tree:

Object Tree
id n id[]
Object Tree C:\Users\adam.sandman\Documents
Inflectra Library Information System - Intern
Library Information System Log In - Internet
Library Information System Book Managem
ACreate_new_book [(Create new boo
Amsterdam [Amsterdam]
Library Information System Author Manage
Global
🚯 User Functions
var User Variables

To use this new object in our test script, we can simply use the test editor to add the appropriate command. For example, if you wanted to get the textual value of the cell in your test, you should click on the row in the grid after the **Book Management – DoClick**:

RVL	+Add New	\						
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue	F
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
2	#	Click on Log	In					
3 /		Action	A Log_In	DoClick				
4 1	#	Set Text libra	arian in Username:					
5 /		Action	🐱 Username_	DoSetText	txt	string	librarian	
6 /	#	Set Text libra	arian in Password:					
7 🌙		Action	💩 Password_	DoSetText	txt	string	librarian	
8 /	#	Click on ctl0	0\$MainContent\$LoginUse	r\$LoginButton				
9 /		Action	ab ctl00\$MainConten	DoClick				
10 🌙	#	Verify that: 1	nnerText=librarian					
11 🌶		Assert			message	string	Verify that: InnerText=	librarian
12 🌙		Action	🗆 librarian	GetInnerText				
13 🌶		Condition		output1 == param2				
14 /		Param			param2	string	librarian	
15 🌶	#	Click on Bool	k Management					
16 🌶		Action	A Book_Management	DoClick				
17 1	#	Click on (Cre	ate new book)					
18 /		Action	A _Create_new_boo	DoClick				
19 4	#	Click on Auth	nor Management					

Now click on the Ins Row button in the main Test ribbon to add a new row:



This will insert a new row into the test. In this new row, right-click on each of the cells (as illustrated below) and pick the following values from the dropdown lists:

14 /	Pa	ram			
15 🥒 #	Cli	ck on Book	Management		
16 🥒 Action		tion	A Book_Management	DoClick	
17 🕨		~	·		
18 #	٢		e new book)		
19 🥒	4	Action	A _Create_new_boo	DoClick	
20 🥒 #	C	Param	⁻ Management		
21 🧳	4	Output	Author_Managem	DoClick	
22 🧳 #	C	Variable	e new author)		
23 🧳	4	Assert	A _Create_new_aut	DoClick	
24 🥒 #	C	Condition			

Then choose the following:

- **Type** = Action
- **Object** = Amsterdam
- Action = GetInnerText

You should now have the following:

14		Param		
1.5	#		ook Management	
1.6		Action	<u>A Book, Management</u>	
17		Action	🗖 Amsterdam	GetInnerText
1.8	#	Click on (C	Create new book)	
1.9		Action	A Create new boo	
	- II	<u> </u>	. 1	Ý

2.2. Learning an Object from XPATH

In addition to letting Rapise automatically learn the object from the Web Spy, you can manually enter in XPATH or CSS queries to find matching elements on the page and then learn those for use in your test.

For example, suppose we want to dynamically find the row that has the cell containing Amsterdam and then click on its Edit hyperlink.

Open up the Web Spy as before:

<u> </u>	Rapise Spy			· -	– ×	
Spy						
Get Snapshot Highlighting Mode			Test XPath Learn]		
(Ctrl-G)				<u>'</u>]		
Browser Object Tools	Lo	cators			~	
🚝 IE						
DOM Tree	DOM Element					
A <html xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"></html>	4					
head>	tagName	TABLE				
	⊿ Attributes					
	border	1				
<pre></pre>	cellspacing	0				
div class="aspNetHidden">	class	class dataGrid				
<script type="text/javascript"></script>	id	MainContent_grdBooks				
<script aspnethidden"="" src="///ebResource.axd?d=9_D5u0u-Rawwl" type="text/javas</td><td>ci rules</td><td colspan=5>all</td></tr><tr><td><div class="></div></td><td>style</td><td colspan=5>border-collapse: collapse; background-color: white;</td></tr><tr><td><pre>⊿ <div class="page"></pre></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td><div class="header"></div></td><td>height</td><td>331</td><td></td><td></td><td></td></tr><tr><td><pre>// <div class="main"></pre></td><td>width</td><td>500</td><td></td><td></td><td></td></tr><tr><td><h2></h2></td><td>x</td><td>225</td><td></td><td></td><td></td></tr><tr><td>4 <0></td><td>У</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>⊿ Selectors</td><td>Recording Act</td><td>tivity for "Inflectra Li</td><td>brary Informa</td><td>tion System</td></tr><tr><td></td><td>CSS</td><td># Ot</td><td>piect Action</td><td>Data</td><td>Commer</td></tr><tr><td>⊿ <div></td><td>xpath</td><td># 00</td><td>oject Action</td><td>Data</td><td>Commer</td></tr><tr><td>(table class="dataGrid" id="MainContent_grdBooks" style="</td><td>Test Results</td><td></td><td></td><td></td><td></td></tr><tr><td>▶</td><td>Test Results</td><td></td><td></td><td></td><td></td></tr><tr><td><div class="clear"></div></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td><div class="footer"></div></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td><script type="text/javascript"></script>						

In the **Locators** section of the Web Spy, enter in the following to locate the table:

//table[@id='MainContent_grdBooks']

Now click on the Test XPath button to display the matching results:

//table[@id='MainContent_grdBooks']	Test XPath L	earn
	Test CSS L	earn
Locators		

Test Results					
⊿ Lo	cator: //table[@id='MainContent_grdBooks'], found: 1				
	<table border="1" class="dataGrid" id="MainContent_grdBooks" n<="" style="border-collapse: collapse; backg" th=""></table>				

Now that we have matched the table, we need to add dynamic XPath to find any row that has the cell containing 'Amsterdam' and find the edit link. You can expand the table and see the rows and cells visually and that will help us create the XPATH:

//table[@id='MainContent_grdBooks']//tr/td[text()='Amsterdam']/../td[5]/a

This XPath consists of the following elements:

1. Finds the table with the specified ID

- 2. Find any row inside that table that contains a cell with the text 'Amsterdam'
- 3. For any matching cell, get its parent row and inside the fifth cell, get any hyperlink

In this case that will correctly locate the Edit link for the book 'Amsterdam'. Now that we have the correct item identified, click on the 'Learn' button to the right of the 'Test XPath' button. That will now learn a new object that corresponds to the 'Edit Amsterdam' object:

# 沙1	Object Edit	Action Learn	Comment Learned Edit			
	/erify Ctrl+1)	Learn (Ctrl+2)	SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel

Click Finish and the object will have been added to the Object Tree of the current test:

r ne						
Object Tree						
v id n id∏						
🧾 Object Tree C:\Users\adam.sandman\Docum 🔺						
Inflectra Library Information System - Int						
Library Information System Log In - Inter						
🔺 🛄 Library Information System Book Manag 🚽						
ACreate_new_book [(Create new						
Amsterdam [Amsterdam]						
A A Edit [Edit]						
=						
■Q DoAddSelection						
≓∳ DoAnalogPlay						
■ DoClick						

To click on this object, simply click on the row in the grid where you want this action to occur and choose "**Ins Row**" from the test ribbon:

5	#		ook Management		
6		Action	A Book Management		
7		Action	🗆 Amsterdam	GetInnerText	
8					
9	#		reate new book)		
0		Action	A Create new boo		
			*		

In this new row, right-click on each of the cells, and pick the following values from the dropdown lists:

- **Type** = Action
- **Object** = Edit
- Action = DoClick

So your test will now look like:

15 🥒 #	Click on Bo	ook Management	
16	Action	A Book_Management	DoClick
17	Action	🗆 Amsterdam	GetInnerText
18 🕨	Action	<u>A</u> Edit	DoClick
19 #	Click on (C	reate new book)	
20 🧳	Action	A _Create_new_boo	DoClick
21 🥒 #	Click on Au	ithor Management	
22 🖉	Action	Author_Managem	DoClick

Since clicking on the Edit link will take you to a different page than where the 'Create New Book' link is available, we need to add another row and add:

- **Type** = Action
- **Object** = Book_Management
- Action = DoClick

so that Rapise goes back to the main book list page before executing the Create New Book step. If we did not do this, the test would have failed.

This means the test will now look like:

15 🥒 #	Click on Bo	ook Management	
16	Action	A Book_Management	DoClick
17	Action	🗆 Amsterdam	GetInnerText
18	Action	A Edit	DoClick
19 🕨	Action	<u>A</u> Book_Management	DoClick
20 #	Click on (C	reate new book)	
21 🧷	Action	A _Create_new_boo	DoClick

Now the line:

Actio	on 🗉	⊐ Amsterdan	n Ge	tInnerT	ext

by itself does not do anything, it just gets the text.

So, to make the test more useful, we can use the **Assert** command to turn this into a test for the correct value. To make the change, simply delete this entire row using the **Del Row** option on the ribbon and add a new blank row in its place.

Now choose the Type as Assert and press ENTER on the keyboard. You will now have the following:

Assert		message	string	TBD
Param		paramit	string	TBD
Condition	param1 == param2			
Param		param2	string	TBD

This is a generic placeholder for any type of conditional test. We need to first change the value of the message "TBD" to something meaningful (e.g. "Check that the name matches"):

Assert					atches
Param		parami	string	TBD	
Condition	param1 == param2				
Param		param2	string	'IBD	

This is the message that will be displayed if the test passes correctly.

Now we need to replace the first parameter with the output from the result of the Amsterdam test. To do this, change the **Condition** on the second row from: **param1 == param2** to **output1 == param2**:

Assert:			message	string	Check that the name matches
Action	TBD	TBD			
Condition		output1 == param2 v			
Param			param2	string	CRU

Notice how Rapise automatically changed the second row from Param to Action.

Now in the TBD cell next to the Action type, choose the **Amsterdam** object name and the **GetInnerText** action from the dropdown lists. Finally change the **param2** string from TBD to the expected value "Amsterdam".

You will now have:

Asseri:			message	string	Check that the name matches
Action	🗆 Amsterdam	GetInnerText			
Condition		output1 == param2			
Param			param2	string	Amsterdam

Now click **Play** to playback the new test:

#	Туре	Start	-	Name	Status	Comment
	140	-	(A)		LAI	a
	Message	00:04:58.315	Starting scenario:	Test	Info	
	Assert	00:04:59.815	Log In.DoClick([])		Pass	Returned Value: true
	Assert	00:05:00.972	Username:.DoSet1	ext(["librarian"])	Pass	Returned Value: true
	Assert	00:05:02.019	Password:.DoSetT	ext(["librarian"])	Pass	Returned Value: true
	Assert	00:05:03.082	ctl00\$MainContent	\$LoginUser\$LoginButton.DoClick([])	Pass	Returned Value: true
	Assert	00:05:03.222	Verify that: InnerT	ext -librarian	Pass	
	Assert	00:05:04.316	Book Management	.DoClick([])	Pass	Returned Value: true
	Assert	00:05:04.472	Check that the na	me matches	Pass	
	Assert	00:05:05.801	Edlt.DoClick([])		Pass	Returned Value: true
	Assert	00:05:07.223	Book Management	.DoClick([])	Pass	Returned Value: true
	Assert	00:05:08.441	(Create new book)	.DoClick([])	Pass	Returned Value: true
	Assert	00:05:09.786	Author Manageme	nt.DoClick([])	Pass	Returned Value: true
	Assert	00:05:10.989	(Create new autho	r).DoClick([])	Pass	Returned Value: true
	Assert	00:05:12.301	Home.DoClick([])		Pass	Returned Value: true
	Assert	00:05:13.645	Log Out.DoClick([])	Pass	Returned Value: true
F.	Test	00:05:13.645	Web Testing 1		Pass	Passed: 14 Failed:0 Time: 15.423s

The test should now pass successfully.

2.2.3.1 Using JavaScript

This section will demonstrate how you can use Rapise to inspect the objects in a web page and Learn them for testing using the JavaScript test script language instead of the Rapise Visual Language (RVL) that was illustrated in the main <u>Web Testing Tutorial</u>.

This is useful in cases where you have more complex applications to test and you want to be able to use the power of a full programming language such as JavaScript to iterate over data, perform calculations and use if...then...else branches to follow different steps during the test.

Learning an Object Using the Web Spy

In the main Test ribbon of Rapise, expand the dropdown list for the '**Spy**' tool and make sure that 'Web Object' is selected:



Now, click on the main 'Spy' tool icon and Rapise will start the Web Spy:

A ≠			Rapise Spy			Ŧ	-		×
Sov S		. 00		Test XPat	h Learn	1			
Get Snapshot (Ctrl-G)	e Learn	Options		Test CSS	Learn				
Browser	Object		Locators			1			~
BIE DOM Tree			DOM Element						
			Reco	rding Activity for	"Inflectra L	ibrary I	nforma	ation Sy	ste
			Test Results #	Object	Action	Data		Co	mm

Go back to the web page and login to the **library information system** with the same **login/password** (**librarian/librarian**) and click on the 'Book Management' menu item so that the list of books is displayed:

LIBRARY INFORMATION SYSTEM

Но	me	Book Managemen	ıt	Author Manag	ement		
		ANAGEMENT	ystem	: (Create new boo	<u>ok)</u>		
ID	Name	2	Aut	ıor	Genre		Edit
1	Houn	d of the Baskervilles	Arth	ur Conan Doyle	Murder	& Mystery	Edit
2	The S	cowrers	Arth	ur Conan Doyle	Murder	& Mystery	Edit
3	Amste	erdam	Ian I	AcEwan	Contem	porary Fiction	Edit
4	Sature	day	Ian I	AcEwan	Contem	porary Fiction	Edit
5	The C	omfort of Strangers	Ian I	AcEwan	Contem	porary Fiction	Edit
6	Chesi	Beach	Ian I	AcEwan	Contem	porary Fiction	Edit
7	Atone	ement	Ian I	AcEwan	Historica	al Fiction	Edit
8	Bleak	House	Cha	les Dickens	Historica	al Fiction	Edit
9	Oliver	Twist	Cha	les Dickens	Historica	al Fiction	Edit
10	Nicho	las Nickleby	Cha	les Dickens	Historica	al Fiction	Edit
11	David	Copperfield	Cha	les Dickens	Historica	al Fiction	Edit
12	The P	ickwick Papers	Cha	les Dickens	Historica	al Fiction	Edit
13	Death	on the Nile	Aga	ha Christie	Murder	& Mystery	Edit
14	Betrar	ns Hotel	Aga	ha Christie	Murder	& Mystery	Edit

Now back in the <u>Web Spy</u>, click on the 'Get Snapshot' option to refresh the Web Spy and display the HTML elements (called the DOM tree) that make up this page:

<mark>2</mark> ∓ R Spy	apise Spy				T -	×
Get Snapshot (Ctrl-G) Browser	Loca	tors	Test XF Test CS			^
(B) IE DOM Tree	DOM Element					
<pre>4 chtml xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"></pre>	a tagName tagName border cellspacing class id rules style a Properies height width x y	1 O dat Mai all)		und-color: whit	e;
4 < div> (table class="dataGrid" id="MainContent_grdBooks" style="bit_books" s	Selectors css xpath Test Results	Recordi	ng Activity for Object	"Inflectra Lil	Data	tion System - Comment

Once it has loaded the DOM tree, you can expand/collapse the elements to see how the web page is constructed. This is useful when testing an application since many of the HTML elements on a page may be used for layout purposes and will not be visible in the browser. In the example page, we have expanded some of the nodes to display the main section of the page and the table that contains the list of books.

In addition, you can use the Track (Ctrl+T) tool to select and item in the web page and then have it be

highlighted in the DOM tree. For example if we want to find the cell that contains the book title "Amsterdam", simply click CTRL+T on the keyboard, move the mouse over the cell in the webpage, wait until the red highlighting rectangle appears and then click CTRL+T again. Rapise will now highlight that item in the DOM Tree automatically:

<u>₽</u> • R	Rapise Spy
Spy Get Snapshot (Ctrl-G) Browser	Test XPath Learn Test CSS Learn Locators
C IE DOM Tree	DOM Element
DOM Tree a <html lang="en" xml="" xmlns="http://www.w3.org/1999/xhtml"> b <htead></htead> a <dody> a <form action="Books.aspx" id="ctl01" method="post"> b <diex.clsss="aspnethilden"> cscript type="text/javascript"> cdiv class="page"> cdiv class="main" cdiv cscript class="teader"></diex.clsss="aspnethilden"></form></dody></html>	UDUM Element innerHTML Amsterdam tagName TD Properties
 > dr> > dr style="background-color. whitesmoke;"> <td></td>	

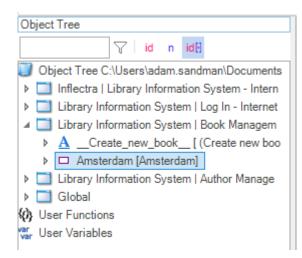
You can see all of the properties of this HTML element displayed on the right, specifically:

- The tagName is displayed as 'TD' (always upper case)
- The innerHTML of the element is displayed (Amsterdam)
- The CSS and XPath for locating this element is displayed

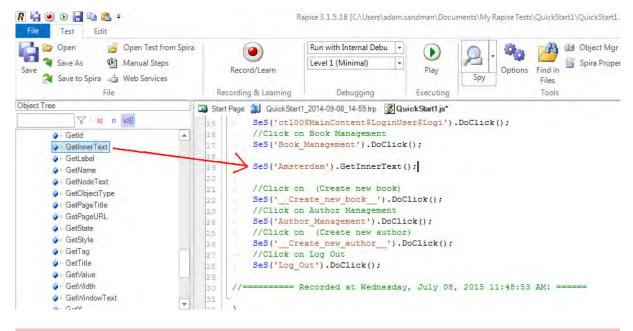
If you want to use this object in a Rapise test script, you can simply click the **Learn** button and the HTML element will be added to the Recording Activity Dialog:

#	Object	Action	Data	Comment			
尚1	Amsterdam	Learn	Amsterda	Learned Amste	erdam		
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel

Click Finish and the object will have been added to your test's Object Tree:



You can now expand this object and drag a test function to your test script. For example if you wanted to get the textual value of the cell in your test, drag the "GetInnerText" function into your test script:



Learning an Object from XPATH

In addition to letting Rapise automatically learn the object from the Web Spy, you can manually enter in XPATH or CSS queries to find matching elements on the page and then learn those for use in your test.

For example, suppose we want to dynamically find the row that has the cell containing Amsterdam and then click on its Edit hyperlink.

Open up the Web Spy as before:

_													
, <u>2</u> =					Rapise Spy						· -	•	×
Spy													
~	Track (Ctrl-T)	-	- Carlor (1997)						Test XI	Path Learn]]		
Get Snapshot (Ctrl-G)	📋 Highlighting Mode	Learn	Options						Test C	S Learn			
	Browser	Object	Tools				Locators						^
n 🖉 IE													
DOM Tree					DOM Element								
A <html p="" xml:lar<=""></html>	ig="en" xmlns="http://www.w3	.org/1999/xhtm	nl">		4								
♦ <head></head>					tagName			T/	BLE				
⊿ <body></body>					⊿ Attribute	5							
	id="ctl01" action="Books.asp	v" method="rv	net">		border			1					
	div class="aspNetHidden">		001 /		cellspaci	ng		0					
	script type="text/javascript">				class				aGrid				
					id			Ma	inContent_grdBo	ooks			
	scriptsrc="///ebResource.ax	_	Rawwl* typ	e="text/javasci	Tuica			all					
	div class="aspNetHidden">				style			bo	der-collapse: co	lapse; backgro	und-color: w	hite;	
	div class="page">				⊿ Propertie	es							
Þ	<pre><div class="header"></div></pre>	v>			height			33					
	<pre><div class="main"></div></pre>				width			50					
	<h2></h2>				x			22	5				
	⊿				A Selector			Record	ing Activity for	"Inflactra I Li	hran Inforr	nation	Suctor
	<a <="" href="BookEdit.a" td=""><td>spx"></td>	spx">				S		Record	ing Activity for	intecuarto	orary inton	nation .	system
	.⊿ <div></div>	1 - C			CSS			#	Object	Action	Data	(Commer
	✓	Srid" id-"Maint	Content and	Booke" etyle="b	xpath								
	stbody>		oomoni_giu	Joona atyre- D	Test Results								
	<div class="clear"></div>	·											
6	div class="footer">												
	script type="text/iavascript">	decripto											
Ň	seripriype- text/dvdseript >	.sourpt/											

In the Locators section of the Web Spy, enter in the following to locate the table:

//table[@id='MainContent_grdBooks']

Now click on the Test XPath button to display the matching results:

//table[@id='MainContent_grdBooks']	Test XPath	Learn
	Test CSS	Learn
Locator		

Test Results

 Locator: //table[@id='MainContent_grdBooks'], found: 1

Now that we have matched the table, we need to add dynamic XPath to find any row that has the cell containing 'Amsterdam' and find the edit link. You can expand the table and see the rows and cells visually and that will help us create the XPATH:

//table[@id='MainContent_grdBooks']//tr/td[text()='Amsterdam']/../td[5]/a

This XPath consists of the following elements:

1. Finds the table with the specified ID

- 2. Find any row inside that table that contains a cell with the text 'Amsterdam'
- 3. For any matching cell, get its parent row and inside the fifth cell, get any hyperlink

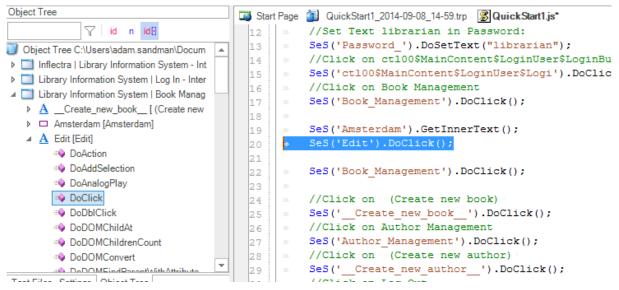
In this case that will correctly locate the Edit link for the book 'Amsterdam'. Now that we have the correct item identified, click on the '**Learn'** button to the right of the '**Test XPath**' button. That will now learn a new object that corresponds to the 'Edit Amsterdam' object:

ŧ	Object	Action		Comment			
<u>©1</u>	Edit	Learn	Edit	Learned Edit			
				0.5%			
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel

Click Finish and the object will have been added to the Object Tree of the current test:

The second se	
Object Tree	
id n id∏	
🧾 Object Tree C:\Users\adam.sandman\Docum 🛛	_
Inflectra Library Information System - Int	
Library Information System Log In - Inter	
🔺 🛄 Library Information System Book Manag	-11
ACreate_new_book [(Create new	
Amsterdam [Amsterdam]	
A Edit [Edit]	
=∳ DoAction	
DoAddSelection	
=∳ DoAnalogPlay	
=∳ DoClick	

To click on this object, simply drag the 'DoClick' function for the new object from the Object Tree into the test script. This will cause the book details page for Amsterdam to be display during test playback:



Since clicking on the Edit link will take you to a different page than where the 'Create New Book' link is available, in the example we have added a second instance of the:

SeS('Book_Management').DoClick();

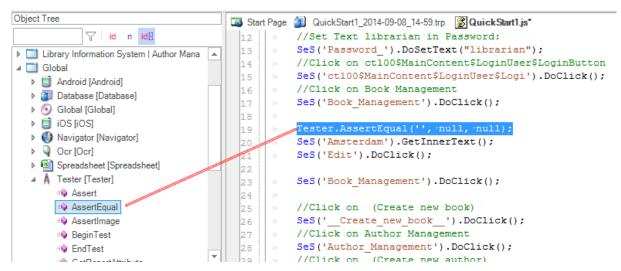
Command, so that Rapise goes back to the main book list page before executing the Create New Book step. If we did not do this, the test would have failed.

Now the line:

```
SeS('Amsterdam').GetInnerText();
```

by itself does not do anything, it just gets the text.

So to make the test more useful, we can use the global **Tester** object to add a step to verify this value. Drag the "AssertEqual" function from the Tester object to your test script just above the SeS("Amsterdam") line:



Now we need to just consolidate these two lines into the actual test. Using the script editor, change the two lines from:

Tester.AssertEqual('', null, null);

```
SeS('Amsterdam').GetInnerText();
```

То

```
Tester.AssertEqual('The values match', 'Amsterdam',
SeS('Amsterdam').GetInnerText());
```

Now click Play to playback the new test:

	Start Page	3	QuickStart1.js	QuickStart1_	2015-07-08	12-27.trp

Drag a column header here to group by that column. # Type Start Name Status Comment Iteration -A A * 🔺 A A Assert 12:27:06.551 ctl00\$MainContent\$LoginUser\$LoginButton.DoClick([]) Pass Returned Value: true 0 Assert 12:27:07.924 Book Management.DoClick([]) Pass Returned Value: true 0 The values match Assert 12:27:08.283 Pass 0 12:27:09.921 Edit.DoClick([]) Pass Returned Value: true 0 Assert 12:27:11.294 Book Management.DoClick([]) Pass Returned Value: true 0 Assert Assert 12:27:12.635 (Create new book) .DoClick([]) Pass Returned Value: true 0 Assert 12:27:14.008 Author Management.DoClick([]) Pass Returned Value: true 0 12:27:15.334 (Create new author) .DoClick([]) Returned Value: true 0 Assert Pass Assert 12:27:16.629 Log Out.DoClick([]) Pass Returned Value: true 0 12:27:16.629 QuickStart1 <u>⊨</u>• ▶ Test Pass Passed:12 Failed:0 Test Pass) <u>-</u>

The test should now pass successfully.

2.2.4 Tutorial: Windows Testing

This section outlines the usage of Rapise for testing a simple Windows Desktop <u>Application Under Test</u> (<u>AUT</u>).

This version of the tutorial uses the **Rapise Visual Language (RVL) scriptless** mode. If you're interested in the <u>JavaScript version</u>, we have a separate tutorial.

Please run the application now. You will find it in the samples directory where you installed Rapise.

By default, that will be C:\Users\Public\Documents\Rapise\Samples\TwoDialogs\TwoDialogs.exe.

You will see the following:

🕌 Inflectra Rapise Two Dialogs Sample	
Please enter your name:	1
Please enter someone else's name:	
Summary:	
OK Canc	el

Please run the application a few times and observe its behaviour. If you press the OK button with the first edit box empty, the application will complain and return you to the dialog box.

-

If you put text in the first edit box but not the second, you will be shown a single line of text in a readonly edit box.

If you enter text in the second edit box as well as the first, pressing OK will put two lines of summary information in the read-only edit box.

An adequate testing strategy for this over-simple application might be to:

1. Put data in the first text box but not the second, and verify that the summary information is correct.

2. Press the OK button with no data in either text box, and verify that a message box is displayed.

3. Verify that if the success "Thank You" message is displayed the edit box input fields are cleared (but not the summary information).

If at this point you do not understand what the application is supposed to do, or the application is not behaving as described here, please contact <u>support</u> and clarify the details before proceeding.

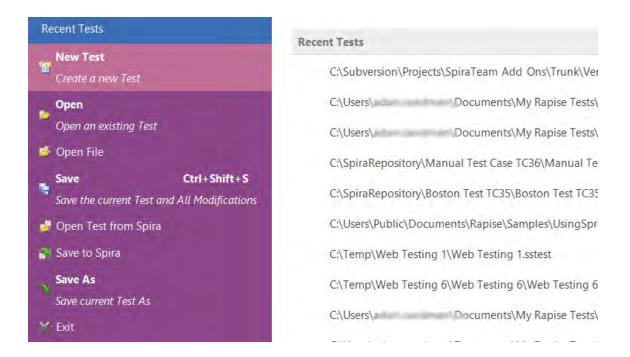
Now, let's use Rapise to implement the first of these tests.

Step 1. Run the TwoDialogs application and leave it in its default start state.

Once you execute the TwoDialogs.exe application it will be displayed on the screen:

🏭 Inflectra	Rapise Two Dia	logs Sar	mple	ж
	Please ente	er your na	ame:	
Plea	e enter someone	e else's na	ame:	
Summary:				
Thank you	. Your name is ch	nris,		
	ОК		Cancel	
			-	

Step 2. Start Rapise and make the window a conveniently large size. Click on the File button (top left). Choose the first option there, "New Test."



Step 3. Navigate to the desired path using the "..." button on the "Create New Test" dialog.

Enter the name of the new test script we're going to write (e.g. "TwoDialogsTest").

Click on the **"Basic: Windows Desktop Application"** methodology. This should always be used for testing Windows desktop applications:

 Create New Test
 X

 Please specify the name of the new test:
 TwoDialogsTest

 Specify the folder where this test should be created:
 C:\Users\adam.sandman\Documents\My Rapise Tests

 C:\Users\adam.sandman\Documents\My Rapise Tests
 ...

 C:\Users\adam.sandman\Documents\My Rapise Tests\TwoDialogsTest\TwoDialogsTest.sstest
 ...

 Rapise supports multiple testing methodologies. Select one to be used for newly created test.
 ...

 Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)
 ...

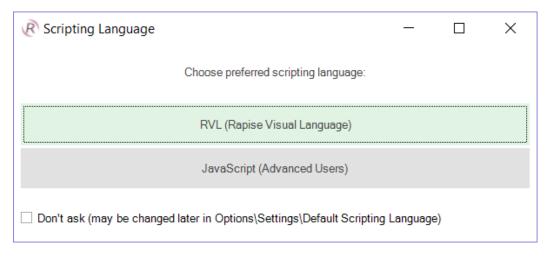
 Manual: Manual-only testing
 ...

 Mobile: Mobile Support
 ...

 Web: Cross-Browser Testing Support
 Cancel

Press the "Create" button.

The following dialog will be displayed:



- Click on the RVL (Rapise Visual Language) button.
- You will now see the following:

File Test Options	RVL						Nocuments\My Rapise Tests\			
open ave 🎓 Save to Spira 🍯 Open Test from Spira	Del Row	🔀 Te		RVL Manual Steps	Record Spy Play	Level 1 (Minimal) +	About Convert Activation Spira Pro Object Mgr		elp	
File	RVL		Shor	touts	Record and Execute	Debugging	Tools	Help		
ect Tree		3 Star	t Page ×	Spira Dashboard	x TwoDialogsTes	t.rvt.xis ×				
V 40 6 46		RVL	+Add New	1						
- 💟 Object Tree C.\Users\adam.sandman\Docum			Flow	Type	Object	Action	ParamName	ParamType	ParamValue	н
Global Global Global		1	Flow	Type	Object	Action	Param Name	Param Type	Param Value	
User Variables		21		1	1					
		3								
		4								
		5								
		6 7								
		8								
		9								
		10								
		11 12								
		13								
		14						0		
		15								
		16								
		17	1							
ject Tree Files Settings	-	18								
perties		19								
21		20								
1										

Step 4. Recording the test sequence.

Press the "Record" button in either the ribbon or on the toolbar. It has an icon like this:



You will see an application selection dialog like the following.

Select From Running Applications	Run Appli	cation		
Available Applications		PID	Path	
Inflectra Rapise Two Dialogs Sam	ple	10796	C:\Users\Public\Docume	ents\Rapise\Sam
Join my network on LinkedIn - Me	ssage (H	7936	C:\Program Files (x86)\M	icrosoft Office \Of
Menu		6120	C:\Program Files (x86)\X-	PRO Vonage X-
🖸 RE: Emails - Message (HTML)		7936	C:\Program Files (x86)\M	icrosoft Office \Of
Snipping Tool		6996	а	
TdmNotify		5012	al	
TdmNotify		200	а	
	10	-		
ibrary	Descri	ption		
Auto	Detect	library aut	omatically	6
Java	Java s	upport for .	AWT/SWING	

- Select the "Inflectra Rapise Two Dialogs Sample" entry.
- Leave the library selection as "Auto."
- Press the "Select" button at the bottom right.

Step 5. Record the activity in the application.

Rapise will pause while it starts the necessary background processes and hooks into the running AUT.

Once those tasks are complete, you will see the following "Recording Activity" for "Inflectra Rapise Two Dialogs Sample" dialog:

#	Object	Action	Data	Comment			
	Verify (Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel

The AUT will be brought to the foreground and Rapise will be minimized.

You will achieve best results in recording if you observe the following guidelines:

(1) Work slowly while recording. Perform one action and wait for the results to be recorded in the Recording Activity dialog as a new grid line-item before going to the next item.

(2) Use the mouse to select controls and operate them. Avoid using keyboard shortcuts and keyboard commands.

Step 6. Perform the Test Activities.

- Click in the first edit box in the TwoDialogs application.
- Type a name in this first edit box.

Watch the Recording activity dialog as you operate the AUT interface. As you press a button or fill a field, notice that the grid in the Recording activity has entries added to it.

船 Inflectra Rapise Two Dialogs Sample	
Please enter your name:	
Please enter someone else's name:	
Summary:	
Thank you, Your name is chris,	
OK Canc	el

As you take these actions, you will see the Recording Activity grid update accordingly:

#	Object	Action	Data	Comment					
尚1	Please en	Set	chris	Please enter your name:: Change text to 'chris'					
2	OK	Action		Press button '(DK'				
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel		

For a full explanation of the controls on this dialog, refer to the reference for <u>Recording Activity</u> <u>Dialog</u>

When you have finished recording the activity for the AUT, press the "Finish" button or CTRL+3.

Note: Do not terminate the TwoDialogs application.

When you do this, the "Recording Activity" dialog will be closed and the AUT will lose focus.

Rapise will now ask you to confirm that you want to add the recorded steps to the current location in the test grid:

🖾 Start Page 🛛 🗙 🔊	Spira Dashboard 🗙	TwoDialogsTest.rvl.xls 🗙	
			New recording is available. Please, choose how to use recorded steps:
Insert Here	Discard		
RVL +Add New			

Click '**Insert** Here', then Rapise will change the view to display the newly recorded test steps. It will look something like the following:

Type Type Please enter Action Press buttor	Object Object r your name:: Change tex Please_enter_you n 'OK'		ParamName Param Name val	ParamType Param Type string	ParamValue Param Value chris	Н
Please enter Action	r your name:: Change tex Please_enter_you	t to 'chris'				
Action	Please_enter_you		val	string	chris	
		DoSetText	val	string	chris	
Press buttor	n 'OK'					
Action	ab OK	DoAction				
Verify that:	Text=Thank you. Your na	ame is chris.				
Assert			message	string	Verify that: Text=Thank	you. Your name
Action	abl Text	GetText				
Condition		output1 == param2				
Param			param2	string	Thank you. Your name	is chris.
	Assert Action Condition	Assert Action Condition	Action Image: Text GetText Condition output1 == param2	Assert message Action Image: Text GetText Condition output1 == param2	Assert message string Action Image: Text GetText GetText Condition output1 == param2 Image: Text Image: Text	Assert message string Verify that: Text=Thank Action Image: Text GetText Image: Text Image: Text Condition output1 == param2 Image: Text Image: Text

Notice that the two steps of the test are automatically documented as Action commands in the test grid, and that they correspond precisely and in the same order as the way they appeared in the Recording Activity dialog during recording. In addition, the Verify checkpoint has been recorded as an Assert...Condition set of steps in the grid.

Step 7: Run ("Play") the recorded test script.

Press the "Play" button on the ribbon or the toolbar.



As the script runs, the Rapise window will be minimized to the taskbar and you will see the results of the script's activities on the TwoDialogs application window.

At the end of the script execution, the Rapise window will be restored and the view will be of the report for the test:

C:\Users\adam.sandman\Document	ts\My	Rapis	· 2 🛛	100		• 📼 🗙		Merge Cells	E Collapse				
Plain Hierarchical			Export to Exce	Export to PDF	Save La	ayout	Choose Columns		Images				
File Object Tree 9	1			port		Layout	1	Data	Display				*
Voject nee		Start Pa	ge 📓 TwoDial	ogsTestis	TwoDial	ogsTest_2014-09-04_12	-09.trp						* X
Object Tree C:/Users/adam.sand	D	rag a d	olumn hea	der her	e to group	by that column.							
Inflectra Rapise Two Dialogs		#	Туре		Start			Name		Status		Comment	Iteration
> POKIOK		* 80	M.	-		100				W	(A)		-
Bease_enter_your_name Global			Message	12:0	9:20.795	Starting scenari	o: Test			Info			
(A) User Functions	-		Assert	12:0	9:21.232	Please enter you	ur name:.D	DoSetText(["chris	["])	Pass	Return	ed Value: true	0
📽 User Variables	1-		Assert	12:0	9:21.466	OK.DoAction([])				Pass	Return	ed Value: true	0
	*	+	Test	12:0	9:21.466	TwoDialogsTest				Pass	Passed	:2 Failed:0	
							Te	st Pass					
	14						U Tota	al:4 Pass:3 Fail:0	Info:1				
Test Files Settings Object Tree													
Properties 4													
Properties #													

Step 8: A refinement on the launching of TwoDialogs.exe.

To date, we have operated on the assumption that the TwoDialogs sample program (application) is running. If this situation remained, the test script would require that the AUT be running before the script started. That would require that the person running the test remembered where it resided. To overcome this, Rapise provides a way to have the script run the program (AUT) before beginning the test.

Rapise comes with a series of useful Global utility objects that can do things such as start applications, kill processes, access the file system, etc. To launch the TwoDialogs application at the start of the test, go to the first row in the test grid and click "**Ins Row**" in the RVL ribbon:

Del Row Ins Row RVL	🌋 Te 💰 Fu	nctions d Short	Manual Steps Web Services tcuts		Level 1 (Minimal) Pebugging	About Convert Activation Spira Pro Object Mgr <u>Tools</u> _2017-05-15_22-52.trp		Find in Help Files Help	
	RVL	+Add New							
		Flow	Type	Object	Action	ParamName	Para	amType	ParamValue
	1	Flow	Туре	Object	Action	Param Name	Param Ty	ре	Param Value
s	2)	#	Please enter y	our name:: Change tex	t to 'chris'				
	3 .	0	Action	🔤 Please_enter_you	DoSetText	val	string		chris
	4 .	1 #	Press button	OK'					
	5 .	1	Action	ab OK	DoAction				
	6.	1 #	Verify that: Te	ext=Thank you. Your na	me is chris.				
	7.3	1	Assert			message	string		Verify that: Text=Thank
	8.	1	Action	abl Text	GetText				
	9 .	0	Condition		output1 == param2				
	10 .	Ø	Param			param2	string		Thank you. Your name
	11								

Now in the new row that was created, choose Action as the type of row:

1 1 1	n	1
Act	ion	~
F		π
/	Action	Ľ
F	Param)/
/	Output	
l	Variable	<u>et</u>
/	Assert	
/	Conditio	n [

Then in the next cell, choose "Global" as the object:

Action	9 G	obal 🗸
Please enter y	oui 🍈	Android 🔺
Action	abl 🎒	Database
Press button '	0К' 👌	File
Action	ab {()}	Functions
Verify that: Te	ext= 🧿	Global
Assert		iOS Global in Global
Action	abl 🚯	Navigator
Condition	0	NeoLoad 🔹 🤊
<u> </u>		

Then in the next cell, choose **DoLaunch** and press ENTER on the keyboard:

Туре	Object	Action	ParamName	ParamType	ParamValue
Туре	Object	Action	Param Name	Param Type	Param Value
Action	💽 Global	DoLaunch	cmdLine	string	
	- 1				

Now you just need to enter in the location of the TwoDialogs application in the final cell (ParamValue) - C:\Users\Public\Documents\Rapise\Samples\TwoDialogs\TwoDialogs.exe:

Flow	Туре	Object	Action	ParamName	ParamType	ParamValue	Н
Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
	Action	🕑 Global	DoLaunch	cmdLine	string	C:\Users\Public\Docun	nents\Rapise\Sampl

Now if you click **Play**, Rapise will launch the application and then complete the recorded test steps.

Advanced Testing using the Object Spy

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the **Spy** button and Rapise will display the <u>Accessible Spy</u>:

🔎 Rapise Spy		-				×
Start Tracking (Ctrl-G)				Le	arn	Objec
- Accessible - Tree		Proper	ties			
- HWND - Tree	-	Proper	ties			

Press **CTRL+G** on the keyboard to start tracking. Hover the mouse over one of the text boxes in the TwoDialogs application and press **CTRL+G** again to stop tracking:

Rapise Spy		— 🗆 ×
Start Tracking (Ctrl-G) Learn Object		Ħ
Accessible		
- Tree	- Properties	
[ROLE_SYSTEM_TEXT.]	StateText	read only focusable
	✓ Primary	
	ChildCount	0
	ChildId	CHILDID_SELF
	DefaultAction	
	Description	
	HWND	[Edit]
	IndexInParent	3
	LocationRECT	{X=759,Y=537,Width=402,Height=42}
	Name	(······)
	Role	ROLE SYSTEM_TEXT
	State	STATE_SYSTEM_READONLY, STATE_SYSTEM_
	Value	
HWND		
Tree	- Properties	
[Edit]	ProcessName	C:\Users\Public\Documents\Rapise\Samples\TwoD
	ScreenHeight	1080
	ScreenWidth	1920
	ScreenX	0
	ScreenY	0
	> Size	406, 46
	✓ Primary	
	AccessibleObject	[ROLE_SYSTEM_WINDOW]
	ClassName	Edit
	> Rect	757, 535, 406, 46
	Text	
	Visible	True 1342179460

This shows you the object you selected, together with its various Windows attributes. If you want to see its place in the hierarchy of the application, right click on [ROLE_SYSTEM_TEXT] in the top-left pane and choose **Parent**. That will display its parent objects:

Rapise Spy		- 🗆 ×
Start Tracking (CtrI-G) Learn Object		Ħ
Accessible	- Properties	
Inflectra Rapise Two Dialogs Sample[RoLE_SYSTEM_DIALOG.] ORIGROLE_SYSTEM_WINDOW.] Cancel[RoLE_SYSTEM_WINDOW.] Please enter your name.[ROLE_SYSTEM_WINDOW.] Please enter your name.[ROLE_SYSTEM_WINDOW.] Please enter someone else's name.[ROLE_SYSTEM_WINDOW.] Summary.[ROLE_SYSTEM_WINDOW.]	StateText Vinitage ChildCount ChildI DefaultAction Description HVND IndexinParent LocationRECT Name Role	focusable 8 CHILDID_SELF OK Inflectra Rapise Two Dialogs Sample[#32770] 3 (X=745,Y=419.Width=431.Height=222) Inflectra Rapise Two Dialogs Sample ROLE_SYSTEM_DIALOG
-	State Value	STATE_SYSTEM_FOCUSABLE
Tree	- Properties	
Inflectra Rapise Two Dialogs Sample[#32770] OK[Button] Cancel[Button] Please enter your name: [Static] [Edit] Please enter someone else's name: [Static]	ProcessName Screen/Height Screen/Width ScreenX ScreenY > Size	C:\Users\Public\Documents\Rapise\Samples\TwoDi 1080 1920 0 0 437.257
[Edit] [Edit] Summary:[Static]	 Primary AccessibleObject ClassName Rect Text 	Inflectra Rapise Two Dialogs Sample[ROLE_SYSTE #32770 742, 387, 437, 257 Inflectra Rapise Two Dialogs Sample
	Visible WindowStyle	True 2496331980

For example in this view you can see all three text boxes, the labels and some of the Windows standard objects (the Window title bar, close icons, etc.). Each of these can be expanded to show their children, and any of the objects can Learned by clicking the **Learn Object** button in the top of the Spy. Once learned, you can use one of the options described above to write a test using it.

2.2.4.1 Using JavaScript

This section outlines the usage of Rapise for testing a simple Windows Desktop <u>Application Under Test</u> (<u>AUT</u>). This version of the tutorial uses the JavaScript test editor option in Rapise. If you'd prefer to use the <u>Rapise Visual Language (RVL)</u>, please go to the main <u>Tutorial</u> instead.

Please run the application now. You will find it in the samples directory where you installed Rapise.

By default, that will be C:\Users\Public\Documents\Rapise\Samples\TwoDialogs\TwoDialogs.exe.

You will see the following:

船 Inflectra Rapise Two Dialogs Sample	
Please enter your name:	
Please enter someone else's name:	
Summary:	
OK Canc	el

Please run the application a few times and observe its behaviour. If you press the OK button with the first edit box empty, the application will complain and return you to the dialog box.

If you put text in the first edit box but not the second, you will be shown a single line of text in a readonly edit box.

If you enter text in the second edit box as well as the first, pressing OK will put two lines of summary information in the read-only edit box.

An adequate testing strategy for this over-simple application might be to:

1. Put data in the first text box but not the second, and verify that the summary information is correct.

2. Press the OK button with no data in either text box, and verify that a message box is displayed.

3. Verify that if the success "Thank You" message is displayed the edit box input fields are cleared (but not the summary information).

If at this point you do not understand what the application is supposed to do, or the application is not behaving as described here, please contact <u>support</u> and clarify the details before proceeding.

Now, let's use Rapise to implement the first of these tests.

Step 1. Run the TwoDialogs application and leave it in its default start state.

Once you execute the TwoDialogs.exe application it will be displayed on the screen:

🛔 Inflectra Rapise Two Dialogs Sample	
Please enter your name:	
Please enter someone else's name:	
Summary:	
Thank you. Your name is chris.	- 1
OK Cance	

Step 2. Start Rapise and make the window a conveniently large size. Click on the File button (top left). Choose the first option there, "New Test."

Recent Tests	Recent Tests
New Test Create a new Test	C:\Subversion\Projects\SpiraTeam Add Ons\Trunk\Ver
Open	C:\Users\ Documents\My Rapise Tests\
Open an existing Test	C:\Users\ Documents\My Rapise Tests\
Open File Save Ctrl+Shift+S Save the current Test and All Modifications	C:\SpiraRepository\Manual Test Case TC36\Manual Te C:\SpiraRepository\Boston Test TC35\Boston Test TC35
Øpen Test from Spira	C:\Users\Public\Documents\Rapise\Samples\UsingSpr
💦 Save to Spira	C:\Temp\Web Testing 1\Web Testing 1.sstest
Save As	C:\Temp\Web Testing 6\Web Testing 6\Web Testing 6
🗙 Exit	C:\Users\ Documents\My Rapise Tests\

Step 3. Navigate to the desired path using the "..." button on the "Create New Test" dialog.

Enter the name of the new test script we're going to write (e.g. "TwoDialogsTest").

Click on the **"Basic: Windows Desktop Application"** methodology. This should always be used for testing Windows desktop applications:

Press the "Create" button.

💐 Create New Test		×
	Please specify the name of the new test:	
TwoDialogsTest		
	Specify the folder where this test should be created:	
C:\Users\adam.sandr	nan\Documents\My Rapise Tests	
	C:\Users\adam.sandman\Documents\My Rapise Tests\TwoDialogsTest\TwoDialogsTest.sstest	
	Rapise supports multiple testing methodologies. Select one to be used for newly created test.	
	Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
	Manual: Manual-only testing	
	Mobile: Mobile Support	
	Web: Cross-Browser Testing Support	
	Cance	əl

The following dialog will be displayed:

R Scripting Language	_		×
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scripting	Language)	

- Click on the JavaScript (Advanced Users) button.
- You will now see the following:

R 😫 🔿 🕞 🛃 4a 🙇 +	RapixeCloud 5.0.7.33 (C)(Isters)adamsani(Documents)(My Rapixe Tests) TwoDialogsTest(TwoDialogsTest)	10 - 15 ×
	Page 2: Spira Dashboard Image: Spira	
Objects	Start Pope 2 TwoDalogsText	- x
T = + ■ O Clean T free Cherner(Doc) + □ Cload W Lear Function © User Vanabler S Dare Vanabler	<pre>//Use 'Record/Learn' button to begin test recording function Test() = {</pre>	~
Oyumin (Piles Settings Population 21		*
	e Datat	5
	Macrosoft/Windows Scient Healt Vennon 5.02 Carpinght (), Microsoft Corporation. Al rights reserved Sel Recordsr Starting.	•
	Selo Tercoran Stating. Verbons Inter: Recording provide InterQUiterstadam sandmari/Document/My Rapine Testing 1(Web Testing 1):s Recording under Interpret: false mape: * Canadina & Josen + ITM Canadina & Josen + ITM	

Step 4. Recording the test sequence.

Press the "Record" button in either the ribbon or on the toolbar. It has an icon like this:



You will see an application selection dialog like the following.

Select From Running Applications	Run Appli	cation			
Available Applications		PID	Path		
👫 Inflectra Rapise Two Dialogs Sam	ple	10796	C:\Users\Public\Documents	\Rapise\Sam	
🖸 Join my network on LinkedIn - Me	ssage (H	7936	C:\Program Files (x86)\Micro	soft Office \Of_	
Menu		6120	C:\Program Files (x86)\X-PR	O Vonage\X-	
ORE: Emails - Message (HTML)		7936	C:\Program Files (x86)\Microsoft Office\Of		
Snipping Tool		6996	а		
🕞 TdmNotify		5012	al		
TdmNotify		200	а		
*	10			*	
Library	Descri	ption			
Z Auto	Detect	library au	tomatically	1	
Java	Java s	upport for	AWT/SWING		
Cancel	Show All	Re Re	fresh	Select	

- Select the "Inflectra Rapise Two Dialogs Sample" entry.
- · Leave the library selection as "Auto."
- Press the "Select" button at the bottom right.

Step 5. Record the activity in the application.

Rapise will pause while it starts the necessary background processes and hooks into the running AUT.

Once those tasks are complete, you will see the following "Recording Activity" for "Inflectra Rapise Two Dialogs Sample" dialog:

Reco	Recording Activity for "Inflectra Rapise Two Dialogs Sample"								
#	Object	Action	Data	Comment					
	Verify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel		
Ready						Advanced>>	Transparent		

The AUT will be brought to the foreground and Rapise will be minimized.

You will achieve best results in recording if you observe the following guidelines:

(1) Work slowly while recording. Perform one action and wait for the results to be recorded in the Recording Activity dialog as a new grid line-item before going to the next item.

(2) Use the mouse to select controls and operate them. Avoid using keyboard shortcuts and keyboard commands.

Step 6. Perform the Test Activities.

- Click in the first edit box in the TwoDialogs application.
- Type a name in this first edit box.

Watch the Recording activity dialog as you operate the AUT interface. As you press a button or fill a field, notice that the grid in the Recording activity has entries added to it.

el

As you take these actions, you will see the Recording Activity grid update accordingly:

Recording Activity for "Inflectra Rapise Two Dialogs Sample"									
#	Object	Action	Data	Comment	Comment				
圆1	Please en	Set	chris	Please enter y	Please enter your name:: Change text to 'chris'				
≥2	ОК	Action		Press button '(Press button 'OK'				
	′erify trl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel		
Paused						Advanced>>	Transparent		

For a full explanation of the controls on this dialog, refer to the reference for <u>Recording Activity</u> <u>Dialog</u>

When you have finished recording the activity for the AUT, press the "Finish" button or CTRL+3.

Note: Do not terminate the TwoDialogs application.

When you do this, the "Recording Activity" dialog will be closed and the AUT will lose focus. Rapise

File Test Options	Edit				
Gen Save ⋧ Save to Spira Open Test from Spira	 Start Page Spira Dashboard Test Manual Steps Functions		No Debugging + Level 1 (Minimal) +	 About Convert to NeoLoad Activation Spira Properties Object Mgr 	Find in Help Files
File	Shortcuts	Record and Execute	Debugging	Tools	Help
Objects Id n IdS Image: Clubers adam.sand Image: Clubers adam.sand Image: Clubers adam.sand Image: Clubers adam.sand Image: Cluber adam.sand Image: Clubers adam.sand Image: Cluber adam.sand Image: Cluber adam.sand	Sample 3 //###################################	## Script Steps ###	ay, December 30, : Change text tr me_').DoSetText		
Objects Files Settings					

will change the view to display the newly recorded script. It will look something like the following:

Notice that the two steps of the script are automatically documented and that they correspond precisely and in the same order as the way they appeared in the Recording Activity dialog during recording.

Step 7: Run ("Play") the recorded test script.

Press the "Play" button on the ribbon or the toolbar.



As the script runs, the Rapise window will be minimized to the taskbar and you will see the results of the script's activities on the TwoDialogs application window.

At the end of the script execution, the Rapise window will be restored and the view will be of the report for the test:

C:\Users\adam.sandman\Document	s\My	Rapis		and the second s		- 🗁 🗙		Merge Cells	1	EE Collapse				
Plain * Hierarchical			Export to Exce	Export to PDF	Save L	Layout	Choose Columns		Images					
File	-		E	xport		Layout		Data		Display				*
Dbject Tree 4		Start Pag	e 📓 TwoDia	logsTest.js	🔄 TwoDia	logsTest_2014-09-04_12-0	19.trp							- 3
Y la n laß	D	anac	olumn hea	der her	e to grou	p by that column.								
Object Tree C:\Users\adam.sand	1			uci iici		p by that column								
Inflectra Rapise Two Dialogs		#	Туре		Start	*		Name			Status		Comment	Iteration
 b) OK [OK] b) Please enter your name 	Ы	* 10	(A)	-		N.					W.	AT.		-
Fiease_enter_your_name			Message	12:0	9:20.795	Starting scenario	: Test				Info			
User Functions			Assert	12:0	9:21.232	Please enter your	name:.D	oSetText(["chris	'D		Pass	Return	ed Value: true	0
👹 User Variables			Assert	12:0	9:21.466	OK.DoAction([])					Pass	Return	ed Value: true	0
			Test	12:0	9:21.466	TwoDialogsTest					Pass	Passed	:2 Failed:0	
							Ter	t Pass						
	14-							4 Pass 3 Fail 0						
							lota	4 Pass 3 Fail 0	nto 1					
	1													
Test Files Settings Object Tree														
Properties 4														
21 21														

Step 8: A refinement on the launching of TwoDialogs.exe.

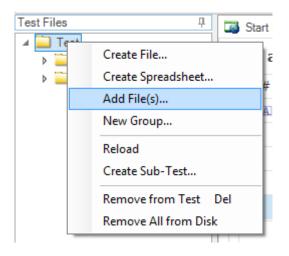
To date, we have operated on the assumption that the TwoDialogs sample program (application) is running. If this situation remained, the test script would require that the AUT be running before the script started. That would require that the person running the test remembered where it resided. To overcome this, Rapise provides a way to have the script run the program (AUT) before beginning the test.

Rapise has an underlying scripting language based on JavaScript (see <u>Scripting</u>). This help system covers available scripting objects in detail from a practical perspective. For the moment, we want to simply take the shortest path to starting the application before attempting to run the test.

There are at least 3 ways of adding application launch code to your test.

Way 1: Drag The File from the Test Files view

First, switch to Test Files view. Right-click on "Test" folder and choose "Add File(s)..." menu item:



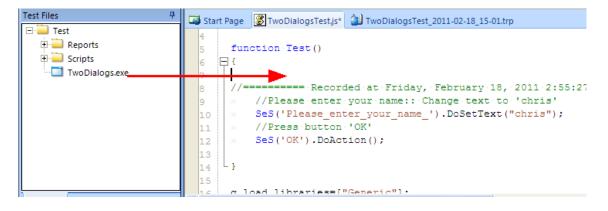
And select the location of the TwoDialogs.exe (normally, it is C:\Program Files\Inflectra\Rapise \Samples\TwoDialogs.exe),

🔍 🗢 📕 « Rapise 🕨 Samples 🕨 TwoDialog	gs ▶ 🚽 ♣ Search TwoDialogs 🔰
Organize New folder	i≡ * □ 0
Samples Adobef TwoDialogs Analog CrossBr DotNet Extensit Java LibraryI Sample Sample Simulat	brary Arrange by: Folder •
UsingC + + File name:	™

Now you have the executable as a part of your test files set:

Test Files							
🔺 🦳 Test							
Reports							
Scripts							
TwoDialogs.exe							
Test Files Settings Object Tree							

If you wish to launch TwoDialogs.exe once then just double-click on it in the tree. If you wish it to be launched every time the test starts then simply drag it from the tree into the source code:



The proper launch statement will be inserted:

```
function Test()
{
     Slobal.DoLaunch('../../Rapise/Samples/TwoDialogs/TwoDialogs.exe');
     //======= Recorded at Tuesday, September 27, 2011 4:06:19 PM: ======
     //Please enter your name:: Change text to 'chris'
     SeS('Please_enter_your_name_').DoSetText("chris");
     //Press button 'OK'
     SeS('OK').DoAction();
}
```

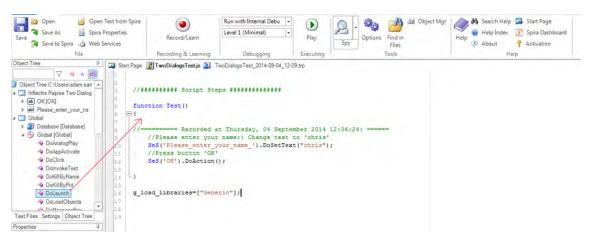
Way 2: Type the Code

```
The Global object contains methods that are available to all scripts.
Select the TwoDialogs. js file in the Test Files view of the Rapise main page.
Double-click the file name to open it in the main (editing) window of Rapise. You will see the
generated script from the recording session from earlier steps in this sample.
Place the cursor in the main editing window and click on the first line after
function Test()
{
Now type
Global.
As soon as you type the ".", Rapise will give you a drop down list of all the available methods
available in the Global object:
=💊 DoAnalogPlay(path, left, to
≡© DoAppActivate(title)
=©DolnvokeTest(pathToTes
=💊 DoKillByName(processNar
≡© DoKillByPid(pid)
=🗣 DoLaunch(cmdLine, wrkD
=💊 DoMessageBox(prompt, b
= OOCrTesseract(img)
=Q DoOcrTextract(img)
=💊 DoSendKeys(keys)
Select the DoLaunch(cmdLine, wrkD) member and hit the Enter key.
Now your script contains the line:
Global.DoLaunch('')
You need to correct the references to the command line:
```

```
Global.DoLaunch('"C:\\Program Files\\Inflectra\\Rapise\\Samples\\TwoDialogs\
\TwoDialogs.exe"');
```

Way 3: Drag the Action from the Objects Tree

You may drag the method template from the <u>Object Tree</u> view. Expand the "Global" node and select the "DoLaunch" action in it. Drag the node into the proper position inside the script source:



Template call is inserted:

4	
5	function Test()
6	₽ (
7	Global.DoLaunch('');
8	//====== Recorded at Friday, Feb
9	» //Please enter your name:: Chang
10	» SeS('Please_enter_your_name_').D
11	» //Press button 'OK'

Now you need to correct the references to the command line:

```
Global.DoLaunch('"C:\\Program Files\\Inflectra\\Rapise\\Samples\\TwoDialogs\
\TwoDialogs.exe"');
```

Advanced Testing using the Object Spy

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the **Spy** button and Rapise will display the <u>Accessible</u> <u>Spy</u>:

🔎 Rapise Spy	-	[j.	×
Start Tracking (Ctrl-G)			Lear	n Objec
- Accessible - Tree	- Proper	ties		
- HWND - Tree	- Prope	ties		

Press **CTRL+G** on the keyboard to start tracking. Hover the mouse over one of the text boxes in the TwoDialogs application and press **CTRL+G** again to stop tracking:

A Rapise Spy		- 🗆 X
Start Tracking (Ctrl-G) Learn Object		H -
Accessible		
- Tree	 Properties 	
[ROLE_SYSTEM_TEXT.]	StateText	read only focusable
	✓ Primary	
	ChildCount	0
	Childld	CHILDID_SELF
	DefaultAction	
	Description	
	HWND	[Edit]
	IndexInParent	3
	LocationRECT	{X=759,Y=537,Width=402,Height=42}
	Name	
	Role	ROLE_SYSTEM_TEXT
	State	STATE_SYSTEM_READONLY, STATE_SYSTEM_F
	Value	~
- HWND		
- Tree	- Properties	
[Edit]	ProcessName	C:\Users\Public\Documents\Rapise\Samples\TwoDi
()	ScreenHeight	1080
	ScreenWidth	1920
	ScreenX	0
	ScreenY	0
	> Size	406, 46
	✓ Primary	
	AccessibleObject	[ROLE_SYSTEM_WINDOW]
	ClassName	Edit
	> Rect	757, 535, 406, 46
	Text	
	Visible	True
	WindowStyle	1342179460
		*

This shows you the object you selected, together with its various Windows attributes. If you want to see its place in the hierarchy of the application, right click on [ROLE_SYSTEM_TEXT] in the top-left pane and choose **Parent**. That will display its parent objects:

Rapise Spy		- 🗆 ×
Start Tracking (Ctrl-G) Learn Object		
Accessible		
Tree	- Properties	
 Inflectra Rapise Two Dialogs Sample[ROLE_SYSTEM_DIALOG,] 	StateText	focusable
 OK[ROLE_SYSTEM_WINDOW.] 	Y Primary	
Cancel[ROLE_SYSTEM_WINDOW.]	ChildCount	8
Please enter your name: [ROLE_SYSTEM_WINDOW,]	Childld	CHILDID_SELF
Please enter your name: [ROLE_SYSTEM_WINDOW,]	DefaultAction	OK
 Please enter someone else's name: [ROLE_SYSTEM_WINDOW,] 	Description	
 Please enter someone else's name: [ROLE_STSTEM_WINDOW,] Please enter someone else's name: [ROLE_STSTEM_WINDOW,] 	HWND	Inflectra Rapise Two Dialogs Sample[#32770]
	IndexInParent	3
[ROLE_SYSTEM_WINDOW.]	LocationRECT	{X=745,Y=419,Width=431,Height=222}
Summary:[ROLE_SYSTEM_WINDOW.]	Name	Inflectra Rapise Two Dialogs Sample
	Role	ROLE_SYSTEM_DIALOG
	State	STATE_SYSTEM_FOCUSABLE
	Value	
HWND		
Tree	- Properties	
 Inflectra Rapise Two Dialogs Sample[#32770] 	ProcessName	C:\Users\Public\Documents\Rapise\Samples\TwoD
OK[Button]	ScreenHeight	1080
Cancel[Button]	ScreenWidth	1920
Please enter your name: [Static]	ScreenX	0
[Edit]	ScreenY	0
	> Size	437, 257
Please enter someone else's name:[Static]	✓ Primary	
[Edit]	AccessibleObject	Inflectra Rapise Two Dialogs Sample[ROLE_SYSTI
[Edit]	ClassName	#32770
	> Rect	742, 387, 437, 257
Summary:[Static]		
	Text	Inflectra Rapise Two Dialogs Sample
		Inflectra Rapise Two Dialogs Sample True

For example in this view you can see all three text boxes, the labels and some of the Windows standard objects (the Window title bar, close icons, etc.). Each of these can be expanded to show their children, and any of the objects can Learned by clicking the **Learn Object** button in the top of the Spy. Once learned, you can use one of the options described above to write a test using it.

2.2.5 Tutorial: REST Web Services

In this section you shall learn how to test a RESTful web services API using Rapise. We shall be using a demo application called **Library Information System** that has a dummy RESTful web service API available for learning purposes. You can access this sample application at http://www.libraryinformationsystem.org, and its RESTful web service API can be found at: www.libraryinformationsystem.org, and its RESTful web service API can be found at:

What is REST and what is a RESTful web service?

REpresentational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a web API design model that offers greater simplicity over other web service protocols such as SOAP and XML-RPC.

A RESTful web API (also called a RESTful web service) is a web API implemented using HTTP and REST principles. Unlike SOAP-based web services, there is no "official" standard for RESTful web APIs. This is because REST is an architectural style, unlike SOAP, which is a protocol.

Overview

Creating a REST web service test in Rapise consists of the following steps:

- 1. Using the REST query builder to create the various REST web service requests and verify that they return the expected data in the expected format.
- Parameterizing these REST web service requests into reusable templates and saving as Rapise learned objects.
- 3. Generating the test script in Javascript that uses the learned Rapise web service objects.

We shall discuss each of these steps in turn.

1. Using the REST Query Builder

Create a new test in Rapise called MyRestTest1.sstest. For methodology, choose **Basic: Standard Scripting Mode** and Rapise will create a new blank test project.

Once you have created it, click on the "Web Services" icon in the Test ribbon to add a new web service definition to your test project:

🖾 Start Page	📀 Spira Dashboard					
遼 Test	🐏 Manual Steps					
Functions	👍 Web Services					
Shortcuts						

This will display the Add New Web Service dialog box:

Add New Web Service		×						
	This wizard will create a new web service definition file inside the current test:							
●REST ○SOAP								
LibraryInformationSystem	lrest							
C:\Users\adam.sandman\	C:\Users\adam.sandman\Documents\My Rapise Tests\MyRestTest1							
	Create Cancel							

Choose **REST** as the type of web service you want to create.

Then, enter the name of the web service that you're going to add, in this case enter "LibraryInformationSystem.rest" and click "Create".

This will add the REST web services definition file to your test project:

File Test Options REST			
Save Update Add Ad	dd Add	Create Script Remove Step	
Requests Object Tree Header Paran	meter Credentials	Dean	
File	Edit	Script Steps	۸. ۱۹۱۹
	REST Request	MyRestTest1 p 🔯 LibraryInformationSystem.m	- 2
 LIS nett 		New_Request	
 New_Request LoraryInformationSystem rest 	100110		
-> New_Request	Method	GET v	9 Send
	Body:	<u>o</u>	
		Auto Raw XML JSON	
			^
		4	
Objects Files PEST Settings			
Properties			
34CL 2.4 (102			
	Response Body		
	Auto Raw XI	AL ISON	
	<		
		Imponse Hedders Output Warnings Error Find Results	

You will see on the right hand side, there is a new document editor for the .rest file. This is the REST web services query form. It lets you send test HTTP requests to the web service under test and inspect the output being returned.

If you open up API documentation for our sample application (<u>www.libraryinformationsystem.org/</u> <u>Services/RestService.aspx</u>) you will see that it exposes several operations for retrieving, adding, updating and deleting books and authors in the system. For this tutorial we shall perform the following operations:

- 1. Get the special SessionID to identify our test session
- 2. Get a list of books in the system
- 3. Add a new book to the system and verify that it was added

According to the documentation that means we will need to send the following requests:

(i) Get a Unique Session

	URL:		http://www.libraryinformationsystem.org/ Services/RestService.svc/session
	Method:		GET
	Returns:		Unique session ID that is passed to other requests to keep data separate for different demo users
(ii)	Get this list	t of books	
	URL:		http://www.libraryinformationsystem.org/ Services/RestService.svc/book? session_id={session_id}
	Method:		GET
	Returns:		Array of book objects
(iii) Add a new	book to the list	
	URL:	http://www.libraryinformationsystem session_id={session_id}	n.org/Services/RestService.svc/book?
	Method:	POST	
	Body:	Pass a populated book object:	

The first request will be to get the unique session ID that we will need to pass to the other requests. This is needed by our sample application to prevent testing by different users interfering with each other. To create this request, simply enter the following information on the REST Request form:

- Name: Get_Session
- Method: GET
- URL: http://www.libraryinformationsystem.org/Services/RestService.svc/session

You should now have it populated as illustrated below:



This web service request requires that we pass credentials by means of HTTP Basic authentication. So click on the "REST" tab in the Rapise ribbon and click on the "Add Credentials" button.

File	Test Opti	ons R	EST			
Save Requests	Update Object Tree	Add Header	Add Parameter	Add Credentials		Create Script Remove Step Clean
	File		Edit		Script Steps	

This will display the "Add Credentials" dialog box:

Usemame:	librarian		•
assword:	******		
		Add Cancel	_

Enter librarian as both the username and password and click "Add".

Now click the "Send" button and the request will get sent to the web service:

🗔 Start Page 💈	MyRestTest1.js	LibraryInforma	tionSystem.rest
REST Request			
Name:	Get_Session		
Method:	GET 🗸	http://www.librar	yinformationsystem.org/services/restservice.svc/session
Credentials:	librarian 🗙		
Body:	\odot		
Response Headers			
	Name		
Status Code			200 OK
Content-Length			113
Cache-Control			private
Content-Type			application/xml; charset=utf-8
Date			Mon, 02 Jan 2017 17:38:22 GMT
Set-Cookie			ASP.NET_SessionId=aucc1bizwswrj1diqurgdjmk; path=/; HttpOnly
Server			Microsoft-IIS/8.0
X-AspNet-Versi	ion		4.0.30319
X-Powered-By			ASP.NET

The Response Header tab will display the headers coming back from the web service. The Status Code **200 OK** means that the request succeeded and that data was returned. If you click on the "**Response Body - XML**" tab, you will see the XML serialized data returned from the web service:

🖾 Start Page 👔	MyRestTest1.js 🔂 LibraryInformationSystem.rest
REST Request	
Name:	Get_Session
Method:	GET v http://www.libraryinformationsystem.org/services/restservice.svc/session
Credentials:	librarian 🔀
Body:	\odot
Response Body	
Auto Raw X	ML JSON
	<pre>zersion="1.0" encoding="utf-8"?> g xmlns="http://schemas.microsoft.com/2003/10/Serialization/">37f5c5d6-f21e-43bc-9954-052ffb486f7e</pre>

Since Rapise uses JavaScript as its scripting language, it is usually easier to work with JSON (JavaScript Object Notation) serialized data rather than XML. In the case of the sample Library Information System web service, you can change the format that it accepts and retrieves by sending two special HTTP headers:

- Content-Type: application/json
- Accept: application/json

To add these headers to the request, simply click on the "Add Header" button in the REST ribbon tab. This will display the following dialog box:

C + + + I	e: Accept	
ue: application/json	e: application/json	

Choose the HTTP Header "Accept" from the list and enter "application/json" as the value. Repeat for the "Content-Type" header. You should now have the following populated request:

REST Request		aryInformationSystem.rest					
Name:	ne: Get_Session						
Method	GET o http:	//www.libraryinformationsystem.org/services/rest	v	Send			
Credentials: Headers:	librarian 🔀						
	Name	Value	Operations				
	Accept	application/json	×				
	Content-Type	application/ison	X				

Now click the "Send" button and the request will get sent to the web service:

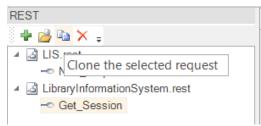
Response Headers	
Name	
Status Code	200 OK
Content-Length	38
Cache-Control	private
Content-Type	application/json; charset=utf-8
Date	Tue, 03 Jan 2017 00:25:48 GMT
Set-Cookie	ASP.NET_SessionId=f1lhsizzrhgbwlaexymzw3qq; path=/; HttpOnly
Server	Microsoft-IIS/8.0
X-AspNet-Version	4.0.30319
X-Powered-By	ASP.NET
Response Body Response Headers Output Warning	s Errors Find Results

The Response Header tab will display the headers coming back from the web service. Note that the returned Content-Type is listed as "application/json" as requested. If you click on the "Response Body - JSON" tab, you will see the JSON serialized data returned from the web service:

Response Body					
Auto Raw XML JSON					
1 " fda25e06-119e-40	a0-adf	d-cf509	ce7e	4db"	
<					
Response Body Response Headers	Output	Warnings	Errors	Find Results	

We have now completed the creation of our first test operation. Click on the "Save Requests" button in the Rapise REST Ribbon to make sure our changes have been saved.

Now click on the "Clone request" icon in the REST request explorer in the left-hand side of the screen:



This will display the Clone Request dialog box. This lets us create a new REST request that contains the headers and authentication already defined on our existing request. This will save time over creating a new REST request from scratch:

Please er	ter the name of the new clon	ed copy of the request:	
Name:	Get_Book		
		Clone	Cancel

Enter the name "Get_Books" in the dialog box and click the "Clone" button. This will create a new REST request with this name:

		brary Information System.re		
REST Request				
Name:	Get_Book			
Method:	GET v http	//www.librarymformationsystem.org/services/rests	mes.svojsession	* Send
Credentials. Headers:				
	Name	Value	Operations	1
	Accept	application/json	×	
	Content-Type	application()son	×	
Body	Content-Type	application(json	8	

For this request we need to pass through the SessionID in the querystring. Rather than hardcoding it in the URL, we can make use of the parameterization feature of Rapise. Click on the **"Add Parameter"** button in the Rapise REST Ribbon. This will display the "Add Request Parameter" dialog box:

e: session_id	
e: 82499bcc-37e4-4c64-820e-a2d798cd1e84	

Enter in the following:

o Name: session_id

o Value: 82499bcc-37e4-4c64-820e-a2d798cd1e84

(you can also copy and paste the value returned by the Get_Session command)

Click the "Add" button and the parameter will be added to the request. Now change the URL to: URL: http://www.libraryinformationsystem.org/Services/R

Then position the caret at the end of this URL and click the "Insert in URL" button. This will insert the parameter token in the URL at the specified point:

Start Page	MyRestTest1.a 🔄 Lib	raryInformationSystem.rest		
REST Request				
Namo	Get_Book			
Method.	GET ~ http	//www.libraryinformationsystem.org/services/restservice.	avc/book?session_id+(session_id)	Send
Credentials:	librarian 🔀			
Headers	\odot			
	Name	Value	Operations	
	Accept	application/json		
	Content-Type	application (son		
Parameters:	۲			
	Namo	Value	Operations	
	(session_id)	82499bcc-37e4-4c64-820e-a2d798cd1e84	Insert in URL X	
	1.			
Body	0			
BOOA	O			

Now click the "Send" button and the request will get sent to the web service. This will return the list of books serialized as a JSON array of objects:

Stort Page	MyRestTestLis Libr	aryInformationSystem.rest			
REST Reques	t				
Name	Get_Book				
Method:	GET ~ http:	/www.libraryinformationsystem.org/services/restserv	ice.svc/baak?session_id=(session_id)	~ s	end Record
Credentials:	librarian 😿				
Headers;	\odot				
	Name	Value	Operations		
	Accept	application/json	× .		
	Content-Type	application/json	×		
Parameters:	•				
	Name	Value	Operations		
	(session_id)	82499bcc-37e4-4c64-820e-a2d798cd1e84	Insert in URL X		
Body.	\odot				
Response Body					
Auto Raw	XML JSON				

Aud Kaw An [Sun] [["Author":["Age":125,"14":3,"Name":"Arthur Conan Doyle"], "AuthorId":3,"DateAdded":"//Date(1421609052000-0500)//","DateAddedIso":"2015-01-18719:24:12","Gen ~

We have now completed the creation of our second test operation. Click on the "Save Requests" button in the Rapise REST Ribbon to make sure our changes have been saved.

Now click on the "Clone request" icon in the REST request explorer in the right-hand side of the screen. Enter the name "Add_Book" in the dialog box and click the "Clone" button. This will create a new REST request with this name:

EST Request				-
Name	Add_Book			
Method:	GET v http://w	ww.libraryinformationsystem.org/services/restservice	svc/book?session_id=(session_id)	 Bend
Credentials:	librarian 🔀			
Headers	\odot			
	Name	Value	Operations	
	Accept	application/json	×	
	Content-Type	application/json	×	
Parameters.	•			
	Name	Value	Operations	_
	(session_id)	82499bcc-37e4-4c64-820e-a2d798cd1e84	Insert in URL X	
	-			
Body.	\odot			

This operation will add a new book to the system, so it's a POST request. Change the Method type in the dropdown list from "GET" to "**POST**".

Expand the "Body" field on the form. This is where you can enter in an XML or JSON serialized Book record that will get added to the system. For now we'll leave this blank and let Rapise serialize the body for us later on when we actually write our test script. So we should now have:

	🕈 MyRestTest1.js 📓 Libr	aryInformationSystem.re			
REST Request					
Name.	Add_Book				
Method:	POST ~ http://	www.libraryinformationsys	tem.org/services/restse	vice.svc/book?session_id={session_id}	- Send
Credentials. Headers	librarian 😿				
	Nome		Value	Operations	
	Accept	application/json		×	
	Content-Type	application/json		×	
Parameters	\odot				
	Name		Value	Operations.	
	(session_id)	82499bcc-37e4-4c	64-820e-a2d798cd1e84	Indent in URL 🕅 🔀	
Body:	\odot				
	Auto Raw XML	ISON			
	1				<u>^</u>
Data Data					

We have now completed the creation of our third test operation. Click on the "Save Requests" button in the Rapise REST Ribbon to make sure our changes have been saved.

2. Saving the REST Requests as Objects

Now that we have created our three REST requests, the next step is to actually create the Rapise objects that we can use in our JavaScript test scripts. Click on the "Update Object Tree" button in the Rapise REST Ribbon to tell Rapise to update the Object Tree with our new requests:



Rapise will now take you REST requests and use that to update the main <u>Object tree</u>. Click on the "Object" tab of the main Rapise explorer, click the Refresh icon and you will see the "LibraryInformationSystem" heading displayed, with the three saved REST request listed underneath:

Objects	
v id n id⊞	
▲ 💟 Object Tree C:\Users\adam.sandman\	•
LibraryInformationSystem	
Add_Book	
▹ - Get_Book	
✓ -● Get_Session	
≡le DoExecute	
≡le DoVerify	
→ SetCredential	•
4 •	
Objects Files REST Settings	

If you expand one of the REST requests (e.g. Add_Book), you'll see that it has a single operation **"DoExecute"** that executes the web services and a series of properties available for inspecting or updating any part of the REST request prior to it being sent to the server.

In the next section we shall illustrate how you can write a test script using these learned objects.

- a) You can either have Rapise **generate test scripts** and verification points automatically (described in section 3a), or
- b) You can manually **write the test scripts** using the objects and the Rapise code editor (described in section 3b)

3a. Generating REST Test Scripts

Inside the REST request explorer, double-click on the Get_Session function to open up the request:

Name:	Get_Session				
Method:	GET v http:	//www.libraryinformationsystem.org/services/rests	ervice svc/session	*	Sent
Credentials:	librarian 🔀				
Headers:					
Headers:	\odot				
Heoders:	0	Value	Operations		
teaders:	0	Value	Operations		

Click on the **Send** button to send the sample request. Once that has succeeded, you will see the **Record** button appear to the right:

Nome	Cet_Session									
Method	GET - http:	//www.libraryinformationsystem.org/services/rests	ervice.svc/session	~	Send Record					
Credentials:	librarian 🔀									
Headers:	1991									
Headers:	۲									
Headers:	Name	Value	Operations.							
Headers:	2	Value application/json	Operations.							

Click that button and the request will get added to the list of recorded steps:

Get_Session GET		Create Script
		Remove Step
		Clean
	Script Steps	

Now open up the Get_Book request and follow the same procedure:

- 1. Click on the 'Send' button to execute the request
- 2. Click on the 'Record' button to record the action as a script step

This time we also want to verify the result. You will see a list of books returned in the **Verify** box underneath the Body section:

GET v http://www.libraryinformationsystem.org/services/restservice.svc/book?session_id=(session_id) v	Send	Record	Verify
librarian 🔀			
\odot			
\odot			
\odot			
response[14]			
4 0 4 Author			
Age=125			
ld=3 Name=Arthur Conan Doyle			
Authord#=3			
DateAdded=1/18/2015 2:24:12 PM			
DateAddedIso=1/18/2015 7:24:12 PM			
> Genre			
Genreld=2			

If you select the overall array **response[14]** and click the main '**Verify**' button next to the Record button, the system will automatically add a verification step that verifies all of the values. To try this, click the **Verify** button. This will add a bold verification step to the recorded script:

Get_Session GET
*Get_Books GET

You will see a script step recorded with a verification test added (it's shown in bold with an asterisk*):

However, in many cases you only want to verify certain properties. For example, we might want to just verify that 14 books are returned, and that the first book has the right name.

To do this, right-click on the **response**[14] entry to display the verification content menu:

response[14]			
⊿ 0	Read Response		
▶ Aut	Verify Response :: [{ "Author": {	"Age": 125,	"ld": 3,
Aut	Read Response length		
Dat			
Dat	Verify Response length 14		

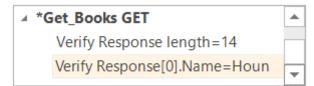
Choose the option 'Verify Response length=14'. This adds the following step to the recorded script:



Now we want to verify the name of the first book returned. To do that, expand the "0" index entry and then right-click on the "Name" property returned:

ſ	respo	nse[14]
	a 0	
	Þ	Author
		AuthorId=3
		DateAdded=1/18/2015 2:24:12 PM
		DateAddedIso=1/18/2015 7:24:12 PM
	Þ	Genre
		Genreld=2
		ld=1
		IsOutOfPrint=False
		Name=Hound of the Baskervilles
	▶ 1	Read Response[0].Name
_	5	Verify Response[0].Name Hound of the Baskervilles

Choose the option to **Verify Repsonse[0].Name = Hound of the Baskervilles**. This will add a verification step for this specific property:



Now we add the last two requests - adding a book and verifying that it was added. To do that open up the two requests and click **Send** then **Record**:

Add_Book (POST)Get_Book (GET)

The **Add_Book** won't actually work at this point because we've not populated the body, but it will be good enough to create the test script. For the second instance of Get_Book don't use the Verify option since we will want to code that by hand to match the book we actually added.

Once you are done, you should have:

*Get_Book GET		Create Script
Add_Book POST		Remove Step
Get_Book GET	-	Clean
Script Steps		-

Now click on the Create Script option and Rapise will generate the following code for you:

```
function Test()
{
      var
LibraryInformationSystem_Get_Session=SeS
('LibraryInformationSystem_Get_Session');
LibraryInformationSystem_Get_Session.SetRequestHeaders([{"Name"
: "Accept", "Value": "application/json" }, { "Name": "Content-
Type", "Value": "application/json"}]);
      LibraryInformationSystem_Get_Session.DoExecute();
      var
LibraryInformationSystem_Get_Books=SeS
('LibraryInformationSystem Get Books');
LibraryInformationSystem_Get_Books.SetRequestHeaders([{"Name"
:"Accept","Value":"application/json"},{"Name":"Content-
Type", "Value": "application/json" ]);
      LibraryInformationSystem Get Books.DoExecute();
      Tester.Assert('Compare call result for http://
www.libraryinformationsystem.org/services/restservice.svc/book?
session id={session id}',
LibraryInformationSystem_Get_Books.GetResponseBodyText(), "...");
LibraryInformationSystem_Get_Books.DoVerify('LibraryInformationSystem_Get_
Books Response', "length", 14);
LibraryInformationSystem_Get_Books.DoVerify('LibraryInformationSystem_Get_
Books Response', "[0].Name", "Hound of the Baskervilles");
      var
LibraryInformationSystem_Add_Book=SeS
('LibraryInformationSystem_Add_Book');
```

LibraryInformationSystem_Add_Book.SetRequestHeaders([{"Name"

```
:"Accept", "Value": "application/json" }, { "Name": "Content-
Type", "Value": "application/json" }]);
LibraryInformationSystem_Add_Book.DoExecute();
var
LibraryInformationSystem_Add_Book=SeS
('LibraryInformationSystem_Add_Book.SetRequestHeaders([{"Name"
:"Accept", "Value": "application/json" }, { "Name": "Content-
Type", "Value": "application/json" }]);
LibraryInformationSystem_Get_Books.SetRequestHeaders([{"Name"
:"Accept", "Value": "application/json" }]);
LibraryInformationSystem_Get_Books.SetRequestHeaders([{"Name"
:"Accept", "Value": "application/json" }]);
LibraryInformationSystem_Get_Books.SetRequestHeaders([{"Name"
:"Accept", "Value": "application/json" }]);
LibraryInformationSystem_Get_Books.DoExecute();
}
```

If you click **Play** on this script as written, you will see that the tests to retrieve the books work correctly, but the test of adding a new book fails:

ag a c	column head	ler here to group	by that column.				
#	Type	Start	-	Name	Status	Comment	Iteration
r 10	Message	= 17:52:55.623	Starting scenario: Test		Info	16) 1	-(
	Assert	17:52:56.511	Get_Session.DoExecute([])		Pass	Returned Value: true	0
	Assert	17:52:56.978	Get_Book.DoExecute([])		Pass	Returned Value: true	0
	Assert	17:52:56.985	Compare call result for http://	www.libraryinformationsystem.org/servi	Pass	[{"Author":{"Age":125,"Id":3,"Name":"Arthur Cor	ian 0
	Assert	17:52:57.210	Add_Book.DoExecute([])		Fall	Returned Value: false	0
	Assert	17:52:57.793	Get_Book.DoExecute([])		Pass	Returned Value: true	0
÷	Test	17:52:57.798	MyRestTest1		Fail	Passed:4 Failed:1	

This is as we'd expect since we've not populated the new book yet!

To make the template test script more useful, we should make the following changes:

- Add comments to each of the sections to describe the purpose
- Add code to get the session ID from the first call and pass to the subsequent calls
- Create a JavaScript object to contain the new book information, and pass that to the Add Book function
- o Get the new book ID from the result of the Add Book function and use it later on.
- o Change the Tester.Assert code to check just specific properties rather than the entire object list.

The complete updated test script looks like the following. We have highlighted the new/changed lines in yellow:

```
//First get the session
var
LibraryInformationSystem_Get_Session=SeS
('LibraryInformationSystem_Get_Session');
LibraryInformationSystem_Get_Session.SetRequestHeaders([{"Name"
"."Accept", "Value": "application/json"}, {"Name": "Content-
```

```
Type", "Value": "application/json" ]);
      LibraryInformationSystem_Get_Session.DoExecute();
      var sessionId =
LibraryInformationSystem_Get_Session.GetResponseBodyObject();
      Tester.Message('Session ID: ' + sessionId);
      //Get the list of books
      var
LibraryInformationSystem Get Book=SeS
('LibraryInformationSystem_Get_Book');
LibraryInformationSystem_Get_Book.SetRequestHeaders([{"Name"
:"Accept", "Value": "application/json" }, { "Name": "Content-
Type", "Value": "application/json" ]);
     LibraryInformationSystem Get Book.DoExecute({ "session id":
sessionId });
      //Verify the data
LibraryInformationSystem Get Books.DoVerify('LibraryInformationSystem Get
Books Response', "length", 14);
LibraryInformationSystem_Get_Books.DoVerify('LibraryInformationSystem_Get_
Books Response', "[0].Name", "Hound of the Baskervilles");
      //Add a book
      var newBook = {
            Name: "A Christmas Carol",
            AuthorId: 2,
            GenreId: 3
      };
      var
LibraryInformationSystem Add Book=SeS
('LibraryInformationSystem Add Book');
LibraryInformationSystem_Add_Book.SetRequestHeaders([{"Name"
:"Accept", "Value": "application/json" }, { "Name": "Content-
Type", "Value": "application/json" ]);
      LibraryInformationSystem_Add_Book.SetRequestBodyObject(newBook)
      LibraryInformationSystem Add Book.DoExecute({ "session id":
sessionId });
      //Get the ID of the new book
      newBook = LibraryInformationSystem Add Book.GetResponseBodyObject();
      Tester.Message("New Book ID: " + newBook.Id);
      //Verify the data
LibraryInformationSystem_Get_Book.SetRequestHeaders([{"Name"
: "Accept", "Value": "application/json" }, { "Name": "Content-
Type", "Value": "application/json" }]);
      LibraryInformationSystem_Get_Book.DoExecute({    "session_id":
```

```
sessionId });
```

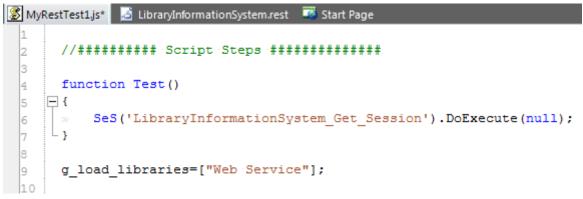
```
Tester.AssertEqual("Book Count", <mark>15</mark>,
LibraryInformationSystem_Get_Book.GetResponseBodyObject().length);
```

3b. Writing REST Test Scripts

Open up the main **MyRestTest1.js** file in the Rapise editor. It will initially consist of a single empty function Test():



The first task is to get a new SessionId from the server using the **Get_Session** operation. To do this, drag the **"DoExecute"** operation from under the **"Get_Session"** object into the script editor, in between the opening and closing braces of the Test() function:



This will execute the web serviced and return the SessionId. To actually access the retrieved value, you need to drag the **"GetResponseBodyObject"** property to the script editor, under the previous line. Then add the JavaScript code var sessionId = to actually store the value. We will also add a Tester.Message(sessionId); line afterwards to write out the value of the sessionId to the test report. This will help us make sure we are getting back a valid response from the web service. You should now have the following code:

```
function Test()

{
    SeS('LibraryInformationSystem_Get_Session').DoExecute(null);
    var sessionId = SeS('LibraryInformationSystem_Get_Session').GetResponseBodyObject();
    Tester.Message(sessionId);
}
```

```
g_load_libraries=["Web Service"];
```

Save this test and click "Play" to execute the test. You should now see a report similar to the following:

#	Name	Start	Туре	Status	Comment	Iteration
		=				=
	Starting scenario: Test	13:37:16.020	Message	Info		
	Get_Session.DoExecute([null])	13:37:17.486	Assert	Pass	Returned Value: true	0
	d51f97ea-d879-4eb1-b585-55469b88cef7	13:37:17.486	Message	Info		0
• •	MyRestTest1	13:37:17.486	Test	Pass	Passed:1 Failed:0	

Now we need to add the code to get the list of books. To do that, simply drag the **"DoExecute"** operation from under the **"Get_Books"** object into the script editor. Then change the (null) argument to instead provide the session id as a Javascript dictionary:

SeS('LibraryInformationSystem Get Books').DoExecute({

To get the list of books as a JavaScript array, drag the **"GetResponseBodyObject"** property to the script editor, under the previous line. Then assign the value of this property to a variable such as "books":

var books = SeS('LibraryInformationSystem_Get_Books')

Now we can add code to test that the number of books returned matches the expected value. Type in the following code:

Tester.AssertEqual('Book count matches', 14, books.le

You should now have the following code:

```
function Test()
{
    SeS('LibraryInformationSystem_Get_Session').DoExecute(null);
    var sessionId = SeS('LibraryInformationSystem_Get_Session').GetResponseBodyObject();
    Tester.Message(sessionId);
    SeS('LibraryInformationSystem_Get_Books').DoExecute({"session_id":sessionId});
    var books = SeS('LibraryInformationSystem_Get_Books').GetResponseBodyObject();
    Tester.AssertEqual('Book count matches', 14, books.length);
}
```

g_load_libraries=["Web Service"];

Finally we need to add the code to add a new book to the system. To do that, simply drag the **"DoExecute"** operation from under the **"Add_Book"** object into the script editor. Then change the (null) argument to instead provide the session id as a Javascript dictionary:

SeS('LibraryInformationSystem_Add_Book').DoExecute({"

To provide the data for a new book, we will need to drag the "SetRequestBodyObject" property of the "Add_Book" object to the line above the DoExecute and pass in a populated JavaScript object:

```
var newBook = {};
newBook.Name = 'A Christmas Carol';
newBook.AuthorId = 2;
newBook.GenreId = 3;
```

SeS('LibraryInformationSystem_Add_Book').SetRequestBo

Finally Add code to test that our new book was added correctly and the count has increased by one:

SeS('LibraryInformationSystem_Get_Books').DoExecute({"se books = SeS('LibraryInformationSystem_Get_Books').GetRes Tester.AssertEqual('Book count matches', 15, books.lengt)

You should now have the following code:

```
function Test()
- {
    Ses('LibraryInformationSystem Get Session').DoExecute(null);
    var sessionId = SeS('LibraryInformationSystem Get Session').GetResponseBodyObject();
    Tester.Message(sessionId);
    SeS('LibraryInformationSystem Get Books').DoExecute({"session id":sessionId});
    var books = SeS('LibraryInformationSystem Get Books').GetResponseBodyObject();
    Tester.AssertEqual('Book count matches', 14, books.length);
    var newBook = {};
    newBook.Name '= 'A Christmas Carol';
    newBook.AuthorId = 2;
    newBook.GenreId = 3:
    SeS('LibraryInformationSystem Add Book').SetRequestBodyObject(newBook);
    Ses('LibraryInformationSystem Add Book').DoExecute({"session id":sessionId});
    Ses('LibraryInformationSystem Get Books').DoExecute({"session id":sessionId});
    books = SeS('LibraryInformationSystem Get Books').GetResponseBodyObject();
    Tester.AssertEqual('Book count matches', 15, books.length);
- }
```

Save this test and click "Play" to execute the test. You should now see a report similar to the following:

#	Name	Start	Туре	Status	Comment	Iteration
	Δ	=				=
	Starting scenario: Test	14:49:03.725	Message	Info		
	Get_Session.DoExecute([null])	14:49:04.334	Assert	Pass	Returned Value: true	0
	c3d8dcd4-6125-427d-939a-0dd181b3cce1	14:49:04.334	Message	Info		0
	Get_Books.DoExecute([{"session_id":"c3d8dcd4-6125-4	14:49:05.051	Assert	Pass	Returned Value: true	0
	Book count matches	14:49:05.051	Assert	Pass		0
	Add_Book.DoExecute([{"session_id":"c3d8dcd4-6125-4	14:49:05.379	Assert	Pass	Returned Value: true	0
	Get_Books.DoExecute([{"session_id":"c3d8dcd4-6125-4	14:49:05.597	Assert	Pass	Returned Value: true	0
	Book count matches	14:49:05.597	Assert	Pass		0
•	MyRestTest1	14:49:05.597	Test	Pass	Passed:6 Failed:0	

Congratulations! You have just created your first test script that tests a RESTful web service.

2.2.6 Tutorial: SOAP Web Services

In this section you shall learn how to test a SOAP web services API using Rapise. We shall be using a demo application called **Library Information System** that has a dummy SOAP web service API available for learning purposes. You can access this sample application at http://www.libraryinformationsystem.org, and its SOAP web service API can be found at: www.libraryinformationsystem.org/Services/SoapService.aspx.

What is SOAP and what is a SOAP web service?

SOAP is the Simple Object Access Protocol, and allows you to make API calls over HTTP/HTTPS using specially formatted XML. SOAP web servicesmake use of the Web Service Definition Language (WDSL) and communicate using HTTP POST requests. They are essentially a serialization of RPC object calls into XML that can then be passed to the web service. The XML passed to the SOAP web

services needs to match the format specified in the WSDL.

SOAP web services are fully self-descripting, so most clients do not directly work with the SOAP XML language, but instead use a client-side proxy generator that creates client object representations of the web service (e.g. Java, .NET objects). The web service consumers interact with these language-specific representations of the SOAP web service. However when these SOAP calls fail you need a way of testing them that includes being able to inspect the raw SOAP XML that is actually being sent.

Overview

Creating a SOAP web service test in Rapise consists of the following steps:

- 1. Using the SOAP web services studio to inspect the SOAP WSDL
- 2. Invoke the various SOAP operations and verify that they return the expected data in the expected format.
- 3. Generating the test script in JavaScript that uses the learned Rapise web service objects based on the WSDL.

We shall discuss each of these steps in turn.

1. Inspecting the SOAP WSDL Endpoint

Create a new test in Rapise called MySoapTest1.sstest. For methodology, choose **Basic: Standard Scripting Mode** and Rapise will create a new blank test project.

Once you have created it, click on the "Web Services" icon in the Test ribbon to add a new web service definition to your test project:

🖾 Start Page	📀 Spira Dashboard
💰 Test	🕲 Manual Steps
Functions	👍 Web Services
S	hortcuts

This will display the Add New Web Service dialog box:

🔄 Add Nev	w Web Service		\times
		This wizard will create a new web service definition file inside the current test:	
OREST	● SOAP		
LibraryInfo	rmationSystem	soap	
C:\Users\a	idam.sandman\[Documents\My Rapise Tests\MySoapTest1	
		Create Cancel	

Choose **SOAP** as the type of web service you want to create.

Then, enter the name of the web service that you're going to add, in this case enter "LibraryInformationSystem.soap" and click "Create".

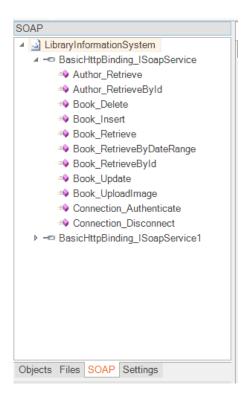
This will add the SOAP web services definition file to your test project:

File Test Options SOAP			
http://www.libraryinformationsyste	em.org/Services Get WSDL		🛅 Create Script 🛛 🌳
Custom Endpoint			👗 Remove Step 🛛 🖡
Save			X Clean Credentials
File Endpoi	nt	Script Steps	HTTP
SOAP	Start Page 📓 MyScapTest1.js 🔤 LibraryInformationSy	/stem.soa	
LibraryInformationSystem	Invoke Request/Response		
	land.	Value	
	Input	value	
	Output	Value Invoke	
	_		
Objects Files SOAP Settings			
Properties			
21 21 🖂			
	Response Body		
	Auto Raw XML JSON		
	1 var LibraryInformationSystem	_Get_Session=SeS('LibraryInformationSystem	
		Session.SetRequestHeaders([{"Name":"Accep	t","Value":"application
	3 LibraryInformationSystem_Get	_Session.DoExecute();	
	4 5 var LibraryInformationSystem	Get Book=SeS('LibraryInformationSystem Ge	Book!):
	6 LibraryInformationSystem Get	Book.SetRequestHeaders([{"Name":"Accept",	"Value": "application/is
	7 LibraryInformationSystem_Get		
	<		
	Response Body Response Headers Output Warnings Erro	ors Find Results	

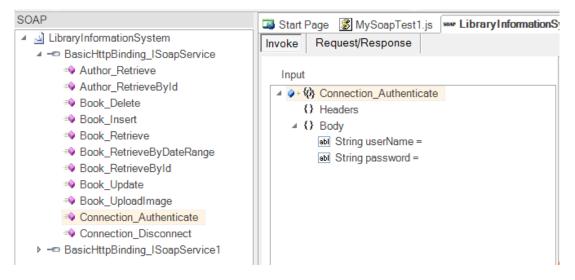
In the **Endpoint** section of the <u>SOAP ribbon</u>, enter the following URL to the sample application's WSDL file:

o http://www.libraryinformationsystem.org/Services/SoapService.svc?wsdl

then click the Get WSDL to load the list of SOAP operations:



Now click on the Connection_Authenticate operation in the SOAP explorer:



This is the first operation we will need to invoke since it is used to authenticate with the online library system before calling the other functions.

You can click on each of the different SOAP operations (e.g. for inserting, retrieving, deleting or updating a book) and the SOAP studio will display the expected input and output parameters as well as any headers.

In the next section we shall be performing the following actions:

- o Authenticating as a specific user
- o Viewing the list of books

- o Inserting a new book
- Viewing the updated list of books
- o Disconnecting

Each one will involve calling a specific SOAP operation with some input parameters, viewing the data returned and adding a verification step if appropriate.

2. Invoking the SOAP Actions

Starting with the **Connection_Authenticate** operation that we had selected, click on the two Input parameters in turn and enter values:

 \circ userName = librarian

o password = librarian

Then click the Invoke button underneath:

Start Page 📓 MySoapTest1.js 👐 LibraryInformationS	ystem.soa	
Invoke Request/Response		
Input ▲ ↔ ۞ Connection_Authenticate ○ Headers ▲ ○ Body event String userName = librarian event String password = librarian	Value Value IsNull Type	librarian False System.String
Output	Value Invoke Record	
♦ ₩ Connection_Authenticate	Value	True
() Headers	Туре	System.Boolean
A {} Body		
Boolean Connection_AuthenticateResult = True		
Boolean Connection_AuthenticateResultSpecifi		

You can see that the response to our Invoked operation as a simple boolean value of **True** returned. That indicated that we authenticated correctly. If you try putting in an incorrect login/password, you'll get back **False** instead.

If you have a SOAP web service that doesn't behave as expected, you may want to view the raw SOAP XML that is being sent to/from the web service. To view this, click on the **Request/Response** tab of the SOAP studio editor and the following will be displayed:

roke Request/Response	ry/n/connetionSystem see	9		
RequestPropr	Request			
AllowAutoRedire False	Auto Raw XML JSON			
AllowWriteStrear False	<pre><?xml version="1.0" encoding="utf-8"?></pre>			
BasicAuthPassw		and the second sec		
BasicAuthUserN	<pre>2 <soap:envelope <="" td="" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/</pre></td><td>/XMLSchema-instance"></soap:envelope></pre>			
ContentType text/xml; charset-utf-				
HttpPraxy	Connection_Authenticate xmlns="http://www.inflectra.com/LibraryInformationSystem/Services/">			
KeepAlive False	s <username>librarian</username>			
Method POST	<pre>6 <password>librarian</password></pre>			
Pipelined False	<pre>//Connection Authenticate></pre>			
PreAuthenticate False		,		
SendChunked False	4	>		
SOAPAction "http://www.inflectra.c	C Barrense			
Timeout 100000	Response	Send		
Url http://www.libraryinfo	of ResponseCode: 200 (OK)			
UseCookieConta True UseDefaultCrede False	Content-Length 296			
	Server:Microsoft-II/5/8.0 X-AspNet-Version:4.0.30319 X-Powerde-By-ASP.NET			
	<pre><s:envelope xmlns.s="http://schemas.xmlsoap.org/soap/envelope/"><s:body><connection_authenticateresponse "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns="http://www.inflectra
<</pre></td><td>a.com/LibraryInformationSyst</td></tr><tr><td>sponse Body</td><td><s: Envelope xmlns:s="><s:body><connection_authenticateresponse "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns="http://www.inflectra</td><td>a.com/LibraryInformationSyst</td></tr><tr><td></td><td><s: Envelope xmlns:s="><s:body><connection_authenticateresponse "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns="http://www.inflectra</td><td>a.com/LibraryInformationSyst</td></tr><tr><td>uto Raw XML JSON</td><td><s:Envelope xmlns:s="><s:body><connection_authenticateresponse 1.0"="" encomposition<="" pre="" xmlns="http://www.inflectra</td><td>a.com/LibraryInformationSyst</td></tr><tr><td><pre>vuto Raw XML JSON <?xml version="></connection_authenticateresponse></s:body></connection_authenticateresponse></s:body></connection_authenticateresponse></s:body></connection_authenticateresponse></s:body></s:envelope></pre>	<pre><s envelope="" s="http://schemas.xmlsoap.org/soap/envelope/" xmlns=""><s body=""> <connection_authenticateresponse 1.0"="" <="" enc="<s:Envelope" pre="" xmlns="http://www.inflectra
c</pre></td><td>a.com/LibraryInformationSyst</td></tr><tr><td><pre>wto Raw XML JSON <7xml version=" xmlnsis="ht"></connection_authenticateresponse></s></s></pre>	<s:envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:body><connection_authenticateresponse 1.0"="" enc<br="" xmlns="http://www.inflectra</td><td>s.com/LibraryInformationSyst</td></tr><tr><td>wto Raw XML JSON
<7xml version="><s:envelope "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns:s="ht
<s:Eody></td><td><pre>csEnvelope xmlns s="><seody><connection_authenticateresponse utf='8""?' xmlns="http://www.inflectra
c
boding="> tp://schemas.xmlsoap.org/soap/envelope/"></connection_authenticateresponse></seody></s:envelope></connection_authenticateresponse></s:body></s:envelope>	a.com/LibraryInformationSyst
uto Raw XML JSON <7xml version="1.0" enc <s:envelope "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns:s="ht
<s:Eody>
<Connection Authent</td><td><pre>cs:Envelope xmlns s="><s:body><connection_authenticateresponse utf='8"?' xmlns="http://www.inflectra
c
boding="> ttp://schemas.xmlsoap.org/soap/envelope/"> ttp://schemas.xmlsoap.org/soap/envelope/"> ttp://schemas.xmlsoap.org/soap/envelope/"></connection_authenticateresponse></s:body></s:envelope>	a.com/LibraryInformationSyste			
uto Raw XML JSON <7xml version="1.0" enc <s:envelope "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns:s="ht
<s:Eody>
<Connection Authent</td><td><pre>csEnvelope xmlns s="><seody><connection_authenticateresponse utf='8""?' xmlns="http://www.inflectra
c
boding="> tp://schemas.xmlsoap.org/soap/envelope/"></connection_authenticateresponse></seody></s:envelope>	a.com/LibraryInformationSyst			
uto Raw XML JSON <7xml version="1.0" enc <s:envelope "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns:s="ht
<s:Eody>
<Connection Authent</td><td><pre>cs Envelope xmlms s="><ss body=""><connection_authenticateresponse utf='8"?' xmlms="http://www.inflectra
coding="> ctp://schemas.xmlsoap.org/soap/envelope/"> ctp://schemas.xmlsoap.org/soap/envelope/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"></connection_authenticateresponse></ss></s:envelope>	a.com/LibraryInformationSyst			
uto Raw XML JSON <rr> <rxml eno<br="" version="1.0"><rs:envelope "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlns:s="ht
<rs:Eody>
</r> <rs:Eody>
</r> <connection_Authent
</connection_Authent</td></td><td><pre>cs Envelope xmlms s="><ss body=""><connection_authenticateresponse utf='8"?' xmlms="http://www.inflectra
coding="> ctp://schemas.xmlsoap.org/soap/envelope/"> ctp://schemas.xmlsoap.org/soap/envelope/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"></connection_authenticateresponse></ss></rs:envelope></rxml></rr>	a.com/LibraryInformationSyst			
uto Raw XML JSON <7xml version="1.0" enc <s:envelope "="" envelope="" http:="" schemas.xmlsoap.org="" soap="" xmlnsts="ht
<s:Eody>
<connection_Authent.
<Connection_Authent</td><td><pre>cs Envelope xmlms s="><ss body=""><connection_authenticateresponse utf='8"?' xmlms="http://www.inflectra
coding="> ctp://schemas.xmlsoap.org/soap/envelope/"> ctp://schemas.xmlsoap.org/soap/envelope/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"></connection_authenticateresponse></ss></s:envelope>	a.com/LibraryInformationSyst			
who Raw XML JSON <pre> </pre> <pre></pre>	<pre>csEnvelope xmlms s="http://schemas.xmlsoap.org/soap/envelope/"><seody><connection_authenticateresponse utf='8"?' xmlms="http://www.inflectra
coding="> ctp://schemas.xmlsoap.org/soap/envelope/"> ctp://schemas.xmlsoap.org/soap/envelope/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"> ctoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"> ttoicateResponse xmlms="http://www.inflectra.com/LibraryInformationSystem/Services/"></connection_authenticateresponse></seody></pre>	a.com/LibraryInformationSyst		

This view lets you see the Request and Response HTTP headers body, with the body displayed in a friendly, easy to read color-coded XML format. That way you can easily invoke the SOAP operations using the Rapise SOAP studio GUI and view the raw SOAP XML being sent to/from the server. This is invaluable when debugging a failing SOAP web service.

In the case of our test of **Connection_Authenticate**, we can now click the **Record** button (next to Send) to add this operation to our list of recorded test steps:

Connection_Authenticate {"userName":"librarian","password":"librarian"}	🛅 Create Script
	👗 Remove Step
	🗙 Clean
Script Steps	

Once you have added the operation to the list of recorded steps, you can go one step further and ask Rapise to verify the data returned. To do that, click on the **Verify** button that is displayed next to the **Record** button. The step will now switch to **bold** to indicate that a verification step is also included.

Connection_Authenticate {"userName":"librarian","password":"librarian"}	🛅 Create Script
	🐰 Remove Step
	🗙 Clean
Script Steps	

Now we need to repeat this process for the following additional operations:

- o Book_Retrieve
 - No Input Parameters
 - Press Invoke to test the retrieve
 - Press Record to record the test script

- Click Verify to add a verification step
- o Book_Insert
 - Populate the Book input object with these values:
 - AuthorId = 2
 - Genreld = 3
 - Name = 'A Christmas Carol'
 - DateAdded = (pick a date using the date picker)
 - DateAddedIso = 2017-01-04T07:46:36
 - Press Invoke to test the insert
 - Press Record to record the test script

DateTime DateAdded = 1/1/0001 12:0		Value			A Christmas Carol
Boolean DateAddedSpecified = False		IsNull			False
String DateAddedIso =		Туре			System.String
Alternative Genre Genre					
12 Int32 Genreld = 3					
Boolean GenreldSpecified = False					
12 Int32 Id = 0					
Boolean IdSpecified = False					
Boolean IsOutOfPrint = False					
Boolean IsOutOfPrintSpecified = Fals					
Boll String Name = A Christmas Carol					
	-	ļ			

o Book_Retrieve

- No Input Parameters
- Press Invoke to test the retrieve
- Press Record to record the test script
- Click Verify to add a verification step
- Connection_Disconnect
 - No Input Parameters
 - Press Invoke to test the retrieve
 - Press Record to record the test script

Once you have completed all these steps, you will see the following recorded in the Script Steps box:

Book_Insert {"book":{"Author":{"Name":""},"DateAdded":"2017-01-04T07:46:36","	🔺 🛅 Create Script
Book_Retrieve {}	🚽 👗 Remove Step
Connection_Disconnect {}	👻 🗙 Clean
Script Steps	

Now that we have recorded the operations and verifications, we can proceed to generate the test script in Rapise that will regression test the web service.

3. Generating the Rapise Test Script

In the SOAP ribbon, click on the Create Script button to generate the initial test script:

```
Book_Insert {"book":{"Author":{"Name":""},"DateAdded":"2017-01-04T07:46:36","
Book_Retrieve {}
Connection_Disconnect {}
Script Steps
Conscient Steps
```

Click on the Test shortcut in the main test ribbon, and Rapise will display the MySoapTest.js file.

In the main Rapise test script file, you will see the following generated:

```
function Test()
{
      var LibraryInformationSystem=SeS('LibraryInformationSystem');
      LibraryInformationSystem.DoExecute('Connection_Authenticate',
{"userName":"librarian","password":"librarian"});
      Tester.Assert('Connection_Authenticate Response',
LibraryInformationSystem.GetResponseObject(), { "Body":
{ "Connection_AuthenticateResult"
:true, "Connection_AuthenticateResultSpecified":true}, "Headers":{}});
      LibraryInformationSystem.DoExecute('Book_Retrieve', {});
      Tester.Assert('Book_Retrieve Response',
LibraryInformationSystem.GetResponseObject(), {...}]}, "Headers":{}});
      LibraryInformationSystem.DoExecute('Book_Insert', { "book": { "Author":
{"Name":""},"DateAdded":"2017-01-
04T07:46:36", "DateAddedSpecified":true, "DateAddedIso":"2017-01-
04T07:46:36", "Genre": {"Name": "}, "Name": "A Christmas Carol"});
      LibraryInformationSystem.DoExecute('Book_Retrieve', {});
      Tester.Assert('Book_Retrieve Response',
LibraryInformationSystem.GetResponseObject(), {...}, "Headers":{}});
      LibraryInformationSystem.DoExecute('Connection_Disconnect', {});
}
```

You will see each of the SOAP functions called in turn, with verification code automatically added.

We can add some comments to make it easier to read:

```
//Authenticate
var LibraryInformationSystem=SeS('LibraryInformationSystem');
LibraryInformationSystem.DoExecute('Connection_Authenticate',
{"userName":"librarian","password":"librarian"});
Tester.Assert('Connection_Authenticate Response',
LibraryInformationSystem.GetResponseObject(), {"Body":
{"Connection_AuthenticateResult"
:true,"Connection_AuthenticateResultSpecified":true},"Headers":{}});
//Verify the initial list of books
LibraryInformationSystem.DoExecute('Book_Retrieve', {});
Tester.Assert('Book_Retrieve Response',
LibraryInformationSystem.GetResponseObject(), {...}]},"Headers":{}});
LibraryInformationSystem.DoExecute('Book_Insert', {"book":{"Author":
{"Name":""},"DateAdded":"2017-01-
```

When you click the **Play** button in the main test ribbon, you will see the following result:

#	Type	Start		Name	Status		Comment
(A)	A	-	(A)		A	(A)	
	Message	08:46:21.701	Starting scenario	o: Test	Info		
	Assert	08:46:22.539	LibraryInformatio	onSystem.DoExecute(["Connection_Authenticate",{"us	Pass	Returned Value: true	
	Assert	08:46:22.546	Connection_Aut	henticate Response	Pass		
	Assert	08:46:22.918	LibraryInformatio	onSystem.DoExecute(["Book_Retrieve",{}])	Pass	Returned Value: true	
	Assert	08:46:22.926	Book_Retrieve R	lesponse	Pass		
	Assert	08:46:23.135	LibraryInformation	onSystem.DoExecute(["Book_Insert",{"book":{"Author	Pass	Returned Value: true	
	Assert	08:46:23.353	LibraryInformatio	onSystem.DoExecute(["Book_Retrieve",{}])	Pass	Returned Value: true	
	Assert	08:46:23.358	Book_Retrieve R	lesponse	Pass		
	Assert	08:46:23.556	LibraryInformatio	onSystem.DoExecute(["Connection_Disconnect",{}])	Pass	Returned Value: true	
F	Test	08:46:23.561	MySoapTest1		Pass	Passed:8 Failed:0	

Congratulations! You have recorded and executed a SOAP web service test.

2.2.7 Tutorial: Mobile Testing

Purpose

Rapise lets you record and play automated tests against native applications on a variety of mobile devices using either <u>Apple iOS</u> or <u>Android</u>. Rapise gives you the flexibility to test your applications on either real or simulated devices.

This tutorial is a **simple example** of using Rapise to record and playback a simple test against a sample **Android application** running on the **Android Simulator** on your local PC. It does not require any physical mobile devices and only uses the PC that you have already installed Rapise on. (*There is other documentation* that describes the full range of mobile testing options)

1) Setting up Appium and the Android SDK

The first thing you need to do is go to the **Appium** website (<u>http://appium.io</u>) and install the latest version of Appium. Once it is installed, you can start it up and click the Play button to start the Appium server:



Once that is installed, you will then need to install the Android SDK (you may already have it installed if you are doing Android development). You can download it from: <u>https://developer.android.com/sdk</u>.

Once it has installed, you will use the **Android SDK Manager** to download and install the necessary packages:

Packages Tools					
DK Path: C:\Program Files (x86)\Android\android-sdk					
Packages					
👘 Name	API	Rev.	Status		
a 🔲 🦲 Tools					1
Android SDK Tools		24.1.2	😿 Installed		10
Android SDK Platform-tools		22	🔯 Installed		
🔲 🥕 Android SDK Build-tools		22.0.1	🔯 Installed		
🔲 🥕 Android SDK Build-tools		21.1.2	Not installed	ł	
🔲 🥜 Android SDK Build-tools		20	Not installed	ł	
🔲 🥕 Android SDK Build-tools		19.1	Not installed	1	
Android 5.1.1 (API 22)					
Documentation for Android SDK	22	1	🔯 Installed		
🔲 🚔 SDK Platform	22	2	🔯 Installed		
🔄 👗 Samples for SDK	22	5	🔯 Installed		
Android TV ARM EABI v7a System Image	22	1	🔯 Installed		
📰 🌆 Android TV Intel x86 Atom System Image	22	1	🔯 Installed		
🕅 📷 ARM EABI v7a System Image	22	1	😿 Installed		Ŧ
Show: Vpdates/New VIII Installed Select New or Updates				Install packages	
Obsolete Deselect All				Delete packages	
	_				

Make sure you have installed the **Android ARM images** using the SDK manager. Then you can launch (from the Windows Start Menu) the **Android Virtual Device (AVD) Manager**:

ols			
idroid Virtual Dev	vices Device Definitions		
ist of existing An	droid Virtual Devices located a	t C:\Users\adam.sandman\.android\avd	
AVD Name	Target Name	Platfor API Le CPU/ABI	Create
	No AVD available		Start
			Edit
			<u></u>
			Repair
			Delete
			Details
			Refresh

Use the $\ensuremath{\textbf{Create}}$ button to create the following Virtual Device:

AVD Name:	AVD_for_Nexus_7_by_Google		
Device:	Nexus 7 (7.02", 1200 × 1920: xhdpi)		
Target:	Android 5.1.1 - API Level 22		
CPU/ABI:	ARM (armeabi-v7a)		
Keyboard:	📝 Hardware keyboard present		
Skin:	No skin		
Front Camera:	None		
Back Camera:	None		
Memory Options:	RAM: 2048	VM Heap: 64	
Internal Storage:	200		Мів
SD Card:			
	• Size:		MiB 🔻
	() File:		Browse.
Emulation Options:	Snapshot Use Host GPU		
			OK Cancel

You may need to modify the RAM / Heap parameters to match that which is supported by the physical PC that you are using. Once the device has been created:

ools					
Android Virtual Devices	Device Definitions				
List of existing Android	Virtual Devices located at	C:\Users\adam.	sandman\.	android\avd	
AVD Name	Target Name	Platfor	API Le	CPU/ABI	Create
AVD_for_Ne	Android 5.1.1	5.1.1	22	ARM (armeabi-v7a)	Start
					Delete
					Refresh

you can then click Start to start the device and then connect to it using Rapise.

2) Configure the Mobile Profile

To begin the actual mobile testing, <u>create a new test</u>, using the **File > New Test** option in Rapise. Make sure you choose the mobile methodology option "Mobile: Mobile Support":

Create or Open T	est	and the second s		
Server:	http://d	doctor/spirateam/		
Project:				
Library Information	n System			
Test Cases:			New Folder	New Test Case
Ability t Ability t New To Test th	Tests to create new book to edit existing book to create new author test Case (Ability to report tat the system lets you rep to edit existing author			F
Step	Description	Expected Result	Actual F	les⊔it
Step		Expected Result	Actual F	(esuit
Step Mobile: Mobile Sup	oport		Actual F	;esuit
Step	pport		Actual F	iesuit

Once you have entered the name for the new test (with the mobile methodology selected) you will be asked to choose the mobile profile. Rapise ships with several default profiles, for now select the one that is closed to the device you want to test (we recommend the **android** generic profile):

•
ОК

When you click the **[OK]** button, Rapise will ask you to choose the Scripting Language:

R Scripting Language			×
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scripting	g Languag	e)	

Please choose the **Rapise Visual Language (RVL)**. Rapise will then create a new mobile test with the Android Simulator profile selected.

RVL	+Add New	\					
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value
2)		~					
3							
4							
5							
6							
7							
8							
9							
10							

Now you need to modify the profile so that it correctly matches the type of device you are testing and also so that it correctly points to the <u>Appium</u> server that you are using to host the mobile devices. Click on Options > Tools > Mobile Settings to bring up the <u>Mobile Settings</u> dialog box:

elect Profile: android	Save Test Url Duplicate	Create Rename Delete
₽ 2↓ ©		
Connection		
Uri	http://127.0.0.1:4723/wd/hub	ſ
Details		
 Primary Capabilities 	(6)	
арр	C:\Users\Public\Documents\Ra	pise\Samples\UsingMobile\AUT\AUT
browserName		
deviceName	Android Emulator	
platformName	Android	
platformVersion	5.1.1	
udid		
 Common Capabilities 	(9)	
autoLaunch		
automationName		
autoWebview		
deviceName		
The kind of mobile device or emulator to use		
	e Retina 4-inch, Android Emulator, Galaxy S4, etc. On iO	
devices returned by instruments with instrument	s -s devices. On Android this capability is currently ignore	d.

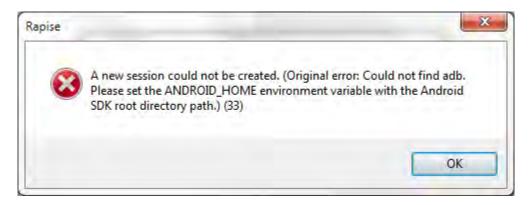
In the mobile profile screen, make sure you change the following:

- **app** this needs to the path to the Application being tested on the device (e.g. C:\Users\Public \Documents\Rapise\Samples\UsingMobile\AUT\AUTAndroid\bin\AUTAndroid.apk). This path should be already correct, but it is worth double-checking
- deviceName this needs to match the name of the device being connected
- platformName this needs to be set to 'Android'
- **platformVersion** this needs to be set to the same version of Android that the virtual device is running (the one specified in the Android Virtual Device screen earlier)

Once you have entered in the information and saved the profile, make sure that Appium is running on the PC and then click the **[Test URL]** button to verify the connection with Appium:

-			-		
0	Successfu http://127	uly conn 7.0.0.1:47	ected 723/w	to: d/hub	
			-	OK	

Now when you try and <u>connect to the device</u> using the <u>Rapise mobile spy</u>, you may get the following message:



This means you need to use the Windows control panel to add a **System environment variable** called **ANDROID_HOME** and set it to the path of the installed Android SDK (typically C:\Program Files (x86)\Android\android-sdk).

Once you have configured the ANDROID_HOME and it connects, you can start testing your mobile Android application.

3) Using the Mobile Spy

The Mobile Spy will let you view an application running on the mobile device, take a snapshot of its screen and then interactively inspect the objects in the application being tested. This is a useful first step to make sure that Rapise recognizes the application and has access to the objects in the user interface.

To start the Mobile Spy, open the Spy icon on the main <u>Test ribbon</u> and select the Mobile option and the Mobile Spy will be displayed in **Discovery Mode**. Now click the **[Get Snapshot]** button to display the application specified in the <u>mobile profile</u> on the screen:

	elect Profile android-nexus7	Discovery Mode	
Get Snapshot	Edit Profiles	Select Context NATIVE_	APP +
(Ctrl-G) Test Locator Connection Device	Profile	Context	
(0) android widget FrameLayout[0.0][800,1205] ▲ (0) android widget.View[0.0][800,1205] ▲ (0) android widget.InearLayout[0.33][800,108] ▲ (0) android widget.FrameLayout[0.33][800,108] ▲ (0) android widget.InearLayout[1.33][280, ▶ (0) android widget.InearLayout[1.13][228, ▶ (1) android widget.InearLayout[715,33][80 ▲ (0) android widget.InearLayout[0,108][800,1205] ▲ (0) android widget.InearLayout[1,108][800,1205] ▲ (0) android widget.InearLayout[1,108][800,1205] ▲ (0) android widget.InearLayout[21,129][779,45 ▶ (0) android widget.ListView[21,129][779,45	Calculated Automation Id IndexInParent TagName Text TagName Text Title XPath Varbout Height Width X Y Properties checkable checkable checkable foccusable foccusable foccusable foccusable foccusable scrollable selected	0 android.widget.FrameLayout (0) android.widget.FrameLayout[0,0][//android.widget.FrameLayout[1] 1205 800 0 False false false false false false false false false false false false false false false false	AUTAndroid Login DropDownSelect Buttons Bars Views

In the example above, we are displaying the sample Android application that comes with Rapise (AUTAndroid).

If you click on one objects in the user interface, it will be highlighted in Red and the tree hierarchy on

Disconnect	Page Source	Sele	ct Profile android-nex	cus7 👻	Discovery Mode		
Get Snapshot (Ctrl-G)	🔍 Test Locator	🛃 Edit	Profiles		Select Context NATIVE	APP -	
Connection	Device	_	Profile		Context		
android widget FrameLayout[0,0][800,1205] (0) android view. View[0,0][800,1205] (0) android widget LinearLayout[0,33][800,108] (0) android view. View[0,33][800,108] (0) android view. View[0,33][800,108] (0) android view. View[0,33][800,108] (1) android widget LinearLayout[11,33][828,108] (1) android widget LinearLayout[715,33][800,108] (1) android widget LinearLayout[715,33][800,108] (0) android widget LinearLayout[715,33][800,108] (1) android widget LinearLayout[715,33][800,108] (2) android widget LinearLayout[715,33][800,108] (3) android widget LinearLayout[715,33][800,108] (3) android widget LinearLayout[715,33][800,108] (3) android widget LinearLayout[715,33][800,108] (3) android widget LinearLayout[715,33][800,108] (4) android widget LinearLayout[715,33][800,108]			Calculated Automation Id IndexInParent	android id.	Aext 1	a a a a a a a a a a a a a a a a a a a	₹ ∎24
			TagName	android wi	idget.TextView	Login	
			8] Title (nSelect d.widget.TextView[21,194][DropDownSelect	
		t[0,108][800,1205] / Layout		//android	.widget.FrameLayout[1]/and	Buttons	
				64		Bars	
			Width	758	Dais		
(0) android.widge	t.TextView[21,129][779	,193]	х	21	Views	Views	
	t.TextView[21,194][779		Y	194			
	t.TextView[21,259][779		Properties				
	t.TextView[21,324][779		checkable	false			
(4) and out widge			clickable	true			
			enabled	true			
			focusable	false			
			focused	false			
			scrollable selected	false false			

the left will expand to show the properties of that object:

If you want to view the contents of the Spy as a text file, just click the **'Page Source'** button and you will see the contents of the Spy properties window as a text file.

If you want to perform an action on the application (e.g. click on the selected item), switch the ribbon to the 'Events' view:

Spy	Events			
Тар	Precise Tap 💌	Text		Accept Alert
Swipe 🔻	Scroll To	Send Keys		Dismiss Alert
Shake		Execute Script		Change Orientation
	Touch		Text	Misc

Now click on '**Tap**' and Rapise will send a tap event to the application, switching the application to the next page:

Tap Precise Tap	▼ Text			Accept Alert		
Swipe 👻 Scroll To	Send Keys			Dismiss Alert		
Shake	Execute Script			Change Orientation		
Touch		Text		Misc		
android.widget.FrameLa						₩ 2 :
(0) android.view.View[0,	,0][800,1205]	Automation			(Rapise DropDownSelectActivity	4
		ld IndexInParent	0			
		TagName		idget.FrameLayout	Select Item	
		Text	7 Y	-3	Mercury	
		Title	(0) android	.widget.FrameLayout[0,0][Mercury	
		XPath	//android	widget.FrameLayout[1]		
		⊿ Layout		and the second second		
		Height	1205			
		Width	800			
		Ŷ	0 0			
			ų			
		checkable	false			
		checked	false			
		clickable	false			
		enabled	true			
		focusable	false			
		focused scrollable	false false			
		selected	false			
		text	Talac			

Assuming that you can see your application in the Spy and that the objects can be inspected (similar to that shown above) you can now begin the process of testing your mobile application. Click on **Disconnect** to end your Spy session and close the Rapise Spy dialog. You will now will be returned back to your test script.

4) Recording and Playing a Test

With the new Rapise mobile test script open, click on the **Record/Learn** button in Rapise and that will display the <u>recording activity dialog</u>:

Object	Action	Data C	Comment		
Verify (Ctrl+1)	Learn (Ct	rl+2) 🍑 Spy	y(Ctrl+5)	Pick Object	Resume

Now click on the **[Spy]** or **[Pick Object]** button (they both do the same thing for mobile testing) and the Rapise Spy will be displayed in **Recording Mode**:

Disconnect 🚔 🗖 Pa	ge Source 🛛 🦪 Select Profile	android-nexus7 -	Recording M	ode	
Get Snapshot Learn G	To Url 🔐 Edit Profiles		Select Context	NATIVE_APP +	
Connection Device		Profile	0	Context	
ndroid.widget.FrameLayout[0.0][800.1205] 0) android.view.View[(0.0][800.1205] • (0) android.widget.LinearLayout[0.33][800.108] • (1) android.widget.FrameLayout[0.108][800.1205] • (0) android.widget.LinearLayout[0.108][800.1205] • (0) android.widget.LinearLayout[0.1129][775 • (0) android.widget.LinearLayout[0.1129][775 • (0) android.widget.LinearLayout[2.1,229][775 • (0) android.widget.ExtView[21.129] (1) android.widget.TextView[21.129] (2) android.widget.TextView[21.129] (3) android.widget.TextView[21.324] (4) android.widget.TextView[21.389]	(453] Text 453] Title (779.193] XPath (779.258] Layout (779.328] Height (779.388] Width	android Jid/text 1 0 android widget. Tex Login (0) android widget. Te (0) android widget. Fi 64 758 21 129 false false false false false false false false false false false false false	TextView[21,129][AUTAndroid Login DropDownSelect Buttons Bars Views	i a v :

We now want to record a click on one of the menu options, simply highlight one of the menu entries (e.g. "Login"):

Disconnect	Page Source	Select Profile	android-nexus7 -	Recording M	ode	
Get Snapshot (Ctrl-G) Connection	Learn Go To Url Object Test Locator Device	Edit Profiles	Profile	Select Context	NATIVE_APP	
(1) android.widge(2) android.widge(3) android.widge	0,33][800,108] 0,108][800,1205] put[0,108][800,1205] Layout[21,129][779,453]	Calculated Automation Id IndexInParent TagName Text Tritle XPath Layout Height Width X Y Properties checkable checkable checkable enabled focuaedle scrollable aelected text	android id/text 1 0 android widget. Tex Login (0) android widget. Fi (/android widget. Fi 64 758 21 129 false false false false false false false false false false false false false	TextView[21,129][AurAndroid AurAndroid Login DropDownSelect Buttons Bars Views	

Now click the [Learn Object] button and the object will be added to the Rapise object tree.

Now select the **Events** ribbon and click the **[Tap]** menu entry to move the sample app to the next screen; Rapise will automatically reload the page in the Mobile Spy to get the updated screen:

Spy Events						
p Precise Tap + vipe + Scroll To ake Touch	Text Send Keys Execute Script	Text		Accept Alert Dismiss Alert Change Orientation Misc		
droid.widget.FrameLayo)) android.view.View(0.0] (0) android.widget.Line (1) android.widget.Fra (0) android.widget. (0) android.widget (1) android.wid (2) android.wid (4) android.wid (4) android.wid			1 android (1) andr	ectra autandroid id/edit_usen widget. Edit Text aid widget. Edit Text[21, 158][7 id widget FrameLayout[1]/ank		

Now click on some of the objects and choose **Learn** to add them to the <u>object tree</u>. Once you are finished, click on the **Disconnect** button. You will see the events in the recording activity dialog:

#	Object	Action	Data	Comment
2	lext1	Leam	Login	Learned text1
13	edit_usemame	Leam		Learned edit_usemame
14	edit_password	Leam		Learned edit_password
25	android.widg	Leam	Login	Learned android.widget.Button
06	home	Leam		Leamed home
Ven	n official	Learn (O		Pick Object

Now click on the **Finish** button and you will be taken back to the test script with the Android objects listed:

Open Del Row Save to Spira Open Test from Spira File RVL	🖪 Te		Manual Steps Web Services	Record	Spy Play	Level 1 (Minimal) · ?	About Conve Activation Spira Object Mgr Tools	rt to NeoLoad Properties	Find in He Files Help	
bject Tree	Star	t Page x	Spira Dashboard	x	Mobile_Test.rvl.xk	s x				
√ 46 n KE		+Add New		- 1						
a 💟 Object Tree C (Users)adam sandman(Docum		Flow	Туре		Object	Action	ParamName	Pa	ramType	ParamValue
Device Device		Flow	Туре	Objec	t	Action	Param Name	Param T	ype	Param Value
			~							
📓 home [home]	4									
A text1 [text1] Global	5									
4 User Functions	6									
🙄 User Vanables	7									
	8									
	9									
	10									
	11 12									
	13									
	14									
	15	1								
	16							×		
	17									
Diject True Files Settings	18									

Now that we have the objects, we can simply use the Rapise Visual Language (RVL) test editor to choose:

- Type = Action
- Object = <<u>Name of Object></u>
- Action = DoClick, DoSetText, DoAction

Each of the different types of object recorded will have their own list of available actions:

RVL \	+Add New	./					
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value
2		Action	A text1	DoClick			
3		Action	🛯 edit_username	DoSetText	value	string	test user
4		Action	🛯 edit_password	DoSetText	value	string	test pwd
5		Action	android_widget_B	. DoClick			
6		Action	🔊 home	DoAction			
7							

This will click on the first menu entry, then enter a username and password and then finally return back to the main menu.

Now to playback the test simply click **Play** in the Rapise test ribbon and the test will play back in the mobile device:

#	Туре	Start	Name	Status	Comment	Iteration
*		-				-
	Message	23:50:41.088	Starting scenario: Test	Info		
	Assert	23:50:56.250	text1.DoClick([])	Pass	Returned Value: true	0
	Assert	23:51:05.477	edit_username.DoSetText(["test user"])	Pass	Returned Value: true	0
	Assert	23:51:14.211	edit_password.DoSetText(["test pwd"])	Pass	Returned Value: true	0
	Assert	23:51:18.253	android.widget.Button.DoClick([])	Pass	Returned Value: true	0
	Assert	23:51:19.129	home.DoAction([])	Pass	Returned Value: true	0
•	Test	23:51:19.131	Android Test 3	Pass	Passed:5 Failed:0	

This is the report of the test being executed.

Example

You can find the Android sample tests and sample Application (called AUTAndroid) in your Rapise installation at the following locations:

Sample Android Tests:

- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AppAndroid (testing a native App)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\WebAndroid (testing a web app)

Sample Application (AUTAndroid)

• C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTAndroid

(we supply the sample application as both a compiled .apk binary and an Android Studio Java project with source code)

See Also

- <u>Mobile Testing</u>, for an overview of mobile testing with sub-sections on testing:
 o using <u>iOS</u>
 - o using <u>Android</u>.
- <u>Mobile Settings Dialog</u> for information on setting up the different mobile profiles for the mobile devices you will be testing
- <u>Mobile Object Spy</u> for information on how Rapise connects to the device and lets you view the objects in the application being tested
- <u>Technologies Mobile Testing</u>, for instructions on preparing your environment for mobile testing, including instructions for installing the necessary prerequisites and configuring the various third-party components that Rapise uses to connect to the device.
 <u>Mobile Testing</u> ios Setup, the stope for acting up Youds and the ios SDK for testing ios devices.

o Mobile Testing: iOS Setup - the steps for setting up Xcode and the iOS SDK for testing iOS devices

2.2.8 Tutorial: Manual Testing

Purpose

Exploratory manual testing is used for situations where you have a **new or changing application** and the user interface is still evolving. Traditional manual testing, where you create a test case ahead of time, define the prescriptive test steps and then assign it to the tester does not make sense in such cases. The solution is to perform **exploratory testing**, where you explore using the application at the same time as creating the test script. The created test script can then be published to your test management system (i.e. <u>SpiraTest</u>) for future regression testing.

Rapise can help **accelerate and optimize** exploratory manual testing. Rapise lets you walk through the application, capturing your interactions as you use it, recording screenshots of the objects and screens you interact with. From this, Rapise will create a fully formed test script ready to use.

Step 1 - Creating a New Test

To start manual testing, simply create your test as normal using the <u>New Test</u> dialog box. Then once the test has been created, click on the "Manual Steps" icon in the Test ribbon and then you will be taken to the <u>Manual Editor</u> with the <u>Manual Test Ribbon</u> Visible:

File	Test N	lanual			
Save Save	Reload Save Local	Add Step Kemove Step	 View in Browser New Incident 	Release Execute Manual	
S	pira	Editor	External Tools	Playback	Recording

The test step list will initially be empty:

🔯 Start Page	🛞 MS-Paint Test TC34.js 🛯 🏙 Manual Steps.rmt		
MS-Paint	Test		
StepId	Description	Expected Result	Sample Data

Step 2 - Recording Some Steps

Now you should open up the application you want to record from. In this example we shall be testing the built-in **Microsoft Paint** application. This is a good candidate for manual testing as a lot of the functionality is hard to test automatically since there is a simple drawing canvas rather than discrete buttons and data elements to test.

To make sure that we have screenshots recorded, whilst keeping the size of the screenshots reasonable, use the following recording options:

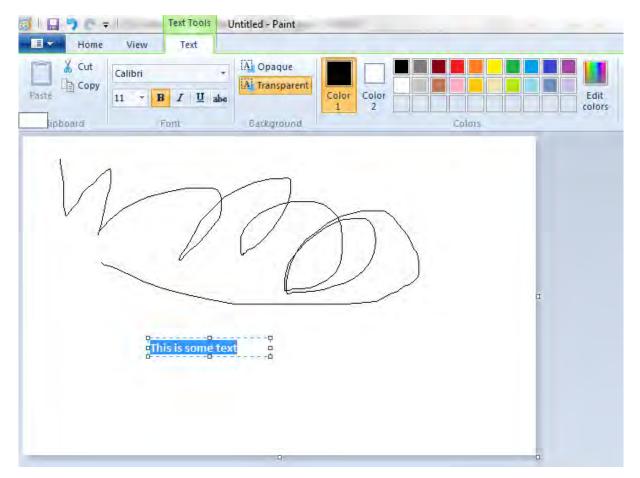
10	1	Record Screenshots
		Whole Window
Record Manual	1	With Cursor
	F	lecording

Now click the '**Record Manual**' button and choose MS-Paint from the list of running applications in <u>Select Application to Record</u> dialog and then click **'Select'** to start recording.

As you click through the application, the recording will display the list of steps and actions being captured:

#	Object	Action	Data	Comment
11	Application	LClick	37,11	User clicks at: 37, 11 in 'Application menu'
212	Application	LClick	42,12	User clicks at: 42, 12 in 'Application menu'
🖄 3 New		LClick	44,13	User clicks at: 44, 13 in 'New'
¥ Pencil		LClick	15,9	User clicks at: 15, 9 in 'Pencil'
🔰 5 Text		LClick	14,16	User clicks at: 14, 16 in 'Text'
26	Simulated	LClick	156,256	User clicks at: 156, 256 in "
17	Text	SetText	This is some	: Change text to 'This is some text'
815	Bold	LClick	11.14	User clicks at: 11, 14 in 'Bold'
Verit	iy (Ctrl+1)	Learn (Ctr	1+2)	Spy(Ctrl+5) Pick Object Resume
Analog (Ctrl+4)			_Simulated	✓ Cancel Finish (Ctrl+3)

In this example, we created a new canvas, chose the Pencil tool, created a drawing using the pencil, entered some text and then made it bold:



When you click Finish to complete the recording, Rapise will now display the list of populated manual

test steps with the embedded screen captures:

MS-Paint	Test		
StepId	Description	Expected Result	Sample Data
Step 1	User clicks at: 37, 11 in 'Application menu'		SeS('Application_menu').DoLClick(37, 11);
Step 2	User clicks at: 42, 12 in 'Application menu'		SeS('Application_menu').DoLClick(42, 12);
Step 3	User clicks at: 44, 13 in 'New'	C Sew	SeS('New').DoL Click(44, 13);
Step 4	User clicks at: 15, 9 in 'Pencil'	<u>M</u>	SeS('Pencil').DoLClick(15, 9);
M Step 5	User clicks at: 14, 16 in 'Text'	A	SeS('Text') DoLClick(14, 16);
M Step 6	User clicks at: 156, 256 in "	3 2900	SeS('Simulated').DoLClick(156, 256);

You will notice that the description of each test step will use the form "User [action] at [coordinates] in '[object name]'" and the expected result will include the screenshot of what the user was doing. In addition, the sample data will contains the equivalent Rapise automation code for reference. This can be useful later if you decide to automate this test.

Step 3 - Editing the Steps

Typically you may want to **add some additional steps** (e.g. we added a line to describe the process of starting up MS Paint), **delete any duplicate/unnecessary steps** and **reword them** so that they make the most sense to the tester. In our example we used the <u>manual editing</u> screen to update the steps as follows:

MS-Paint			
StepId	Description	Expected Result	Sample Data
Step 1	User starts up the MS-Paint Application	The application loads with a blank canvas	
Step 2	User clicks the main 'Application menu'		SeS('Application_menu').DoLClick(42, 12);
Notep 3	User clicks the 'New' entry	L. New	<u>SeS('New').DoLClick(</u> 44, 13);
Step 4	User clicks on 'Pencil'	-R	SeS('Pencil').DoL Click(15, 9);
Step 5	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);
C Step 6	User clicks at: 156, 256 in the canvas	72900	SeS('Simulated').DoLClick(156, 256);

Click **Save** to make sure the updates are all saved locally. Now before you can <u>execute these tests</u>, you will need to Save them to <u>Spira</u> (our web-based test management system).

Step 4 - Saving to Spira

Click on the option to **Save to Spira**, you will be asked to confirm the creation of the document folder in Spira that will hold the test files:

apise will create a new	folder in the selected Spira project to stor	re the test script and associated files:
ocal Repository Path	C:\SpiraRepository\MS-Paint Test TC34	1
Remote Folder:	Root Folder/MS-Paint Test TC34	

Click on '**Create'** and then the manual test will be saved to Spira. You will see that this process adds the unique Spira test step IDs to each step. They are displayed using the format [TS:xxx]. This special token [TS:xxx] can be used in Tester.Assert commands to relate specific verification points with test steps during automated testing.

MS-Paint	est			
StepId	Description	Expected Result	Sample Data	
[TS:47]		15°W		
Step 4	User clicks on 'Pencil'	*	SeS('Pencil').DoLClick(15, 9);	
[TS:48]				
Step 5	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);	1
[TS:49]				
Step 6	User clicks at: 156, 256 in the canvas	Same ashes	SeS('Simulated') DoLClick(156, 256);	
[TS:50]				
Step 7	Enters text 'This is some text '	This is some text	SeS('Text1').DoSetText("This is some	
[TS:51]			text\r");	
Step 8	User clicks on the 'Bold' button	B	SeS('Bold').DoLClick(11, 14);	

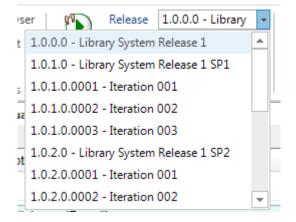
Now that the test has been saved in Spira, you can click on the 'View in Browser' option to see how the test steps look inside Spira.

> <u>Ins</u>	ert S	tep Insert l	<u>_ink Delete Copy Refresh</u>	- Show/hide columns 🔻	Edit Parameters			
	U	Step #	Description	Expected Result	Sample Data	Execution Status	ID	► Edit
		1 Step 1	User starts up the MS- Paint Application	The application loads with a blank canvas		Not Run	TS000045	► Edit
	U	1 Step 2	User clicks the main 'Application menu'		SeS('Application_menu').DoLClick(42, 12);	Not Run	TS000046	► Edit
	U	🖺 <u>Step 3</u>	User clicks the 'New' entry	Sew.	SeS('New').DoLClick(44, 13);	Not Run	TS000047	► Edit
	U	E Step 4	User clicks on 'Pencil'	1	SeS('Pencil').DoLClick(15, 9);	Not Run	TS000048	► Edit
	U	Extep 5	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);	Not Run	TS000049	► Edit
	U	🖺 <u>Step 6</u>	User clicks at: 156, 256 in the canvas		SeS('Simulated').DoLClick(156, 256);	Not Run	TS000050	► Edit
	U	1 Step 7	Enters text 'This is some text '	This is some text	SeS('Text1').DoSetText("This is some text\r");	Not Run	TS000051	► Edit
	U	1 Step 8	User clicks on the 'Bold' button	R	SeS('Bold').DoLClick(11, 14);	Not Run	TS000052	► Edit

Now that we have finished the recording, we can now play back this manual test.

Step 5 - Executing the Manual Test

Choose the Release from the list of those available in the project:



Then click on the '**Execute'** icon to start manual test execution. That will bring up the <u>manual playback</u> screen:

Executing MS-Paint Test		
NS-Paint Test		
Image: Big I marked in the line of		
User starts up the MS-Paint Application		
Expected Result:	Sample Data:	
The application loads with a blank canvas		
Actual Result- Please enter the actual test result if it differs, then click the appropriate but	ton:	
[22] B I U A Arial • A ³ 10 • A • ♥ • ■ ■ = := := := :=		ja,
Add Attachment Log Incident		

On this screen, we shall follow through the steps listed in the test case. This involves opening up MS Paint, creating a new canvas, adding some lines using the pencil and then adding some text using the text tool. As you perform these steps, click on the **Pass** button to indicate that each step has passed. You can also minimize the manual playback screen by clicking the > | button.

Once you get to Step 7, we shall pretend that MS Paint failed to display the text correctly. Enter in the

Actual Result a message to that effect:

Executing MS-Paint Test		
NS-Paint Test		>
Image: Big I and Image: A Arial Ima	8	
 Step 7 of 8 - Please follow the directions outlined in the box below Enters text 'This is some text ' 	N.	
Expected Result:	Sample Data:	
This is some text	SeS('Text1').DoSetText("This is some text\r");	
Actual Result- Please enter the actual test result if it differs, then cli The text did not display correctly.	ck the appropriate button:	
	12 12 60	

Next we shall attach a screenshot of what actually happened and log a test failure and associated incident / defect.

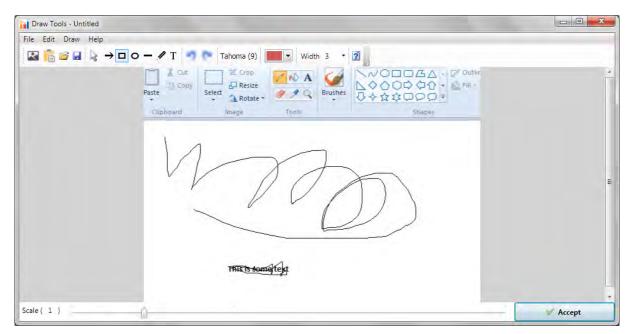
Step 6 - Capturing and Annotating a Screenshot

Click on the **Image icon** in the rich text editor associated with the **Actual Result** text box. That will bring up the <u>Drawing Tools</u> screen that asks you to draw a rectangle to select a portion of the current screen to capture:

Home View Home View Cup Bosid Home View Home V		ize T T T T T T T T T T T T T
MAR	\mathcal{D}	
TRIETS somethed	Select a screen region to	o take snapshot or press ESC to abort
	a	

If the MS Paint application is not in the foreground, just click ESC on your keyboard to abort, rearrange your windows and then try again.

Once you have selected the rectangle, the drawing tools will display your selected image in the image editor:



You can now use the annotation tools to add labels, text and other items to explain the issue that you found:

To Draw Tools - Untitled *	
File Edit Draw Help	
🔛 🔓 🗃 🛃 💊 → 🗆 O — 🖋 T ಶ 🏊 Tahoma (9) 📕 🔹 Width 3 🔹 🕅	
Copy Paste Dipboard Di	
MAD	H
THIE's sometreet	
Scale (1 .)	Second Accept

In the example above, we added a red ellipse, arrow and text to mark the issue that was seen in MS-Paint. Once you are happy with your image, click **Accept** and the image will be included in the test Actual Result:

Executing MS-Paint Test		
MS-Paint Test		>
Image: Step 7 Image: Step 7 Image: Step 7	Step 8	
Enters text 'This is some text '		
Expected Result:	Sample Data:	
This is some text	SeS('Text1').DoSetText("This is some text/r");	
Actual Result- Please enter the actual test result if it differs, the	en click the appropriate button:	
	→ Wrong	
(THEISSONIATE	
[□] B <i>I</i> <u>U</u> A Arial ▼ A ^a 10 ▼ A ▼ ^{ab} ▼		
Add Attachment		

Now we can log an incident that is associated with this test failure.

Step 7 - Logging the Incident / Defect

New Incident *	Concept New York		
🕞 Save			
💿 Details / Descrip	ition		×
Name: The text di	d not display correctly in MS Paint		
Description:			
BIU A Ari	ial 🔹 🗚 🕯 10 🔹 📥 📽 📰 🖼 🖽 🖽 📟		
Type:	Bug *	Status: New	
Detected By:	Fred Bloggs •		
	2 - High	Severity: None	
Priority:	2 rigi	sevency. None	
	1.0.0.0 - Library System Release 1 -	Seventy. None	

Click on the 'Log Incident' button to display the new incident entry screen:

Choose the **type** of incident, enter the **name**, **description**, **priority**, **detected release** and any other required fields as defined by the workflow in the project that you are connected to. Once you have entered in the various fields, click the **'Save'** icon in the top left.

This will return you to the <u>manual execution</u> screen with the **Incident ID** [IN:xxx] and **name** displayed at the bottom. Now click on the '**Fail**' button and the test case will be marked as failed:

114	Rapise User Manual
-----	--------------------

MS-Paint Test		>
E B Z U A Arial • A ³ 10 • A • • · · · · · · · · · · · · · · · ·	3	
Pass V Pass All Blocked O Caution Fail X	+ Finish	
 Step 8 of 8 - Please follow the directions outlined in the bo User clicks on the 'Bold' button 	k below:	
Expected Result:	Sample Data:	
R	SeS('Bold').DoLClick(11, 14);	
Actual Result- Please enter the actual test result if it differs, th	ten click the annronziste button.	
riceau resure ricese enter the actual rest resure in runners, t	ien eine appropriate batton.	
🖾 B Z U A Arial - A ³ 10 - A v 🕬 -		i.
[E] B I U A Arial ▼ Aª 10 ▼ A▼ ♥ ▼ ■ Add Attachment Log Incident		i.

Finally, click on the Finish button and the results will be posted to Spira.

Step 8 - Viewing the Results

Now to view the results in Spira, click on the <u>Spira Dashboard</u> item in the main Rapise <u>Test ribbon</u>. Then under the 'My Created' test cases, click on the link for the test case you execute. That will bring up the test case in Spira. Now click on the 'Failed' hyperlink in Spira and the new test run will be displayed:

Test Run: 🗎 I	VS-Paint Test [TR:000041]			
Overview *	Attachments # Incidents #			
▼ Details				
Release #:	1.0.0.0 - Library System Release 1	Estimated Duration:	hours	
Tester Name:*	Fred Bloggs	Actual Duration:	0.05 hours	
Test Set:		Execution Date:	3/31/2015 1:54:11 PM	
Test Case #:*	TC000034	Execution Status:*	Failed	
Build:	None 🔻	Test Run Type:*	Manual	
Web Browser:	Please Select T	Operating System:	Please Select	▼
Notes:	Font			

If you scroll down, you can see the individual test steps that were executed, with the associated actual result (including the captured screenshot):

D	Test Step Description	Expected Result	Sample Data	Test #/ Step #	Actual Result	Execution Status
RS000084	User starts up the MS- Paint Application	The application loads with a blank canvas		TC000034 / TS000045		Passed
RS000085	User clicks the main 'Application menu'	10 ×	SeS ('Application_menu').DoLClick (42, 12);	TC000034 / TS000046		Passed
RS000086	User clicks the 'New' entry		SeS('New').DoLClick(44, 13);	TC000034 / TS000047		Passed
R\$000087	User clicks on 'Pencil'	2N	SeS('Pencil').DoLClick(15, 9);	TC000034 / TS000048		Passed
RS000088	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);	TC000034 / TS000049		Passed
RS000089	User clicks at: 156, 256 in the canvas		SeS('Simulated').DoLClick (156, 256);	TC000034 / TS000050		Passed
R5000000	Enters text This is some text '	This is some text	Se6(Text1).DoSetText("This is some text?");	TC000034 / TS000051	Falled with the text being illegible.	Falled
R\$000091	User clicks on the 'Bold' button	ę.	SeS('Bold').DoLClick(11, 14);	TC000034 / TS000052		Not Run

If you click on the **Incidents** tab, you can also see the new incident that was logged, linked to this test run:

Overview *	Attachments *	Incidents *								
Display List of Incid	Display List of Incidents: > Refresh Apply Filter Clear Filter - Show/hide columns -									
Displaying 1 - 1 out o	f 1 incident(s) linked to t	this test run. Filterir	ng results by Test F	Run #. <u>(Clear Filte</u>	rs)					
🗸 🕘 Name 🔺 🖲		Type ▲▼	Status ▲▼	Priority ▲▼	Detected By ▲▼	Creation Date ▲▼	Owner ▲▼	Progress	ID ∆ ▼	Edit
		Any 🔻	Any 🔻	Any 🔻	Any 🔻		Any 🔻	Any 🔻		► Edit
Did not	display the correct text	Incident	New	2 - High	Fred Bloggs	31-Mar-2015			IN:000071	► Edit
Show 15 V rows	per page							🛏 ৰ Displaying	page 1	of 1 ► ►

Congratulations! You have now successfully executed a manual test using Rapise.

See Also

- Manual Testing
- Manual Recording
- Manual Playback

2.2.9 Tutorial: Java Testing

In this section, you will learn how to record and execute a Rapise script against a Java applications. We will show you have to test the following three different types of Java application:

- o Java AWT Apps
- o Java Swing Apps
- o Java SWT Apps

This version of the tutorial uses the **Rapise Visual Language (RVL) scriptless** mode. If you're interested in the <u>JavaScript version</u>, we have a separate tutorial.

Example 1 - Launching the Sample AWT/Swing Application

On the <u>Start Page</u> of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

Then go to C:\Users\Public\Documents\Rapise\Samples\Java\AUTJAVA and right-click on the x86run.cmd file and choose Run as Administrator.

If you have Java configured correctly, you will see:

🛓 AUTJAVA		- 🗆	×
AWT label1			
AWT TextArea	button1 checkbox1 textField2	AWT ListItem AWT ListItem AWT ListItem AWT ListItem AWT ListItem	2 3 4
	Choice1 ~		
open dialog Advanced	AWT Controls		
Swing Item jButton2 Item jToggleButt jToggleButt jCheck) jRadioButton1) jRadioButton2 tab: Item 1 v jTextField1	2 3 4 1 tab2 A	Tree colors sports food dvanced Swing	
Swing TextArea Second Line			~

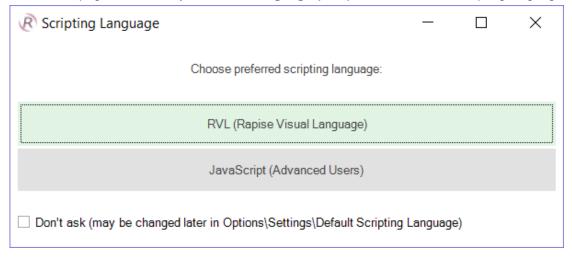
If the application doesn't start correctly, make sure you have Java SE and the <u>Rapise Java Bridge</u> installed and the JAVA_HOME environment variable correctly set to your Java Runtime (JRE). For more details on this, please refer to: <u>Java AWT/Swing Testing</u>.

Once the application is started, open up Rapise and click on FILE > Create New Test:

💐 Create New Test	×
Please specify the name of the new test:	
Java Test 1	
Specify the folder where this test should be created:	
C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\Java Test 1\Java Test 1.	sstest
Rapise supports multiple testing methodologies. Select one to be used for newly created	ated test.
Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
	Cancel

Enter the name "Java Test 1" as the name and choose **Basic: Windows Desktop Application** as the methodology.

On the next page, choose Rapise Visual Language (RVL) as the choice of Scripting language:



Once the test is created, you will see:

RVL	+Add New	· /						
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue	Н
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
2	•		~					
3								
4								
5								
6								
7								
8								
9								
10								
11								
12					~			
13								
14								
15								
16								
17								
18								
19								
20								

Click on the **Record** button to display the <u>"Select an Application to Record"</u> dialog:

R Select an Applicati			
Select From Running App	lications Run Applica	tion	
🚯 📓 AUTJAVA (C	:\Program Files (x86)\Ja	iva\jre1.8.0_112\bin\java.exe)	
 Applications 			
Available Applications	PID	Path	^
—	33212	C:\Windows\explorer.exe	
	39152	C:\Windows\ImmersiveControlPanel\Sy	
e	37028	C:\Windows\SystemApps\Microsoft.Micr	
—	33212	C:\Windows\explorer.exe	
	23784	C:\Program Files\WindowsApps\Micros	
	3192	C:\Program Files\WindowsApps\Micros	
Adobe Photoshop CS	3 Extended 3508	C:\Program Files (x86)\Adobe\Adobe P	
🕌 AUTJAVA	25100	C:\Program Files (x86)\Java\jre1.8.0_112	
📙 AUTJAVA	33212	C:\Windows\explorer.exe	
LAUTJAVA		C:\Windows\explorer.exe C:\Program Files\Dell\Dell Foundation S	
	Dell Founda 3356		~
C:\Program Files\Dell C:\Program Files\Dell	Dell Founda 3356	C:\Program Files\Dell\Dell Foundation S	~
C:\Program Files\Dell	Dell Founda 3356	C:\Program Files\Dell\Dell Foundation S	~
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries	Dell Founda 3356 TPad\Apnte 8572	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	×
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	 -
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library Auto	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support fo	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	 -
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library Auto Java	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support for Managed supp	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library Auto Java Managed	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support for Managed supp	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	 -
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library Auto Java Managed DevExpress	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support fr Managed supp DevExpress D Infragistics su	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library Auto Java Managed DevExpress Infragistics	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support fr Managed supp DevExpress D Infragistics su Telerik RadCou	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support opport	
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Dibrary Auto Java Managed DevExpress Infragistics Telerik	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support fr Managed supp DevExpress D Infragistics su Telerik RadCou	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell C:\Program Files\Dell Libraries Library Auto Java Managed DevExpress Infragistics Telerik Syncfusion	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support for Managed supp DevExpress D Infragistics su Telerik RadCon Syncfusion for Support for Act	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Dell\ C:\Program Files\Dell\ C:\Program Files\Dell' Libraries Library Auto Java Managed DevExpress Infragistics Telerik Syncfusion ActiveX	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support for Managed supp DevExpress D Infragistics su Telerik RadCor Syncfusion for Support for Ac Used for appli	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support poprt ntrols for Winforms Q3 2010 Windows Forms controls support tiveX Libraries	
C:\Program Files\Dell\ C:\Program Files\Dell\ C:\Program Files\Dell' Libraries Library Auto Java Managed DevExpress Infragistics Telerik Syncfusion ActiveX Qt Framework	Dell Founda 3356 TPad\Apnte 8572 Description Detect library a Java support for Managed supp DevExpress D Infragistics sup Telerik RadCoi Syncfusion for Support for Ac: Used for applii Tests .NET, W	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support poport ntrols for Winforms Q3 2010 Windows Forms controls support tiveX Libraries cations written using the Qt Fram	

Choose the **AUT JAVA** process from the list of running applications, change the library selection from Auto to "Java" and click **Select**.

Now in the sample application click on some of the AWT and/or Swing controls. Rapise will record the actions:

# >3. 1	Object	Action	Data	Comment							
≥1 ≥2	button1 advAwtCo	Action Action		Press button 'b Press button 'a							
₩4	list0	Sele	false	Do Selectitem(false) on list0						
1	jButton2	Action		Press button 'jE	Press button 'jButton2'						
105	jTabbedP	LClick	52,10	User clicks at	User clicks at 52, 10 in 'jTabbedPane1'						
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel				

When you click Finish, you will see the recorded test script and learned objects:

When you click **Finish**, Rapise will prompt you to confirm where you want the recorded test steps to be placed:

Start Page 🗙 📀 Spira Dashboard 🗴	Qt Sample Test.rvl.xls x
	New recording is available. Please, choose how to use recorded steps:
Insert Here Discard	

Select the first row in the test grid and click **Insert Here**. You will see the recorded test script and learned objects in Rapise:

Save to Spira	Ins Row 🚺 Ter	art Page 🐁 RVL st 🔹 Manua nctions 👍 Web S Shortcuts	ervices	Spy Play Level 1	bugging - 😻 Abo I (Minimal) + 🦞 Activ Let Obje	vation 🔄 Spira Properties ect Mgr	Find in Help Files		
File bject Tree	RVL	0			ephöðiuð	Tools	Help		
V a n at		RVL +Add New		d x Java Test 1.rvl.x	ISX ×				
Dbject Tree C:\Users\adam.san	dman\Documents\My R	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue	
- AUTJAVA		1 Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
 advAwtControls [advAwt] button1 [button1] 	Controls]	2 9 #	Press button		*				
 iButton2 [jButton2] 		31	Action Dutton1 D		DoAction				
) TabbedPane1 [jTabbedP	Pane1]	4 1 #	Press button	advAwtControls'					
Advanced AWT		50	Action @advAwtControls DoAction						
 button0 (button0) ist0 (list0) 		6 1=	Do SelectIten	"AWT ListItem1") on	list0				
> Global		70	Action	ist0	DoSelectItem	items	string	AWT ListItem1	
(2) User Functions		8 1 =	Press button	'button0'					
🕊 User Variables		9 0	Action	button0	DoAction				
		10 0 =	Press button	jButton2'					
		11 0	Action @jButton2		DoAction				
		12 # User clicks at: 59, 14 in 'jTabbedPane1'		ne1'					
		13 /	Action	jTabbedPane1	DoLClick	x	number	59	
		14 0	Param			У	number	14	
							199		

When you click **Play**, Rapise will play back your test script against the application:

#	Туре	Start	^ Name	Status	Comment
A		-	A	A	A
	Message	15:34:04.961	Starting scenario: Test	Info	
	Assert	15:34:05.354	button1.DoAction([])	Pass	Returned Value: true
	Assert	15:34:05.637	advAwtControls.DoAction([])	Pass	Returned Value: true
	Assert	15:34:05.927	list0.DoSelectItem([false])	Pass	Returned Value: true
	Assert	15:34:06.120	jButton2.DoAction([])	Pass	Returned Value: true
	Assert	15:34:06.462	jTabbedPane1.DoLClick([52,10])	Pass	Returned Value: true
	Test	15:34:06.467	Java Test 1	Pass	Passed:5 Failed:0

You can add steps to your script using any of the learned objects from the left-hand page (or any of the standard Global utility objects).

To do this, click on the blank row at the end of the recording and choose the following options from the dropdown lists in that row, for example:

- Type = Action
- Object = button0
- Action = DoAction

This process is illustrated below:

Action	🗆 button0	•	\sim	
		=	DoAction	-
		=	DoAnalogPlay	
		=	DoClick	
		=	DoDumpWidge DoA	tion()
		=	DoEnsureVisible	
		=	DoGetWidgetProper	ty
		=	DoLButtonDown	
		=	DoLButtonUp	

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the Spy button and Rapise will display the Java Spy:

🔎 Rapise Spy			- 🗆 X
Start Tracking (Ctrl-G)	Learn Object		
- Java Object - Tree		- Properties	
 panel0[java.awt.Panel] label0[java.awt.Label] textArea1[java.awt.TextArea] list1[java.awt.List] button1[java.awt.Button] button2[java.awt.Button] advAwtControls[java.awt.Button] checkbox1[java.awt.Checkbox] textField2[java.awt.TextField] choice2[java.awt.Choice] 		 Primary Handle IndexInParent Name VmID Secondary accessibleContext actionCommand actionListeners background enabled 	A44698794 4 button2 7146536 java.awt.Button\$AccessibleAWTE open dialog [Ljava.awt.event.ActionListener;@ java.awt.Color[r=240,g=240,b=240] true
- HWND - Tree button1[Button]		Focusable Properties	true V

You can then use the <u>Java Spy</u> to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy to the list of learned objects that you can test against.

Example 2 - Launching the Sample SWT Application

On the <u>Start Page</u> of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

Then go to C:\Users\Public\Documents\Rapise\Samples\JavaSWT\AUTJavaSWT and double-click on the JavaSWTAUT.bat file to start the sample application:

If you have Java configured correctly, you will see:

Test Java SWT Application Sample Java SWT Application Sample Text Box Tab 0 Tab 1 Tab 2 Buttons Image Button Simple Button Combos and Sliders This a very simple link widget. Image Substance	Sample Java SWT Application Sample Text Box Tab 0 Tab 1 Tab 2 Buttons Image Button Simple Button Check This O Radio 1 O Radio 2 Combos and Sliders This a very simple link widget.	Sample Java SWT Application Sample Text Box Tab 0 Tab 1 Tab 2 Buttons Image Button Simple Button Check This Radio 1 Radio 2 Combos and Sliders Combos and Sliders This a very simple link widget.	Test Ja											
Tab 0 Tab 1 Tab 2 Buttons Image Button Simple Button Combos and Sliders This a very simple link widget.	Sample Text Box Tab 0 Tab 1 Tab 2 Buttons Image Button Simple Button Combos and Sliders This a very simple link widget.	Sample Text Box Tab 0 Tab 1 Tab 2 Buttons Image Button Simple Button Check This Radio 2 Combos and Sliders This a very simple link widget.		wa SWT	Applica	tion						_		×
Buttons Image Button Simple Button Combos and Sliders This a very simple link widget.	Buttons Simple Button Combos and Sliders This a very simple link widget.	Buttons Image Button Simple Button Combos and Sliders This a very simple link widget.				ation								
Simple Button Check This Radio 1 Radio 2 Radio 2 Combos and Sliders This a very simple link widget.	Simple Button Check This Radio 1 Radio 2 Radio 2 Combos and Sliders This a very simple link widget. Image Button Image Button	Simple Button Check This Radio 1 Radio 2 Radio 2	Tab 0	Tab 1	Tab 2									
Combos and Sliders Combos and Sliders This a very simple link widget.	Combos and Sliders V V This a very simple link widget.	Combos and Sliders This a very simple link widget.	Butto	ns										
This a very simple link widget.	This a very simple link widget.	This a very simple link widget.	?	Image	e Button	Sin	nple B	Button		Check	This			
			Comb	oos and s	Sliders		~	Thi	is a ve	ery sin	nple [<u>ink</u> wi	idget.	
													-	
			<										-	
			<		, ,	1		1	1	1	I		-	
			<		· ·	1		1		1			_	

If the application doesn't start correctly, make sure you have Java SE installed and the JAVA_HOME environment variable correctly set to your Java Runtime (JRE). For more details on this, please refer to: <u>Java SWT Testing</u>.

Once the application is started, open up Rapise and click on FILE > Create New Test:

🂐 Create New Test	\times
Please specify the name of the new test:	
Java Test 1	
Specify the folder where this test should be created:	
C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\Java Test 1\Java Test 1.sstest	
Rapise supports multiple testing methodologies. Select one to be used for newly created test.	
Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
Cancel	

Once the application is started, open up Rapise and click on FILE > Create New Test:

Enter the name "Java Test 2" as the name and choose **Basic: Windows Desktop Application** as the methodology.

On the next page, choose Rapise Visual Language (RVL) as the choice of Scripting language:

R Scripting Language	—		\times
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scripti	ng Languag	e)	

Once the test is created, you will see:

	+Add New	1	(1	
	Flow	Туре	Object	Action	ParamName	ParamType	ParamValue	Н
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
2 →		~						
3								
4								
5								
6								
7								
8								
9								
10								
11								
12					~			
13								
14								
15								
16								
17								
18								
19								
20								
*								

Click on the Record button to display the "Select an Application to Record" dialog:

Select From Running Applica	tions Run Applica	tion	
 Applications 	Application (C:\Prog	ram Files (x86)\Java\jre1.8.0_112\bin\java.exe)	
Available Applications	PID	Path	/
C:\WINDOWS\system32\o	md exe 42892	C:\Windows\System32\cmd.exe	
Calculator	5540	C:\Windows\System32\ApplicationFram	
Euge Groove: Still Euge -		C:\Windows\System32\ApplicationFram	
Groove Music	5540	C:\Windows\System32\ApplicationFram	
on Index - Adam Sandman (Ir		C:\Program Files (x86)\Microsoft Office\r	
Rapise.hmxp in C:\Git\Rap		C:\Program Files (x86)\EC Software\Hel	
RE: [leaders] Election out		C:\Program Files (x86)\Microsoft Office\r	
RE: Monday Blog - Messa		C:\Program Files (x86)\Microsoft Office\r	
Test Java SWT Application	on 30812	C:\Program Files (x86)\Java\jre1.8.0_112	
	d-run ram 26376	C:\Program Files (x86)\Mozilla Firefox\fir	
US driver shot after hit-and		C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho			
US driver shot after hit-and Vonage Business Softpho	one 8988		
US driver shot after hit-and Vonage Business Softpho Libraries Library	Description	C:\Users\adam.sandman\AppData\Loc	_
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft	Description SftTree/OCX of	C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service	Description SftTree/OCX of Used for testin	C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT	Description SftTree/OCX of Used for testin UI Automation	C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML	Description SftTree/OCX of Used for testin UI Automation HTML DOM-ba	C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium	Description SftTree/OCX of Used for testin UI Automation HTML DOM-ba Selenium base	C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm	Description SftTree/OCX of Used for testin UI Automation HTML DOM-base Selenium base Dynamics CRI	C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML	Description SftTree/OCX of Used for testin UI Automation HTML DOM-base Selenium base Dynamics CRI	C:\Users\adam.sandman\AppData\Loc control support g RESTful web services support for Java SWT ased recorder for Internet Explorer ed playback engine M support library ased recorder for Mozilla Firefox	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML USer	Description SftTree/OCX of Used for testin UI Automation HTML DOM-base Dynamics CRI HTML DOM-base Dynamics CRI HTML DOM-base Default user-da	C:\Users\adam.sandman\AppData\Loc control support g RESTful web services support for Java SWT ased recorder for Internet Explorer ed playback engine M support library ased recorder for Mozilla Firefox	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML User Chrome HTML	Description SftTree/OCX of Used for testin UI Automation HTML DOM-base Dynamics CRI HTML DOM-base Default user-da HTML DOM-base	C:\Users\adam.sandman\AppData\Loc control support g RESTful web services support for Java SWT ased recorder for Internet Explorer ed playback engine M support library ased recorder for Mozilla Firefox efined library	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML User Chrome HTML DOM jQuery UI	Description SftTree/OCX of Used for testin UI Automation HTML DOM-base Dynamics CRI HTML DOM-base Default user-da HTML DOM-base	C:\Users\adam.sandman\AppData\Loc control support ig RESTful web services is support for Java SWT ased recorder for Internet Explorer ed playback engine M support library ased recorder for Mozilla Firefox efined library ased recorder for Google Chrome ecorder for jQuery UI widgets	
	8988 Description SftTree/OCX of Used for testin UI Automation HTML DOM-base Dynamics CRI HTML DOM-base Default user-date HTML DOM-base	C:\Users\adam.sandman\AppData\Loc control support ig RESTful web services is support for Java SWT ased recorder for Internet Explorer ed playback engine M support library ased recorder for Mozilla Firefox efined library ased recorder for Google Chrome ecorder for jQuery UI widgets	

Choose the **Test Java SWT Application** from the list of running applications, change the library selection from Auto to "**SWT**" and click **Select**.

Now in the sample application click on some of the SWT controls. Rapise will record the actions:

ŧ	Object	Action	Data	Comment				
2	OK	Action		Press butto	on 'OK'			
3	Simple Bu	Action		Press butto	on 'Simple Button'			
₿4	OK	Action		Press butto	on 'OK'			
5	List	SelectItem	0	Select items:'0' in "				
₿6	Slider	SetValue	10	Set value:'10' in "				
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel	

When you click **Finish**, Rapise will prompt you to confirm where you want the recorded test steps to be placed:

🖾 Start Page 🛛 🗙 📀	Spira Dashboard 🛛 🛛	Java Test 2.rvl.xls 🗙	
			New recording is available. Please, choose how to use recorded steps:
Insert Here	Discard		
RVL +Add New			

Select the first row in the test grid and click **Insert Here**. You will see the recorded test script and learned objects in Rapise:

Save to Spira Gopen Test from Spira File	Ins Row 💽 Tes	st nctions	RVL Manua Web Stortcuts	ervices	d Spy Play Level 1	(Minimal) - ? Abou (Minimal) - ? Activ all Obje (bugging	ation 📓 Spira Properties		
bject Tree		Sta	t Page x	Spira Dashboa	nd x Java Test 2. rvl.xl	sx x			
V kd m kdE		RVL	+Add New	1					
Object Tree C \Users\adam.san	dman\Documents\My R		Flow	Туре	Object	Action	ParamName	ParamType	ParamValue
 Test Java SWT Application Check_This [Check This] 		1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value
Cneck_This (check This) A Image_Button [Image Button] A OK [OK]		2 3	0#	Press button	'Simple Button'	v.			
		3.	0	Action	Simple_Button	DoAction			
Simple_Button [Simple Billing]	utton]	4	0#	Press button	'OK'				
 Slider [Slider] 		.5 ,	0	Action	IN OK	DoAction			
> Di List (List (List))		6	Ø #	Press button	'Image Button'				
List1 [List (List1)]		7	A	Action	Image_Button	DoAction			
Global		8 .	0#	Press button	'OK'				
 War Functions User Variables 		9	0	Action	(I) OK	DoAction			
		10	0#	Do SetCheck	(true) on Check This				
		11 .	0	Action	Check_This	DoSetCheck	bcheck	boolean	True
		12	0 #	Select items.	:'0' in "				
		13	0	Action	E List1	DoSelectItem	items	string[array]	0
		14	0#	Set value: '10	0' in "				
		15 .	0	Action	対 Slider	DoSetValue	val	number	10

When you click Play, Rapise will play back your test script against the application:

#	Type	Start	 Name 	Status	Comment
A		=			
	Message	15:51:20.765	Starting scenario: Test	Info	
	Assert	15:51:21.294	Image Button.DoAction([])	Pass	Returned Value: true
	Assert	15:51:21.762	OK.DoAction([])	Pass	Returned Value: true
	Assert	15:51:22.265	Simple Button.DoAction([])	Pass	Returned Value: true
	Assert	15:51:22.716	OK.DoAction([])	Pass	Returned Value: true
	Assert	15:51:23.310	ComboBox.DoSelectItem(["Alpha"])	Pass	Returned Value: true
	Assert	15:51:24.030	Slider.DoSetValue([10])	Pass	Returned Value: true
	Test	15:51:24.036	Java Test 2	Pass	Passed:6 Failed:0

You can add steps to your script using any of the learned objects from the left-hand page (or any of the standard Global utility objects).

To do this, click on the blank row at the end of the recording and choose the following options from the dropdown lists in that row, for example:

- Type = Action
- Object = Image_Button
- Action = DoAction

This process is illustrated below:

Action	Image Button	\varTheta Do	Action	\sim	
			DoAction		Ľ
		=0	DoAnalog		
		=♦	DoClick	Do Action()	
		=♦	DoDumpW	iuger	
		= \	DoEnsure\	/isible	
			DoFindByT	ext	
		=0	DoGetWidg		у
			DoLButton		

You can drag and drop any of the learned objects from the left-hand pane into the main test script. You can also just type **SeS("OK")** (for example) and Rapise will display the list of available functions. When you click **Play**, Rapise will play back your test script against the application:

#	Туре	Start	 Name 	Status	Comment
A		=	۵.		
	Message	15:51:20.765	Starting scenario: Test	Info	
	Assert	15:51:21.294	Image Button.DoAction([])	Pass	Returned Value: true
	Assert	15:51:21.762	OK.DoAction([])	Pass	Returned Value: true
	Assert	15:51:22.265	Simple Button.DoAction([])	Pass	Returned Value: true
	Assert	15:51:22.716	OK.DoAction([])	Pass	Returned Value: true
	Assert	15:51:23.310	ComboBox.DoSelectItem(["Alpha"])	Pass	Returned Value: true
	Assert	15:51:24.030	Slider.DoSetValue([10])	Pass	Returned Value: true
	Test	15:51:24.036	Java Test 2	Pass	Passed:6 Failed:0

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the **Spy** button and Rapise will display the <u>UIAutomation Spy</u>:

Induct py Learn Object UlAutomation Object Properties Buttons[Group] Image Button[Button] Simple Button[Button] Simple Button[Button] Check This[CheckBox] Radio 2[RadioButton] Radio 2[RadioButton] Primary Primary Properties Physics Primary Automation D 9700336 IndexInParent 1 Name Simple Button ValueProperty 1048576	🔎 Rapise Spy			– 🗆 X
UlAutomation Object Properties Tree HelpProperty Image Button[Button] Simple Button[Button] Simple Button[Button] Check This[CheckBox] Radio 1[RadioButton] SilectionProperty Radio 2[RadioButton] Year ValueProperty UlAutomation Automation Element ValueProperty 1048576 ValueProperty ValueProperty ValueProperty ValueProperty ValueProperty ValueProperty ValueProperty 9700336 IndexInParent 1 Name Simple Button Vertice 9700336 IndexInParent 1 Name 9700336 Vertice 9700336	Va Rupise spy			
Tree Properties Intrage Button[Group] HelpProperty KeyboardShotcutProperty Image Button[Button] Simple Button[Button] Simple Button[Button] RoleProperty Simple Button Simple Button Radio 1[RadioButton] SelectionProperty System.Windows Automation AutomationElement StateProperty 1048576 AutomationID 9700336 IndexInParent 1 Name Simple Button ValueProperty ValueProperty HMND Tree Properties	Start Tracking (Ctrl-G)	Learn Object		H •
Tree Properties Intrage Button[Group] HelpProperty KeyboardShotcutProperty Image Button[Button] Simple Button[Button] Simple Button[Button] RoleProperty Simple Button Simple Button Radio 1[RadioButton] SelectionProperty System.Windows Automation AutomationElement StateProperty 1048576 AutomationID 9700336 IndexInParent 1 Name Simple Button ValueProperty ValueProperty HMND Tree Properties				
Buttons[Group] Image Button[Button] Simple Button[Button] Check This[CheckBax] Radio 1[RadioButton] Radio 2[RadioButton] Radio 2[RadioButton] Radio 2[RadioButton] Radio 1[RadioButton] Radio 2[RadioButton] Pinmay Tomay Radio 2[RadioButton] Radio 2[RadioButton]			- Properties	
Image Button[Button] KeyboardShortcutProperty Simple Button Simple Button[Button] NameProperty Simple Button Check This[CheckBox] SelectionProperty 43 SelectionProperty System Windows Automation AutomationElement Radio 1[RadioButton] StateProperty 1048576 ValueProperty ValueProperty ValueProperty ValueProperty 1048576 ValueProperty ValueProperty ValueProperty ValueProperty <td></td> <td></td> <td></td> <td><u>^</u></td>				<u>^</u>
Simple Button[Button] NameProperty Simple Button Check This[CheckBox] Radio 1[RadioButton] RoleProperty System Windows Automation AutomationElement Radio 2[RadioButton] SelectionProperty 1048576 ValueProperty 1048576 ValueProperty 000336 IndexInParent 1 Name Simple Button Name Simple Button Name Simple Button Tree Properties				
Check This[CheckBox] RoleProperty 43 Radio 1[RadioButton] SelectionProperty System.Windows Automation AutomationElement Radio 2[RadioButton] 1048576 ValueProperty 1048576 ValueProperty ValueProperty ValueProperty				Simple Button
Radio 1[RadioButton] SelectionProperty System: Windows Automation AutomationElement Radio 2[RadioButton] StateProperty 1048576 ValueProperty ValueProperty ValueProperty V Primary AutomationID 9700336 IndexInParent 1 Name Name Simple Button v IHWND Properties v			RoleProperty	43
Radio 2[RadioButton] StateProperty 10485/6 ValueProperty ValueProperty AutomationID 9700336 IndexInParent 1 Name Simple Button NativeHandle 9700336 - HWND - - Tree Properties				
HWND Tree Properties				1048576
AutomationID 9700336 IndexInParent 1 Name Simple Button NativeHandle 9700336 ✓ +HWND □ Tree				
IndexInParent 1 Name Simple Button NativeHandle 9700336 Image: HWND Tree				0700220
Name Simple Button NativeHandle 9700336 →				1
NativeHandle 9700336 ✓ - HWND				Simple Button
HWND Tree Properties				
Tree Properties				
			- Properties	
	Simple Button[Button]			

You can then use the <u>UIAutomation Spy</u> to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy to the list of learned objects that you can test against.

References

- Java AWT/Swing Testing
- Java SWT Testing

2.2.9.1 Using JavaScript

In this section, you will learn how to record and execute a Rapise script against a Java applications. We will show you have to test the following three different types of Java application:

o Java AWT Apps

- o Java Swing Apps
- o Java SWT Apps

This version of the tutorial uses the JavaScript test editor option in Rapise. If you'd prefer to use the Rapise Visual Language (RVL), please go to the main <u>Tutorial</u> instead.

Example 1 - Launching the Sample AWT/Swing Application

On the <u>Start Page</u> of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

Then go to C:\Users\Public\Documents\Rapise\Samples\Java\AUTJAVA and right-click on the x86run.cmd file and choose Run as Administrator.

If you have Java configured correctly, you will see:

🛓 Autjava		_		×
AWT label1				
AWT TextArea	button1	AWT List	titem2 titem3 titem4	
open dialog Adva	anced Awr Control	5		
Swing jButton2 jToggleButt	Item 1 ^ Item 2 V Item 3 V	JTree colors sports		
 jRadioButton1 jRadioButton2 	tab1 tab2	Advanced 9	Swing Con	t
Item 1 V jTextField1		jButton3		
Swing TextArea Second Line				^

If the application doesn't start correctly, make sure you have Java SE and the <u>Rapise Java Bridge</u> installed and the JAVA_HOME environment variable correctly set to your Java Runtime (JRE). For more details on this, please refer to: <u>Java AWT/Swing Testing</u>.

Once the application is started, open up Rapise and click on **FILE** > Create New Test:

×
ncel

Enter the name "Java Test 1" as the name and choose **Basic: Standard Scripting Mode** as the methodology.

Choose the JavaScript option during test creation instead of RVL.

R Scripting Language	_		×
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scriptin	ig Languag	e)	

Once the test is created, you will set:

R 🗟 🔿 🕞 📲 🖏 + File Text Options Edit	RapiseGloud 5.0.733 (C)(bees)aclamitaneiman\Documents\My RapiseTests\TwoDialogsTest\TwoDialogsTest.utest)	□ - 6° ×
Objects	Start Page 2 TwoDistopsText]	- *
V = + = Orgert Two (Utions) (data: sending/Docy + ⊒ Global W User Functions T User Vanable	<pre>//Use 'Record/Learn' button to begin test recording function Test()</pre>	~
Countin (Price Suttings Properties 22.21		
		5
	Outpet	
	Microsoft (R) Windows Script Host Version 5.812 Corpsylipt (c) Microsoft Corpusation: A highs reserved Sea Recording Strating. Veholes level 1 Recording cript file (C) Usersjadem sandmani Documente/My Rapse Testing 1Web Testing 1 (a	*
	Preconding sectors to conservation and conservation response reactive or reading rates reading rates Preconding values there regists " classification regists" reading rates reading rates Preconding values reading rates reading rates reading reading rates reading rates reading rates Preconding values rates reading rates reading rates reading rates reading rates reading rates reading rates Preconding values rates rates reading rates reading rates reading rates reading rates reading rates reading rates Preconding values rates rate	*

Click on the **Record** button to display the <u>"Select an Application to Record"</u> dialog:

Select From Running Ap	plications Run Applicat	tion	
🔮 📓 AUTJAVA (0	C:\Program Files (x86)\Ja	wa\jre1.8.0_112\bin\java.exe)	
Applications			
Available Applications	PID	Path	
	33212	C:\Windows\explorer.exe	
	39152	C:\Windows\explores.exe C:\Windows\ImmersiveControlPanel\Sy	
e	37028	C:\Windows\SystemApps\Microsoft.Micr	
	33212	C:\Windows\csystemApps\wildows\csystemApps\wildows\csystemApps\wildows\csystemApps\wildows\csystemApps\wildows\csystemApps\wildows\csystemApps\\wildows\csystemApps\\wildows\csystemApps\\wildows\csystemApps\\\wildows\csystemApps\\\wildows\csystemApps\\\wildows\csystemApps\\\\wildows\csystemApps\\\\wildows\csystemApps\\\\wildows\csystemApps\\\\\wildows\csystemApps\\\\\wildows\csystemApps\\\\\\wildows\csystemApps\\\\\\wildows\csystemApps\\\\\\wildows\csystemApps\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	
	23784	C:\Program Files\WindowsApps\Micros	
	3192	C:\Program Files\WindowsApps\Micros	
Adobe Photoshop CS		C:\Program Files (x86)\Adobe\Adobe P	
AUTJAVA	25100	C:\Program Files (x86)\Java\jre1.8.0_112	
	33212	C:\Windows\explorer.exe	
~			
AUTJAVA C:\Program Files\Del C:\Program Files\Del C:\Program Files\Del	I\Dell Founda 3356	C:\Program Files\Dell\Dell Foundation S	
~	I\Dell Founda 3356		
 C:\Program Files\Dell C:\Program Files\Dell	I\Dell Founda 3356	C:\Program Files\Dell\Dell Foundation S	
⊷ ■C:\Program Files\Del ●C:\Program Files\Del □Libraries	I\Dell Founda 3356	C:\Program Files\Dell\Dell Foundation S	
← ■ C:\Program Files\Del ● C:\Program Files\Del - Libraries Library	I\Dell Founda 3356 ITPad\Apnte 8572	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
← ■ C:\Program Files\Del ■ C:\Program Files\Del - Libraries Library - Auto	I\Dell Founda 3356 ITPad\Apnte 8572 Description Detect library a	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
← C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java	I\Dell Founda 3356 ITPad\Apnte 8572 Description Detect library a Java support fo	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed	I\Dell Founda 3356 ITPad\Apnte 8572 Description Detect library a Java support fo Managed supp	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed DevExpress	I\Dell Founda 3356 ITPad\Apnte 8572 Description Detect library a Java support fo Managed supp	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
C:\Program Files\Del C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed DevExpress Infragistics	I/Dell Founda 3356 ITPad/Apnte 8572 Description Detect library a Java support fr Managed supp DevExpress D Infragistics sup	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe	
 C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed DevExpress Infragistics Telerik 	I\Dell Founda 3356 ITPad\Apnte 8572 Description Detect library a Java support fo Managed supp DevExpress D Infragistics sup Telerik RadCor	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support opport	
 C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed DevExpress Infragistics Telerik Syncfusion 	I\Dell Founda 3356 ITPad\Apnte 8572 Description Detect library a Java support fo Managed supp DevExpress D Infragistics sup Telerik RadCor	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support oport ntrols for Winforms Q3 2010 Windows Forms controls support	
 C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed DevExpress Infragistics Telerik Syncfusion ActiveX 	I/Dell Founda 3356 ITPad/Apnte 8572 Description Detect library a Java support fo Managed supp DevExpress D Infragistics sup Telerik RadCor Syncfusion for Support for Act	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support oport ntrols for Winforms Q3 2010 Windows Forms controls support	
 C:\Program Files\Del C:\Program Files\Del Libraries Library Auto Java Managed DevExpress Infragistics Telerik Syncfusion ActiveX Qt Framework 	I/Dell Founda 3356 ITPad/Apnte 8572 Description Detect library a Java support for Managed supp DevExpress D Infragistics sup Telerik RadCor Syncfusion for Support for Act Used for applie	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support poport ntrols for Winforms Q3 2010 Windows Forms controls support tiveX Libraries	
C:\Program Files\Del	I/Dell Founda 3356 ITPad/Apnte 8572 Description Detect library a Java support for Managed supp DevExpress D Infragistics sup Telerik RadCor Syncfusion for Support for Act Used for applio Tests .NET, W	C:\Program Files\Dell\Dell Foundation S C:\Program Files\DellTPad\ApntEx.exe automatically or AWT/SWING ort for .NET 2.0, 3.0, 3.5, 4.x Xperience v10 support oport ntrols for Winforms Q3 2010 Windows Forms controls support tiveX Libraries cations written using the Qt Fram	

Choose the **AUT JAVA** process from the list of running applications, change the library selection from Auto to "**Java**" and click **Select**.

Now in the sample application click on some of the AWT and/or Swing controls. Rapise will record the actions:

#	Object	Action	Data	Comment						
尚1	button1	Action		Press button 'bu	Press button 'button1'					
2	advAwtCo	. Action		Press button 'a	dvAwtControls'					
i) 4	list0	Sele	false	Do SelectItem(f	false) on list0					
i) 4	jButton2	Action		Press button 'jB	lutton2'					
jii 5	jTabbedP	. LClick	52,10	User clicks at 5	52, 10 in 'jTabbedF	Pane1'				
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel			

File Test Options E	Edit				
Gen Save [™] Save to Spira	 Start Page Spira Dashboard Test Manual Steps 	Record Play	No Debugging + Level 1 (Minimal) +	 About Convert to NeoLoad Activation Spira Properties 	Pind in Help
💕 Open Test from Spira	Functions 👍 Web Services	Spy		Object Mgr	Files
File	Shortcuts	Record and Execute	Debugging	Tools	Help
Objects	🖾 Start Page 👔 Java Test	tt 1.j			
Objects Files Settings Objects Files Settings	3 //###################################		<pre>day, December 30, ; utrols' utrols' utrols' (false); (); in 'jTabbedPane;</pre>	, 2016 3:28:53 PM: ======	

When you click Finish, you will see the recorded test script and learned objects:

You can drag and drop any of the learned objects from the left-hand pane into the main test script. You can also just type **SeS("button1")** (for example) and Rapise will display the list of available functions. When you click **Play**, Rapise will play back your test script against the application:

#	Туре	Start	^ Name	Status	Comment
	Δ	-	A	A	A
	Message	15:34:04.961	Starting scenario: Test	Info	
	Assert	15:34:05.354	button1.DoAction([])	Pass	Returned Value: true
	Assert	15:34:05.637	advAwtControls.DoAction([])	Pass	Returned Value: true
	Assert	15:34:05.927	list0.DoSelectItem([false])	Pass	Returned Value: true
	Assert	15:34:06.120	jButton2.DoAction([])	Pass	Returned Value: true
	Assert	15:34:06.462	jTabbedPane1.DoLClick([52,10])	Pass	Returned Value: true
•	Test	15:34:06.467	Java Test 1	Pass	Passed:5 Failed:0

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the Spy button and Rapise will display the Java Spy:

Rapise Spy			- 🗆 X
Start Tracking (Ctrl-G)	Learn Object		H
Java Object Tree		Description	
Tree panel0[java.awt.Panel]		Properties Primary	
label0[java.awt.Label]		Handle	44698794
		IndexInParent	4
textArea1[java.awt.TextArea]		Name	button2
list1[java.awt.List]		VmID	7146536
button1[java.awt.Button]		✓ Secondary	7110000
button2[java.awt.Button]		accessibleContext	java.awt.Button\$AccessibleAWT
advAwtControls[java.awt.Button]		actionCommand	open dialog
checkbox1[java.awt.Checkbox]		actionListeners	[Ljava.awt.event.ActionListener;@
textField2[java.awt.TextField]		background	java.awt.Color[r=240,g=240,b=240
choice2[java.awt.Choice]		enabled	true
		focusable	true
- HWND - Tree		- Properties	
button1[Button]			

You can then use the <u>Java Spy</u> to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy to the list of learned objects that you can test against.

Example 2 - Launching the Sample SWT Application

On the <u>Start Page</u> of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

Then go to C:\Users\Public\Documents\Rapise\Samples\JavaSWT\AUTJavaSWT and double-click on the JavaSWTAUT.bat file to start the sample application:

If you have Java configured correctly, you will see:

🔳 Test Ja	va SWT A	pplicati	on					_		×
	ava SWT Text Box		tion							
Tab 0	Tab 1	Tab 2								
Buttor	IS									
?	Image	Button	Simple	e Button		Check	This		idio 1 idio 2	
Comb	os and SI ~	iders	~] Th	is a ve	ery sin	nple <u>I</u>	ink wie	dget.	
<								>		
	1 1	T	· ·	1	I	T	1	1		
-										
-										

If the application doesn't start correctly, make sure you have Java SE installed and the JAVA_HOME environment variable correctly set to your Java Runtime (JRE). For more details on this, please refer to: <u>Java SWT Testing</u>.

Once the application is started, open up Rapise and click on FILE > Create New Test:

Reate New Test	×
Please specify the name of the new test:	
Specify the folder where this test should be created: C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\My Rapise Tests.sste	st
Rapise supports multiple testing methodologies. Select one to be used for newly created	ted test.
Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
	Cancel

Enter the name "Java Test 2" as the name and choose **Basic: Windows Desktop Application** as the methodology.

Choose the JavaScript option during test creation instead of RVL.

R Scripting Language	_		\times
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scriptin	g Languag	e)	

Once the test is created, you will see:

R 😳 🔿 🖨 👍 😋 =	RepsedDoud 5.0.7.33 (C1)bestsedam.sentman(Documents)My Repue Tests)TwoDoalogsTest11TwoDoalogsTest1stmet	⊞ - ß ×
	Page 2 Spira Dashboard Manual Steps Manual Steps Second Spira Dashboard Manual Steps Second Spira Dashboard Spira Das	
Objects	Start Page 21 TwoDeskgsTest j	- 18
V M O Opert Three C (User) (adam, sambure (Doco O Good Variant Sambure Variant Sambure User Variables	//We 'Record/Learn' buttoh to begin test recording function Test() a a a *)	
Control Play Sample Control Play Sample Sector 2010	د Compation	a ^v
	Microsoft/QV midues StrightPato Vession 5.812 Corpright (Officiando Clopanta) and a stright searced SaS Recorder Studing.	*
	Verbose Invet? Recording screet Rec (Userspace) Conservation and Conservation (My Reprint Testing TWeb Testing T) is Creating a local conservation Creating a local conservation Creating Conservation (Conservation) (Conservation	•

Click on the Record button to display the "Select an Application to Record" dialog:

	to Record	—	×
Select From Running Applicat	tions Run Applic	ation	
Test Java SWT A Applications	Application (C:\Prog	gram Files (x86)\Java\jre1.8.0_112\bin\java.exe)	
Available Applications	PID	Path	
C:\WINDOWS\system32\c			
Calculator	5540	C:\Windows\System32\ApplicationFram	
Euge Groove: Still Euge -		C:\Windows\System32\ApplicationFram	
Groove Music	5540	C:\Windows\System32\ApplicationFram	
Inbox - Adam Sandman (Ir		C:\Program Files (x86)\Microsoft Office\r	
Rapise.hmxp in C:\Git\Rap		C:\Program Files (x86)\EC Software\Hel	
RE: [leaders] Election out		C:\Program Files (x86)\Microsoft Office\r	
RE: Monday Blog - Messa		C:\Program Files (x86)\Microsoft Office\r	
Test Java SWT Application	on 30812	C.\Program Files (Xob)\Java\[re1.o.U_112]	
US driver shot after hit-and	l-run ram 26376		
US driver shot after hit-and Vonage Business Softpho	l-run ram 26376	C:\Program Files (x86)\Mozilla Firefox\fir	
US driver shot after hit-and Vonage Business Softpho Libraries	d-run ram 26376 one 8988	C:\Program Files (x86)\Mozilla Firefox\fir	
US driver shot after hit-and Vonage Business Softpho Libraries ibrary	I-run ram 26376 one 8988 Description	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft	I-run ram 26376 one 8988 Description SftTree/OCX	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services n support for Java SWT	
US driver shot after hit-and Vonage Business Softpho Libraries .ibrary ActiveXSft Web Service SWT Internet Explorer HTML	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services n support for Java SWT based recorder for Internet Explorer	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services n support for Java SWT based recorder for Internet Explorer sed playback engine	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas Dynamics CF	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services n support for Java SWT based recorder for Internet Explorer	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas Dynamics CF HTML DOM-t	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services in support for Java SWT based recorder for Internet Explorer sed playback engine RM support library	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas Dynamics CF HTML DOM-t Default user-	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services in support for Java SWT based recorder for Internet Explorer sed playback engine RM support library based recorder for Mozilla Firefox defined library	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML User	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas Dynamics CF HTML DOM-t Default user HTML DOM-t	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services in support for Java SWT based recorder for Internet Explorer sed playback engine RM support library based recorder for Mozilla Firefox	
US driver shot after hit-and Vonage Business Softpho Libraries Library ActiveXSft Web Service SWT Internet Explorer HTML Selenium DomDynamicsCrm Firefox HTML User Chrome HTML	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas Dynamics CF HTML DOM-t Default user HTML DOM-t	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services in support for Java SWT based recorder for Internet Explorer sed playback engine RM support library based recorder for Mozilla Firefox defined library based recorder for Google Chrome recorder for jQuery UI widgets	
Chrome HTML DOM jQuery UI	I-run ram 26376 one 8988 Description SftTree/OCX Used for testi UI Automatio HTML DOM-t Selenium bas Dynamics CF HTML DOM-t Default user-o HTML DOM-t DoM-based r	C:\Program Files (x86)\Mozilla Firefox\fir C:\Users\adam.sandman\AppData\Loc control support ing RESTful web services in support for Java SWT based recorder for Internet Explorer sed playback engine RM support library based recorder for Mozilla Firefox defined library based recorder for Google Chrome recorder for jQuery UI widgets	

Choose the **Test Java SWT Application** from the list of running applications, change the library selection from Auto to "**SWT**" and click **Select**.

Now in the sample application click on some of the SWT controls. Rapise will record the actions:

#	Object	Action	Data	Comment			
2	OK	Action		Press butto	on 'OK'		
13	Simple Bu	Action		Press butto	on 'Simple Button'		
¥ 🖗	OK	Action		Press butto	on 'OK'		
10 5	List	SelectItem	0	Select item	ns:'0' in "		
0 🖏	Slider	SetValue	10	Set value:	10' in ''		
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel

When you click Finish, you will see the recorded test script and learned objects:

Save Save to Spira	Start Page (2) Spira Dashboard Start Page (2) Spira Dashboard Image: Convert to NeoLoad Image: Test Manual Steps Functions Web Services Shortcuts Record and Execute No Debugging Image: Convert to NeoLoad Up of the services Spy Shortcuts Record and Execute Debugging Tools
Objects ✓ Id n Idf ✓ Object Tree C:\Users\adam.sandr ▲ Test Java SWT Application > B: ComboBox [ComboBox] > @ Image_Button [Image Butto > @ OK [OK] > @ Simple_Button [Simple Butto > \$ Silder[Sider] ▲ List [List] > Global Ø User Functions ✓ User Variables	Image: Start Page: Star
Objects Files Settings	22 g_load_libraries=["SWT"]; 23

You can drag and drop any of the learned objects from the left-hand pane into the main test script. You can also just type **SeS("OK")** (for example) and Rapise will display the list of available functions. When you click **Play**, Rapise will play back your test script against the application:

#	Type	Start	^ Name	Status	Comment
0		=	a		A
	Message	15:51:20.765	Starting scenario: Test	Info	
	Assert	15:51:21.294	Image Button.DoAction([])	Pass	Returned Value: true
	Assert	15:51:21.762	OK.DoAction([])	Pass	Returned Value: true
	Assert	15:51:22.265	Simple Button.DoAction([])	Pass	Returned Value: true
	Assert	15:51:22.716	OK.DoAction([])	Pass	Returned Value: true
	Assert	15:51:23.310	ComboBox.DoSelectItem(["Alpha"])	Pass	Returned Value: true
	Assert	15:51:24.030	Slider.DoSetValue([10])	Pass	Returned Value: true
	Test	15:51:24.036	Java Test 2	Pass	Passed:6 Failed:0

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the **Spy** button and Rapise will display the <u>UIAutomation Spy</u>:

🔎 Rapise Spy			– 🗆 X
Start Tracking (Ctrl-G)	Learn Object		
UlAutomation Object Tree UlAutomation Object Image Button[Group] Image Button[Button] Check This[CheckBox] Radio 1[RadioButton] Radio 2[RadioButton]		Properties HelpProperty KeyboardShortcutProperty NameProperty RoleProperty StateProperty StateProperty ValueProperty ValueProperty AutomationID IndexInParent Name NativeHandle	Simple Button 43 System Windows Automation AutomationElement 1048576 9700336 1 Simple Button 9700336
- HWND Tree Simple Button[Button]		Properties	

You can then use the <u>UIAutomation Spy</u> to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy to the list of learned objects that you can test against.

References

- Java AWT/Swing Testing
- Java SWT Testing

2.2.10 Tutorial: Qt Framework

Rapise includes support for testing applications written using the Qt Framework written using QWidget controls.

To ensure that Rapise can access the UI elements and properties in the Qt application being tested, <u>MSAA (Microsoft Active Accessibility) support for your Qt application must be enabled</u>. This provides additional information on Qt UI elements to automation software like Rapise and can be accomplished by shipping and loading the "Accessible Plug-in" included in the Qt SDK (Software Development Kit) with the Qt application under test (see below).

This tutorial illustrates the ability of Rapise to test such Qt applications using a sample application that already has the MSAA support added.

This version of the tutorial uses the **Rapise Visual Language (RVL) scriptless** mode. If you're interested in the <u>JavaScript version</u>, we have a separate tutorial.

Testing the Sample Qt Application

On the <u>Start Page</u> of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

Then go to C:\Users\Public\Documents\Rapise\Samples\QtFramework and double-click on the QtWidgetApp.exe file to start the sample application:

If you have everything configured correctly, you will see:

🚺 Sample C	QtWidget Application			_		×
<u>F</u> ile <u>V</u> iew						
Sample	OtWidget Application for R					
Name:		Filename:				
Description:		Font:			Choose	e
		Color:		Chang	e Color	
Location:	Headquarters 🗾	Print:		Print		
Options: (Option 1 Option 2 Is Active?	Input:				
Font:	MS Shell Dlg 2	Errors:	D	isplay Errors.		
	12:00:00 AM 🛨 1/1/2000 🛨	Message:	Dis	splay Message	e	
Date/Time:	1/1/2000 12:00:00 AM					
Notes:						
,	OK Cancel					

Reate New Test	×
Please specify the name of the new test:	
Specify the folder where this test should be created:	
C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\My Rapise Tests.sst	est
Rapise supports multiple testing methodologies. Select one to be used for newly created	ated test.
Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
	Cancel

Once the application is started, open up Rapise and click on **FILE** > Create New Test:

Enter the name "Qt Sample Test" as the name and choose Basic: Windows Desktop Application as the methodology.

On the next page, choose Rapise Visual Language (RVL) as the choice of Scripting language:

R Scripting Language	_		\times
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scriptin	g Languago	e)	

Once the test is created, you will see:

Open Den Den	st nctions	 RVL Manua Web S Mortcuts 	iervices	Spy Play Level	ebugging -	vation 📓 Spira Properties	Find in Help Files Help		
ect Tree	1			x Qt Sample Tes		1 Lineto	Charge		
V 6 1 d		+Add New		A Get Galinpie Fea					
Object Tree C. (Users)adam sandman/Documents (My R	1	Flow	Type	Object	Action	ParamName	ParamType	ParamValue	н
Global Global Global Global	1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value	
User Variables	2		~						
	.3								
	4								
	5	-							
	6								
	7								
	9								
	10								
	11								
	12							v	
	13								
	14								
	15								
	16	· · · · ·							
>	17	· · · · · ·							
ect Tree Files Settings	18								
perties	19								
21	20								

Click on the **Record** button to display the <u>"Select an Application to Record"</u> dialog:

	Record				
Select From Running Applicatio	ns Run Applic	ation			
🚱 🛛 🕅 Sample QtWidget A	pplication (C:\U	Jsers\Public\Documents\Rapise\Samples\QtF	ramew	vork\QtWidget	tApp.e
- Applications					
Available Applications	PID	Path			/
C:\WINDOWS\system32\cm	d.exe 18972	C:\Windows\System32\cmd.exe			
Calculator	5540	C:\Windows\System32\ApplicationFram	i		
🔤 Euge Groove: Still Euge - Ja	ngo ? 5540	C:\Windows\System32\ApplicationFram			
Groove Music	5540	C:\Windows\System32\ApplicationFram			
💁 Inbox - Adam Sandman (Infle	ectra) 1428	C:\Program Files (x86)\Microsoft Office\r			
QtFramework	33212				
Rapise.hmxp in C:\Git\Rapis	e\Rap 9924	C:\Program Files (x86)\EC Software\Hel			
RE: [leaders] Election outco		C:\Program Files (x86)\Microsoft Office\r			
RE: Monday Blog - Message		C:\Program Files (x86)\Microsoft Office\r			
Sample QtWidget Applicatio					- 1
Test Java SWT Application	3120	C:\Program Files (x86)\Java\jre1.8.0_112	2		
- Libraries	1				
Library	Description				
Infragistics	Infragistics s				
Telerik	Telerik RadC	ontrols for Winforms Q3 2010			
Syncfusion	Syncfusion for	or Windows Forms controls support			
ActiveX		ctiveX Libraries			
Qt Framework	Used for app	lications written using the Qt Fram			
UIAutomation	Tests .NET,	WPF and Silverlight Applications			
ActiveXC1	Support for (Component One 8.0 ActiveX Librari			
ActiveXSft	SftTree/OCX	ee/OCX control support			
	Used for test	ing RESTful web services			
Web Service					
	UI Automatio	n support for Java SWT			
Web ServiceSWTInternet Explorer HTML		n support for Java SWI based recorder for Internet Explorer			•

Choose the Sample QtWidget Application from the list of running applications, change the library

selection from Auto to "Qt Framework" and click Select.

Now in the sample application click on some of the Qt controls. Rapise will record the actions:

#	Object	Action	Data	Comment					
尚1	txtName_ SetText		Adam	Do SetText					
≥2	radOption	SetCheck	True	Do SetCheck(true) on radOption2_					
13	chkActive_	SetCheck	True	Do SetCheck(true) on chkActive_					
₿5	OK_	Action		Press button 'OK_'					
	Verify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel		

When you click **Finish**, Rapise will prompt you to confirm where you want the recorded test steps to be placed:

Start Page 🗙 📀 Spira Dashboard 🗄	The second secon
	New recording is available. Please, choose how to use recorded steps:
Insert Here Discard	

Select the first row in the test grid and click **Insert Here**. You will see the recorded test script and learned objects in Rapise:

Gen Save Save to Spira Open Test from Spira	Ins Row 🛐 Te	art Page RVL est 🔹 Man inctions 👍 Web	ual Steps Reco	Level	ebugging • 🤣 Abo 1 (Minimal) • 📍 Activ all Obje	vation 📓 Spira Properties		
File	RVL.	Shortcuts	Reci	ord and Execute [Jebugging	Tools	Help	
bject Tree		Start Page	Spira Dashbor	ard x Qt Sample Test	Livi xkx ×			
Etter be 🖓		RVL +Add M						
Object Tree C:\Users\adam.sandman\Documents\My R		Flow	Туре	Object	Action	ParamName	ParamType	ParamValue
 Sample QtWidget Applicatio ChkActive_[chkActive_] 		1 Flow	Туре	Object	Action	Param Name	Param Type	Param Value
<pre>> B OK_[OK_]</pre>		2 9	Do SetText("edum") on txtNume_				
> (radOption2_[radOption2]	L	30	Action	iei txtName_	DoSetText	val	string	Adam
> bdtName_[bdtName_]		4 0 #	Do SetCheck	k(true) on radOption2_				
Clobal		50	Action	⊙ radOption2_	DoSetCheck	bcheck	boolean	True
 User Functions User Variables 		6 1#	Do SetCheck	k(true) on chkActive_				
User Variables		70	Action	ChkActive_	DoSetCheck	bcheck	boolean	True
		8 0 #	Press button	'OK_'				
		90	Action	OK_	DoAction			
		10						

When you click Play, Rapise will play back your test script against the application:

-			by that column.		
#	Туре	Start	^ Name	Status	Comment
A	A	-			
	Message	16:04:25.733	Starting scenario: Test	Info	
	Assert	16:04:28.764	txtNameDoSetText(["Adam"])	Pass Ret	urned Value: true
	Assert	16:04:30.682	radOption2DoSetCheck([true])	Pass Ret	urned Value: true
	Assert	16:04:32.308	chkActiveDoSetCheck([true])	Pass Ret	urned Value: true
	Assert	16:04:34.105	OKDoAction([])	Pass Ret	urned Value: true
	Test	16:04:34.112	Qt Sample Test	Pass Pass	sed:4 Failed:0
			Test Pass Total:6 Pass:5 Fail:0 Info:1		

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the **Spy** button and Rapise will display the <u>Accessible</u> <u>Spy</u>:

🔎 Rapise Spy						-		\times	
Start Tracking (Ctrl-G)	Learn Object								•
- Accessible			-	Properties					
 [ROLE_SYSTEM_CLIENT.] Name: [ROLE_SYSTEM_STATICTEXT.] txtName[ROLE_SYSTEM_TEXT.] txtDescription[ROLE_SYSTEM_TEXT.] Description: [ROLE_SYSTEM_STATICTEXT.] cboLocation[ROLE_SYSTEM_STATICTEXT.] cboLocation[ROLE_SYSTEM_STATICTEXT.] Options: [ROLE_SYSTEM_STATICTEXT.] radOption1[ROLE_SYSTEM_RADIOBUTTON.] radOption2[ROLE_SYSTEM_STATICTEXT.] Active: [ROLE_SYSTEM_STATICTEXT.] chkActive[ROLE_SYSTEM_STATICTEXT.] 			~	StateText Primary ChildCount ChildCount ChildId DefaultAction Description HWND IndexInParent LocationRECT Name Role State Value	alert high 4 CHILDID_SE SetFocus txtDescription 2 (X=198,Y=242 txtDescription ROLE_SYST STATE_SYS	n[QWidget] 2,Width=28 n 'EM_TEXT	7,Heigl	ht=72}	~
 HWND Tree formLayoutWidget[QWidget] txtNotes[QWidget] Notes:[QWidget] txtDateTime[QWidget] Date/Ti[QWidget] txtDate[QWidget] txtTime[QWidget] Date/Ti[QWidget] cboFont[QWidget] Font:[QWidget] chkActive:[QWidget] 			~	Properties ProcessName ScreenHeight ScreenWidth ScreenY Size Primary AccessibleObject ClassName Rect Text Visible WindowStyle	C:\Users\Pub 1080 1920 0 287, 72 btDescription QWidget 198, 242, 287, btDescripti True WS_CLIPCH	72	YSTE	M_WINE	~

You can then use the <u>Accessible Spy</u> to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy to the list of learned objects that you can test against.

References

Testing Qt Framework Applications

2.2.10.1 Using JavaScript

Rapise includes support for testing applications written using the Qt Framework written using QWidget controls.

To ensure that Rapise can access the UI elements and properties in the Qt application being tested, <u>MSAA (Microsoft Active Accessibility) support for your Qt application must be enabled</u>. This provides additional information on Qt UI elements to automation software like Rapise and can be accomplished by shipping and loading the "Accessible Plug-in" included in the Qt SDK (Software Development Kit) with the Qt application under test (see below).

This tutorial illustrates the ability of Rapise to test such Qt applications using a sample application that already has the MSAA support added.

This version of the tutorial uses the JavaScript test editor option in Rapise. If you'd prefer to use the Rapise Visual Language (RVL), please go to the main <u>Tutorial</u> instead.

Testing the Sample Qt Application

On the <u>Start Page</u> of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

Then go to C:\Users\Public\Documents\Rapise\Samples\QtFramework and double-click on the QtWidgetApp.exe file to start the sample application:

If you have everything configured correctly, you will see:

🚺 Sample (QtWidget Application		_		×
<u>F</u> ile <u>V</u> iew					
Sample	e OtWidget Application for R				
Name:		Filename:			
Description:		Font:		Choose	e
		Color:	Chang	e Color	
Location:	Headquarters	Print:	Print		
Options: Active:	Option 1 Option 2 Is Active?	Input:	Diselse Serve		
Font:	MS Shell Dlg 2	Errors:	Display Errors.		
Date/Time:	12:00:00 AM ÷ 1/1/2000 ÷	Message:	Display Message	e	
Date/Time:	1/1/2000 12:00:00 AM				
Notes:					
	OK Cancel]			//

Reate New Test	×
Please specify the name of the new test:	
Specify the folder where this test should be created:	
C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\My Rapise Tests.sste	est
Rapise supports multiple testing methodologies. Select one to be used for newly created	ated test.
Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
	Cancel

Once the application is started, open up Rapise and click on **FILE** > Create New Test:

Enter the name "Qt Sample Test" as the name and choose **Basic: Windows Desktop Application** as the methodology. Then choose JavaScript as the scripting choice:

R Scripting Language	_		Х
Choose preferred scripting language:			
RVL (Rapise Visual Language)			
JavaScript (Advanced Users)			
Don't ask (may be changed later in Options\Settings\Default Scripting	g Languag	e)	

Once the test is created, you will see:

Flie Test Options Epit	Rapise/Dourt 5.0.7.33 (C\\taxes\adam.sandman\Documents\My Rapise Tests\TwoDialogsTest\TwoDialogsTest atres)	- 6 ×
the Company of the State of the	ge Spira Dashboard Manual Steps Record Spira Dashboard Bry Manual Steps Spira Properties Spira Properties Spira Properties Body Control Spira Properties Body Control Record Body Defluoging Operuging Torois Heip	
	Start Page 2 TwoDatogr Test,	
Ciged Tree Closes administrations/Docu Ciged Tree Closes administrations/Docu Close Functions Der Vanisher	<pre>Pruse 'Record/Learn' button to begin test recording Function Test() Function Test()</pre>	^
Count (Ple Sittings Popular 2011)		*
0	Mpi	
	Microsoft IVV Windows Script Hord Version 5.812 Copyright (C) Microsoft Corporation AI rights reserved Sea Recorder Starting, Valdose level:1 Recording scription Varentpadamsaandmat/Document/My Reprise Testo(Web Testing 1) is Recording scription Vice register * class register *	•

Click on the **Record** button to display the <u>"Select an Application to Record"</u> dialog:

	_					
Select From Running Applicatio	ns Run	Applicat	ion			
Applications	Application	n (C:\Us	ers\Public\Documents\Rapise\Samples\QtFram	nework\QtWidget	tApp.e	
Available Applications		PID	Path			
C:\WINDOWS\system32\cm	nd exe	18972	C:\Windows\System32\cmd.exe			
Calculator		5540	C:\Windows\System32\ApplicationFram			
Euge Groove: Still Euge - Jango ?			C:\Windows\System32\ApplicationFram			
Groove Music	-	5540	C:\Windows\System32\ApplicationFram			
🖸 Inbox - Adam Sandman (Inflectra)			C:\Program Files (x86)\Microsoft Office\r			
QtFramework	· · ·	33212	C:\Windows\explorer.exe			
Rapise.hmxp in C:\Git\Rapise\Rap						
RE: [leaders] Election outco			C:\Program Files (x86)\EC Software\Hel C:\Program Files (x86)\Microsoft Office\r			
RE: Monday Blog - Messag		1428	C:\Program Files (x86)\Microsoft Office\r			
Sample QtWidget Application		23496	C:\Users\Public\Documents\Rapise\Sa		- 1	
Test Java SWT Application		3120	C:\Program Files (x86)\Java\jre1.8.0_112			
=						
- Libraries						
Library	Descri	ption				
Infragistics	Infragis	stics sup	pport			
🗌 Telerik	Telerik	RadCor	ntrols for Winforms Q3 2010			
Syncfusion	Syncfu	usion for	Windows Forms controls support			
ActiveX	Suppo	rt for Act	iveX Libraries			
Qt Framework	Used f	or applic	cations written using the Qt Fram			
UIAutomation	Tests.	NET, W	PF and Silverlight Applications			
ActiveXC1	Suppo	rt for Co	mponent One 8.0 ActiveX Librari			
ActiveXSft	SftTree	e/OCX c	ontrol support			
Web Service	Used f	or testin	g RESTful web services			
SWT	UI Auto	omation	support for Java SWT			
3001	UI Automation support for Java SWT HTML DOM-based recorder for Internet Explorer				- E	
☐ SWT ☐ Internet Explorer HTML	HTML	DOM-ba	ised recorder for Internet Explorer			

Choose the **Sample QtWidget Application** from the list of running applications, change the library selection from Auto to "**Qt Framework**" and click **Select**.

Now in the sample application click on some of the Qt controls. Rapise will record the actions:

#	Object	Action	Data	Comment				
尚1	txtName_	SetText	Adam	Do SetText	("Adam") on txtNar	ne_		
2	radOption	SetCheck	True	Do SetCheck(true) on radOption2_		on2_		
103	chkActive_ SetCheck Tru		True	Do SetCheck(true) on chkActive_				
105	OK_	Action		Press button 'OK_'				
	Verify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel	

When you click Finish, you will see the recorded test script and learned objects:

File Test Options E	Edit					
🎦 🗁 Open	🖾 Start Pa	Page 🛐 Spira Dashboard)	No Debugging 🔹	About Convert to NeoLoad	
Save to Spira	遼 Test	Manual Steps	Record Play	Level 1 (Minimal) 🔹	💡 Activation 🛛 📓 Spira Properties	Find in Help
💣 Open Test from Spira	🕃 Functio	ions 👍 Web Services	Spy		🔢 Object Mgr	Files
File		Shortcuts	Record and Execute	Debugging	Tools	Help
Objects		🖾 Start Page 👔 Qt Sample	Test.j			
√ id n idH		1				
🔺 💟 Object Tree C:\Users\adam.sand	lman\Do	2				
Sample QtWidget Application		3 //#########	# Script Steps ###	**********		
ChkActive_[chkActive_]		4				
▶ ⓐ OK_[OK_]		5 function Te	st()			
• • radOption2_[radOption2_]		6 🗏 (
weil txtName_[txtName_] Global		7				
Global Global Global Global Global					, 2016 4:03:38 PM: ======	
Var. User Variables			tText("Adam") on t	_		
Var Ober Variables			tName_').DoSetText tCheck(true) on ra			
			dOption2 ').DoSet(
			tCheck(true) on ch			
			kActive ').DoSetCh			
			button 'OK '			
			').DoAction();			
		17				
		18 L}				
		19				
		20 g_load_libra	aries=["Qt Framewo	ork"];		
Obiesta Eilea Cattinga		21				
Objects Files Settings		22				

You can drag and drop any of the learned objects from the left-hand pane into the main test script. You can also just type **SeS("OK_")** (for example) and Rapise will display the list of available functions. When you click **Play**, Rapise will play back your test script against the application:

-			by that column.		
#	Туре	Start	^ Name	Status	Comment
* 🔺	A	-			
	Message	16:04:25.733	Starting scenario: Test	Info	
	Assert	16:04:28.764	txtNameDoSetText(["Adam"])	Pass	Returned Value: true
	Assert	16:04:30.682	radOption2DoSetCheck([true])	Pass	Returned Value: true
	Assert	16:04:32.308	chkActiveDoSetCheck([true])	Pass	Returned Value: true
	Assert	16:04:34.105	OKDoAction([])	Pass	Returned Value: true
×	Test	16:04:34.112	Qt Sample Test	Pass	Passed:4 Failed:0
			🕳 Test Pass		

Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

The Spy tool lets you see the objects in the application in a hierarchy that you can learn.

When you are in the middle of recording, click on the **Spy** button and Rapise will display the <u>Accessible</u> <u>Spy</u>:

Rapise Spy				- 0	×
Start Tracking (Ctrl-G)	Learn Object			Ħ	3
Accessible					
Tree			- Properties		
[ROLE_SYSTEM_CLIENT.]			StateText	alert high	
Name: [ROLE_SYSTEM_STATIC	TEXT.]		Primary		
txtName[ROLE_SYSTEM_TEXT.] txtDescription[ROLE_SYSTEM_TEXT.]			ChildCount	4	
			ChildId	CHILDID_SELF	
Description: [ROLE_SYSTEM_STATICTEXT,]			DefaultAction	SetFocus	
cboLocation[ROLE_SYSTEM_C			Description		
Location: [ROLE_SYSTEM_STATICTEXT,]			HWND	txtDescription[QWidget]	
Options: [ROLE_SYSTEM_STATI			IndexInParent	2	
			LocationRECT	{X=198,Y=242,Width=287,Height=72	2}
radOption1[ROLE_SYSTEM_RAD			Name	txtDescription	
radOption2[ROLE_SYSTEM_RAD			Role	ROLE_SYSTEM_TEXT	
Active:[ROLE_SYSTEM_STATICTEXT,]			State	STATE_SYSTEM_ALERT_HIGH	
chkActive[ROLE_SYSTEM_CHEC	CKBUTTON,]	Ψ.	Value		
HWND					
Tree			- Properties		
formLayoutWidget[QWidget]			ProcessName	C:\Users\Public\Documents\Rapis	e\S
txtNotes[QWidget]			ScreenHeight	1080	
Notes:[QWidget]			ScreenWidth	1920	
txtDateTime[QWidget]			ScreenX	0	
Date/Ti[QWidget]			ScreenY	0	
txtDate[QWidget]			> Size	287, 72	
txtTime[QWidget]			Primary		
			AccessibleObject	txtDescription [ROLE_SYSTEM_W	/INE
Date/Ti[QWidget]			ClassName	QWidget	
cboFont[QWidget]			> Rect	198, 242, 287, 72	
Font:[QWidget]			Text	txtDescription	
chkActive[QWidget]			Visible	True	
Active:[QWidget]		-	WindowStyle	WS_CLIPCHILDREN, WS_CLIPSIE	BLIN

You can then use the <u>Accessible Spy</u> to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy to the list of learned objects that you can test against.

References

• Testing Qt Framework Applications

2.3 Features

Rapise is a feature-rich test automation system, however all the features have been designed to make test automation as easy as possible.

Most of the features of Rapise fall into one of five categories:

- 1. Building test scripts with little or no manual scripting.
- 2. Reading and interpreting results and reports.
- 3. Additional features and capabilities for sophisticated testing.
- 4. Writing more involved or complicated tests using scripting.
- 5. Extending Rapise to learn new or extended libraries of capabilities.

Depending on the application set being tested, not all of these features are necessarily needed for every situation.

For each feature, this document describes:

- 1. The reason you might use a given feature.
- 2. A summary of the basic value of the feature.
- 3. An overview of how the feature works from the perspective of using it.
- 4. At least one useful sample that demonstrates how to use the feature.

2.3.1 Recording and Learning

Purpose

To understand what different objects might be found on a UI screen, and how to recognize them, record their characteristics and interact with them using Rapise.

Value

A UI screen entity (object) may consist of many different parts and components. Actions on these objects, and usage of these controls, must be captured in different ways, depending on the properties of the object. Rapise provides five fundamental methods for capturing objects and corresponding user actions:

- 1. **Recording** Rapise is able to track user interactions with AUT and automatically capture affected objects and corresponding user actions. See <u>Recording</u> for more information.
- 2. Learning there are cases when it is not necessary or is not possible to track user interactions with AUT. In this case user can manually point to an object that should be captured by Rapise. See Learning for more information.
- 3. Analog Recording (Absolute/Relative) this is for objects that are not standard in some important way, and so activity on them cannot be captured using recording or cannot be specified after learning. Absolute Analog Recording is used to track mouse usage (movement and clicks) and keyboard events. For absolute analog recording, the positions these events are recorded relative to the top-left corner of the system screen. (In contrast, in <u>Relative Analog Recording</u>, the events are recorded in a file of type arf (Analog Recording File).
- 4. Simulated Object Recording a Rapise user can use simulated objects when some objects are not natively supported by Rapise (e.g. their internal structure, properties and actions are unknown). In this case, what is recorded are mouse clicks and keyboard activity. Compare to Analog Recording when all mouse and keyboard actions are recorded, including mouse up/down, mouse move events. See <u>Simulated Objects</u> for more information.
- 5. Manual Recording In addition to providing automated testing functionality, Rapise enables you to <u>create manual tests</u> (ones that will be <u>carried out by a human tester</u>) rapidly without having to laboriously enter in test steps and screenshots by hand. It does this by using the same recording mechanism used for automated testing to <u>create a manual test case</u> that contains a list of the tester's interactions and screenshots of what was performed. This is useful for exploratory testing and is a huge time saver.

Usage

Before an operation (press, enter text, select, click, etc.) can be performed on an object automatically, Rapise must be able to identify the object. That identification must be able to locate the object definitively, and it must be able to duplicate the action or operation precisely. This carries several implications. Firstly, if the AUT is in a different position on the screen when it is started, Rapise must still be able to find the objects in the AUT window. Secondly, if the positioning of objects on the AUT window is proportional or relative to the screen size of shape, Rapise must still be able to locate the object.

A secondary set of considerations relates to the fact that the AUT UI layout maybe sensitive to the context of the state of the application. For example, consider the case of a word processor. Pressing the "bold" button doesn't predict what the result will be unless it is known whether the text highlighted is currently bold or not. A far more illustrative example is that of the Microsoft Paint utility. The Microsoft Paint utility is the subject of a Inflectra sample, <u>Simulated Object</u>.

The most instructive way to identify the objects to Rapise is to practice with the tool and different types of objects. The best methodology to use is as follows:

- 1. First, try to use Record/Learn to learn the object and record actions in a single step.
- 2. If learning.recording fails to record actions in the grid, use the <u>Object Spy</u> to observe the object carefully and discover what libraries and classes are being used by the AUT.
- 3. Use Verify (Ctrl+1) from the Recording Activity dialog to get summary information about the object.
- 4. Use a more appropriate set of libraries when selecting the AUT prior to recording.
- 5. Use Analog Recording with absolute positioning to identify and locate the object.
- 6. Use Analog Recording with relative positioning to identify and locate the object.
- 7. Use Simulated Object Recording to track the actions required and at the positions required.
- 8. Look for custom libraries that support the technology being used by the AUT.
- 9. Build your own custom library to support the technology in use by the AUT.
- 10. Finally if it will not be worth developing automated tests for this AUT, use the <u>manual recording</u> <u>feature</u> to speed up your manual test writing.

2.3.1.1 Recording

Purpose

Recording is the name given to having Rapise track your interactions with an application.

Value

The actions you take in using the <u>AUT</u> are observed by Rapise and are transformed into a script (javascript), which you can execute using the Play button. The script can be extended and modified to suit special purposes.

Usage

The **Recording Activity (RA) Dialog** is opened when you start recording using the Record/Learn button. When the Recording Activity dialog appears, Rapise has connected to your AUT and is ready to monitor and record your interactions. You'll find instructions <u>here</u> or look at one of the examples - <u>TwoDialogs</u>, <u>Sample Record and</u> <u>Playback</u>, or <u>Mobile Sample</u>

			Data	Comment			
V	'erify	Learn		SPY	Pause	Finish	Cancel

You'll notice that the RA dialog has a grid. As you interact with the AUT, your actions will be listed in the grid.

If you record an incorrect action, you can right-click on the action and delete it.

To ensure you successfully record your interaction with the AUT, navigate slowly through the AUT. Wait a second or two between each action to make sure Rapise has time to interpret and record your action. Once your interaction is updated in the RA dialog grid, you are free to continue with the next action.

When you are done recording, press the **[Finish]** button on the RA dialog or type **Ctrl+3**. The RA dialog will disappear, and you will see an automatically generated script opened in Rapise.

For **Mobile Testing**, you will need to use the **[Spy]** button which then allows you to pick a specific object from the <u>Mobile Spy</u>:

Rec	ording Activ	vity for "Dev	/ice"				
#	Object	Action	Data	Comment			
	Verify (Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel
Paus	ed					Advanced>>	Transparent

See also

- If you have already recorded a script and want to record additional interactions in the same test, be sure to read <u>Making Multiple Recordings</u>.
- The RA dialog is described more thoroughly in <u>Recording Activity Dialog</u>.
- To learn how to run the script, see <u>Playback</u>. To learn how to modify the script, see <u>Scripting</u>.

- For a detailed tutorial, see <u>Tutorial: Record and Playback</u> in the Getting Started section.
- For more information on the Spy (ObjectSpy) capability, see Object Spy.

2.3.1.2 Learning

Purpose

Objects are the controls and items on the screen of the AUT. "Learning" an object refers to the process of Rapise collecting enough information about the on-screen item to be able to reference the item when the test script is run without ambiguity and regardless of its location on the UI.

Value

When Rapise "learns" an object, it records the object's type, its name and how to find the object again (locator). It saves everything it learns to the script so that the object can be identified when the test is run. Rapise gives the object a simple name so that you can easily refer to it later if you decide to modify the script.

Usage

Objects are learned in three ways: (1) during recording, (2) explicitly using Learn, (3) using the <u>Spy</u> tool.

Recording

During a Recording session, Rapise learns about each object with which you interact. For details, see <u>Recording</u>.

Explicitly

- 1. Open the Recording Activity Dialog. Instructions are HERE.
- 2. Place your mouse over the object you wish to learn. It should become surrounded by a purple box.
- 3. Press CTRL+2.
- 4. You will see a new entry in the Recording Activity Dialog, signifying that the object was learned.

Spy Tool

During a Recording session, you can also click on the **Spy** button inspect the object hierarchy of the application you are testing. The Spy lets you pick specific objects that might not be visible on the screen, or may be parent/child objects of the one highlighted. When you have found the correct object in the Spy, you can use the **Learn button** inside the Spy to add it to Rapise's learned object list.

The Object Tree

Regardless of how you learn the object, it will be displayed in the Object tree:

Objects
√ id n idB
Object Tree C:\Users\Public\Documents\Rapise\Samples\C
Library Information System Log In - Internet Explorer
Inflectra Library Information System - Internet Explorer
Author_[Author:]
A Book_Management [Book Management]
ab ctl00\$MainContent\$btnSubmit [ctl00\$MainContent\$bt
Genre_[Genre:]
A Log_In [Log In]
A Log_Out [Log Out]
Image: Name_[Name:]
Library Information System Book Management - Interne
Global
Ø User Functions
var var User Variables
•
Objects Files Settings

Each object has various properties and methods / actions that can be performed on it. Expanding the object name lets you see the available functions and properties:

Objects
√ id n id∄
🗾 Object Tree C:\Users\Public\Documents\Rapise\Sample 🛛 🔺
Library Information System Log In - Internet Explore
Inflectra Library Information System - Internet Explo
▶ In Author_ [Author:]
A Book_Management [Book Management]
=♦ DoAction
■ DoAddSelection
=∳ DoAnalogPlay
=♦ DoClick
=♦ DoDblClick
DoDOMChildAt
DoDOMChildrenCount
Sobolic Solution
DoDOMFindParentWithAttribute
DoDOMGetAttribute
DoDOMGetAttributes
■ DoDOMNextSibling
Sobometric states and states
SobomerevSibling
•
Objects Files Settings

Everything Rapise learns about an object is saved in **saved_script_objects**. You can see this variable defined in the <project-name> objects.js file that will be listed in the Test Files tab of the Rapise. The following shows what Rapise saved about the "Please enter your name" text box in the <u>TwoDialogs</u> example:

```
Please_enter_your_name_:{
"locations": [
     {
          "locator_name": "Location",
          "location": {
               "location": "4.4",
               "window_name": "param:window_text",
               "window_class": "param:window_class"
          }
     },
     {
          "locator_name": "LocationPath",
          "location": {
               "window_name": "param:window_text",
               "window_class": "param:window_class",
               "path": [
                    {
                          "object_name": "param:object_name",
```

```
"object_class": "param:object_class",
                              "object_role": "param:object_role"
                         },
                         {
                              "object_name": "param:window_text",
                              "object_class": "param:window_class",
                              "object_role": "ROLE_SYSTEM_DIALOG"
                         }
                   ]
              }
         },
         {
              "locator_name": "LocationRect",
              "location": {
                   "window_name": "param:window_text",
                    "window_class": "param:window_class",
                   "rect": {
                         "object_name": "param:object_name",
                         "object_class": "param:object_class",
                         "object_role": "param:object_role",
                         "x": 222,
                         "y": 40,
                         "w": 140,
                         "h": 23
                   }
              }
         }
    ],
    "window_text": "Inflectra Rapise Two Dialogs Sample",
    "window_class": "#32770",
    "object_text": "Chris",
    "object_role": "ROLE_SYSTEM_WINDOW",
    "object_class": "Edit",
    "object_name": "Please enter your name:",
    "version": 0,
    "object_type": "Win32Text",
    "object_flavor": "Text",
    "object_library": "Generic"
},
. . .
```

See Also

<u>Recording</u>

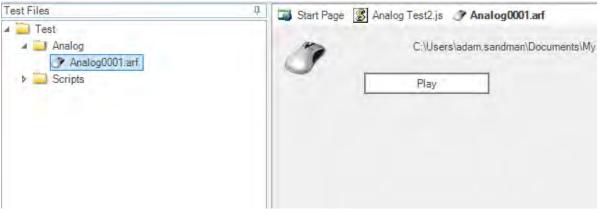
- Learning invisible and <u>Simulated Objects</u> is slightly more complicated. You can find information on both in the <u>Recording Activity Dialog</u> section. Look for descriptions of the **Pick Object** button and the _**Simulated** drop-down menu.
- Learn Object

2.3.1.3 Analog Recording

Concept

Sometimes you have to automate the testing of an application that contains some controls or elements that are not standard objects that can be recognized by Rapise. For example you may have a drawing canvas inside an application that allows you to annotate a diagram. You can use the standard Rapise libraries for the rest of the controls but the actual drawing events cannot be captured that way. Analog recording is available to 'fill in the gaps' in such scenarios.

During **Analog Recording**, Rapise records mouse movements, keyboard inputs, and clicks and stores them in a special .ARF (Analog Recording File) format file:



There are two types of Analog Recording: Absolute and Relative.

- <u>Absolute</u>: Mouse coordinates are recorded relative to the top left corner of the screen.
- <u>Relative</u>: Mouse coordinates are recorded relative to the top left corner of the object beneath the mouse cursor.

Usage

When you are <u>recording your test</u> using the application you may come to a point where there are user actions that you need to record that don't have any identifiable objects (for example drawing a signature). You can click on the 'Analog' button on the recorder to engage Analog mode:

Ħ	Object	Action	Data	Comment	
			Press Ctrl+Br	eak to Stop Analog Recording	

Now when you use the mouse and keyboard to test the application, Rapise is storing the coordinates of your mouse clicks and keyboard events and storing them in a separate .ARF file that is part of your test project.

ŧ	Object	Action	Data	Comment		
包1	Global	Analog	Analog\Ana	Plays recorded e	events from "Analog∖	Analog0001.arf" file
Veri	fy (Ctrl+1)	Learn (Ctrl	(+2)	Biper Shit S	Pul Caasi	Pause

Once completed, the entire analog section is included as one step within the complete test script so you can include an analog sequence within a test script that contains other non-analog events. This lets you have the robustness of true object-based recording for 95% of your test and analog when you need it for the remaining 5%. This is the best of both worlds.

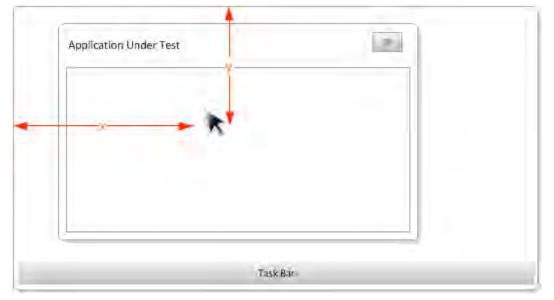
See Also

<u>Recording Activity Dialog</u>

2.3.1.3.1 Absolute Analog Recording

Purpose

Absolute analog recording is used to track mouse usage (movement and clicks) and keyboard events. For absolute analog recording, the positions these events are recorded relative to the top-left corner of the system screen. (In contrast, in relative analog, the events are recorded relative to the upper-left corner of the selected objects.) The events are recorded in a file of type arf (Analog Recording File).



Value

Not all applications can be recorded by locating and learning objects being used. A very good example of this is free-hand drawing in an application such as Microsoft Paint (Start Menu -> Accessories -> Paint). There are several reasons why this application cannot be recorded using object tracking, learning and recording. The most important is that when the mouse is moved free-hand, it is operating on the same object the whole time - the blank "canvas." Another reason is that the application changes behaviour and the positions of the canvas change depending on the size of the canvas and the positions of floating toolbars.

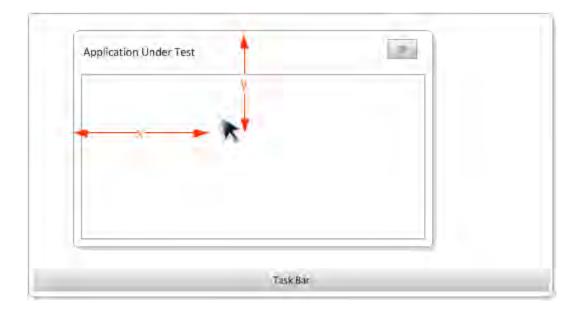
Absolute analog recording is provided by Rapise to make it possible to make it possible to design and implement tests for these types of applications.

See Also

- Do Absolute Analog Recording
- <u>Relative Analog Recording</u>
- 2.3.1.3.2 Relative Analog Recording

Purpose

Relative analog recording is used to track mouse usage (movement and clicks) and keyboard events. For relative analog recording, events are recorded in relation to the top-left corner of the application's window. The events are recorded in a file of type arf (Analog Recording File).



Value

Not all applications can be recorded by locating and learning objects being used. A very good example of this is free-hand drawing in an application such as Microsoft Paint (Start Menu -> Accessories -> Paint). There are several reasons why this application cannot be recorded using object tracking, learning and recording. The most important is that when the mouse is moved free-hand, it is operating on the same object the whole time - the blank "canvas." Another reason is that the application changes behaviour and the positions of the canvas change depending on the size of the canvas and the positions of floating toolbars.

Relative analog recording is provided by Rapise to make it possible to make it possible to design and implement tests for these types of applications.

See Also

- Do Relative Analog Recording
- <u>Absolute Analog Recording</u>

2.3.1.4 Simulated Objects

Purpose

During normal recording, Rapise <u>Learns about the Objects</u> you interact with. If, for some reason, Rapise cannot learn an object, you can create a **Simulated Object**. Rapise identifies a simulated object by its location in the Window or Dialog and can perform certain generic actions on it, such as Click and Fill In. This works in the reverse sense also. That is, if Rapise cannot identify an object, or, for example, you click outside any defined object in the AUT's UI, Rapise will create a simulated object to represent the action.

Value

Not all objects on a screen are "standard" or can be recognized by the libraries loaded. Some are compound objects, consisting of two or more individual objects that work together to deliver a UI effect or behaviour. Simulated objects "fill in the blanks" to allow Rapise to cause an event outside the

normal set of objects.

See Also

- <u>Recording Activity Dialog</u>
- <u>Sample Tests</u>: The SimulatedObject sample.
- Deal with a Simulated Object

2.3.1.5 Object Libraries

Purpose

Object libraries define what objects and interactions Rapise understands during <u>Recording</u> and <u>Learning</u>. Most Object Libraries are specific to an application or a set of applications.

Usage

Rapise comes with several different object libraries:

- 1. Auto
- 2. Core Technologies
 - o Generic
 - o Internet Explorer HTML
 - o Firefox HTML
 - o Java
 - o Java SWT
 - o Managed
 - o UI Automation
 - o Qt Framework
 - o ActiveX
 - o Web Services
 - o User
 - o Advanced Accessibility
 - o Console
- 3. Mobile Libraries
 - Android (via. Appium)
 - o iOS (via. Applium)
- 4. Widget Toolkits
 - o DOM Dynamics CRM
 - **O DOM GWT**
 - o DOM GWT-Ext
 - o DOM SmartGWT
 - o DOM YUI
 - o DOM jQuery UI
 - o DOM Salesforce
 - o HTML 5
 - o DevExpress
 - o Infragistics
 - o Telerik
 - o ActiveX ComponentOne

- o SyncFusion
- o FarPoint
- o Dynamics AX

You can add your own Recording library--one that understands the objects in your application.

- Selecting Auto as the application recording library will cause Rapise to select the one that it deems is most appropriate.
- UlAutomation: Use this library with .NET, WPF, and Silverlight applications. When used with .NET 2.0+ applications you should also include the **Managed** library as well. When used with older .NET applications, you should use the Generic library instead.
- Internet Explorer HTML, Chrome HTML and Firefox HTML are used with Internet Explorer, Google Chrome and Firefox respectively. They understand only the DOM (document object model) and therefore capture interactions with the web application, not the browser. They also have access to passwords. Tests recorded with either of the libraries can be run in any of the three browsers. See <u>Cross Browser Testing</u> for more details.
- User refers to Custom Libraries.
- The DOM GWT library uses the Document Object Model to learn or record objects found in the Google Web Toolkit.
- The DOM GWT-Ext library uses the Document Object Model to learn or record objects found in the Google GWT-Ext library.
- The DOM SmartGWT library uses the Document Object Model to learn or record objects found in the Google SmartGWT library.
- The DOM jQuery UI library uses the Document Object Model to learn or record objects found in the jQuery UI widget library.
- The HTML5 library uses the Document Object Model to learn or record objects found in the HTML 5 extensions library.
- The **DOM YUI** library uses the Document Object Model to learn or record objects found in the Yahoo! User Interface library.
- The **Generic** library uses Microsoft's **MSAA** event model to capture user actions. The Generic library should be used if there is no library more specific to the AUT available. The Generic library will record a large set of applications, but it has drawbacks; it may skip some actions and/or record unintended actions. Passwords are not visible to the Generic library, and must be manually entered into the test after recording.
- The Advanced Accessibility library is for recording with Internet Explorer. In general, you will want to use the Internet Explorer HTML library. However, there is some information available through Advanced Accessibility that is unavailable when looking solely at the DOM. For example: the absolute screen position of an object. Advanced Accessibility is not precise, as Internet Explorer HTML is, and may miss actions or record unintended actions.
- The Java SWT library is for use with the Eclipse Java Standard Widget Toolkit (SWT) applications.
- The **Console** library is for use with Windows Console Applications that run in the command-line.
- The Java library is for use with Java GUI applications that are written using either AWT or SWING. Use the **SWT** library instead if your application was written using SWT.
- The Managed library is for use with Microsoft .NET 2.0 + applications. It adds some additional .NET 2.0+ specific-controls to the list supported in the Generic and UIAutomation libraries.
- The **DevExpress** library allows you to record and learn using the various controls provided in the DevExpress DXperience v1.0 component library. This allows you to save time by having the system recognize the various controls directly.
- The **Infragistics** library allows you to record and learn using the various controls provided in the Infragistics component library. This allows you to save time by having the system recognize the various controls directly.
- The **Telerik** library allows you to record and learn using the various controls provided in the Telerik RadControls for Winforms component library. This allows you to save time by having the system recognize the various controls directly.
- The Qt Framework library is for use with applications that are written using the cross-platform Qt

Framework.

- The Web Services library is for use with API tests that connect to either REST or SOAP web services. See the <u>web service testing</u> topic for more information.
- The Dynamics AX library is for use with the Microsoft Dynamics AX ERP package. See the Dynamics AX topic for more information
- The DOM Dynamics CRM library is for use with the Microsoft Dynamics CRM package. See the <u>Dynamics CRM</u> topic for more information

See Also

- <u>Recording</u>
- To write an Object library specific to your application, see <u>Custom Libraries</u>.
- Cross Browser Testing
- If you interact with an object that is not defined in your chosen recording library, it will be treated as a <u>Simulated Object</u>.

2.3.1.5.1 Custom Libraries

Purpose

If your application doesn't work with the predefined Recording Libraries, you can create your own.

Usage

Your library can provide **Basic** or **Full** support for your application. Basic support allows you to manually <u>Learn</u> objects, <u>write test scripts</u>, and <u>Playback</u> your scripts. Full support allows you to <u>Record</u> as well. Create your library in the LibUser directory. Unless you specified otherwise, you will find it at:

C:\Program Files\Inflectra\Rapise\Engine\Lib\LibUser.

Basic Support

Add a Matcher Rule to the library for every window type in your application. The SeSMatcherRule includes information to identify your application, and a set of behaviors.

```
{
    object_type: "yourAppObject",
    classname: "yourAppFrame", //You can use a regular expression here
    behavior: [yourAppBehavior]
})
```

Override Actions: Override actions in **yourAppBehavior** (above). The action definitions you provides will be used during <u>Playback</u>. Overriding actions does not affect recording.

```
actions: [{
    actionName: "Click",
    DoAction: function(){}
  },
  {
    actionName: "SetText",
    DoAction: function(/**String*/txt){}
 }]
```

Full Support

Enable Recording: You can enable recording in two ways. If your application notifies the Accessibility Events interface about application events, you can override events in the **behavior** section of **SeSMatcherRules**:

```
var newBehavior={
   actions: [{/*section deleted for brevity*/}],
   events:
   {
    OnSelect: function(/**SeSObject*/ param, /**Boolean*/ badd)
    { /*...*/
    },
   OnSelectAdd: function(/**SeSObject*/ param, /**Boolean*/ badd)
    { /*...*/
    }
}
var newRule = new SeSMatcherRule({
    object_type: "someType",
    role: "someRole",
    behavior: [newBehavior],
```

})

Otherwise, you will have to implement **Custom Recording**.

Custom Recording: With custom recording, it is the library's responsibility to:

- detect user actions in the application, and
- call RegisterAction() (which writes the action to the script).

See Also

- To see what actions and events can be overridden, see SeSBehavior.js (in the Rapise Engine).
- Check the Engine/Lib directory for examples.
- You can alter the behavior of an action without creating an entire library. See the <u>Actions</u> section for more details.

```
2.3.1.5.1.1 Actions
```

Purpose

Actions are anything the user can do to a GUI control, such as click, select, fill with text, etc. You can override the behavior of an action, without creating or altering a <u>Recording Library</u>, using **SeSExtendAction()**. Overriding an action affects <u>Playback</u>, but not <u>Recording</u>.

Usage

SeSExtendAction() is used to override an action handler or add a new DoAction handler:

function SeSExtendAction(objectType, doActionName, replacementFunction)

where:

- objectType is the name or <u>regular expression</u> specifying the object type(s) for which this extension should apply.
- doActionName is the name or <u>regular expression</u> specifying the DoAction handler that should be overridden.

• replacementFunction is the function containing overriding behavior.

In most cases SeSExtendAction() should be called from within TestInit().

Calling Base Actions The function you are overriding is called the **BaseAction**. You can call it like this:

this.BaseAction(arguments);

You may override actions several times. For example:

function DoActionB() { this.BaseAction(); function DoActionC() this.BaseAction(); SeSExtendAction("Win32Button", "DoAction", DoActionB); SeSExtendAction("Win32Button", "DoAction", DoActionC);

When DoAction is called for the Win32Button, the following sequence is executed:

```
DoActionC->DoActionB->DoAction
```

See Also

To see what actions can be extended, look in SeSBehavior.js (in the Rapise Engine).

2.3.1.6 Multiple Recordings

Purpose

Every time you record, the script recorder updates your test script. Be cautious about what changes you make to the test script; some changes could be lost if the recorder is re-run (see **Usage**).

Usage

The test script path can be found in the <u>Settings Dialog</u> under **Settings** > **ScriptPath**. Unless you specify otherwise, the test script is named *testname*.js (where *testname* is whatever you named your test).

Note that the Script Recorder only has knowledge of four functions and two data structures:

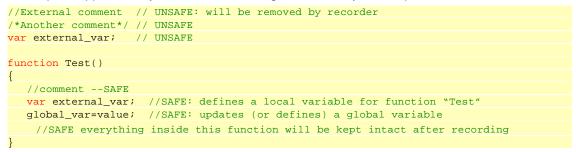
- 1. function Test()
- 2. function TestInit()
- 3. function TestFinish()
- 4. function TestPrepare()
- 5. array "g_load_libraries"
- 6. map "saved_script_objects"

You can make changes to the body of any of the above functions, and you can alter the initialization of **g_load_libraries** and **saved_script_objects**. All other changes are unsafe.

During Recording, the Script Recorder:

- 1. Appends newly recorded actions to the Test() function
- 2. Appends newly encountered objects to the saved_script_objects array
- 3. Updates g_load_libraries to reflect the library selections you made in the <u>Select an Application to</u> <u>Record... Dialog</u>
- 4. Ignores (and leaves intact) the definitions of TestInit(), TestFinish(), and TestPrepare()

For example, suppose that you have the following code inside your script file:



The parts of code marked UNSAFE will be deleted by the script recorder.

See Also

- <u>Settings Dialog</u>
- Select an Application to Record... Dialog
- <u>Recording</u>

2.3.1.7 Object Spy

Purpose

The **Object Spy** allows you to inspect an object's properties and state.

Value

Many controls on <u>User Interfaces</u> are compound objects or there may be many instances of a similar object. To be sure to select precisely the correct object, or to select the correct object from a collection of similar objects, the object's properties can be used to further identify the correct instance.

Usage

To spy on an Object:

1. Choose the type of **Object Spy** that you want to use. This can be done by clicking the down-arrow next to the Spy icon in the <u>Test ribbon</u> (Record and Execute tab):

	2	Ŧ	
	O Acces	Ob	ject
<u>'</u>			d Objec
	Mobile	e C	Object
	O UIAut	om	nation O

Web Object

There are five types of Spy available:

biect

- <u>Accessible</u> This is used to inspect applications that expose their properties using the Microsoft Active Accessibility (MSAA) technology. This is typically used by applications written in MFC, ATL, Qt, C++ and Visual Basic.
- 2. <u>Java Object</u> This is used to inspect applications written using the Java AWT and Swing UI frameworks.
- Managed Object This is used to inspect applications written in .NET 1.1, .NET 2.0, .NET 4.0 using Microsoft Windows Forms.
- 4. <u>Mobile Object</u> This is used to inspect mobile applications running on iOS or Android devices as well as the iOS or Android simulator
- <u>UIAutomation Object</u> This is used to inspect applications that expose their properties using the Microsoft's newer UIAutomation technology. This is typically used by applications written in WPF, Silverlight and Java SWT.

For more details on each Spy type, refer to specific topic above or view the <u>Spy Dialog</u> help topic.

- 2. Open the **Object Spy Dialog**. This can be done directly using the **Spy** button in the main Rapise test ribbon, or by pressing the **Spy** button in the <u>Recording Activity</u> dialog during recording or learning.
- 3. Press the Start Tracking button (or type CTRL+G).
- 4. As you mouse over different objects, you will see the contents of the Object Spy dialog change as it collects information about the object.
- Mouse over the object you wish to spy on and press CTRL+G. The reduced-size tracking dialog will be expanded into the the larger <u>Object Spy Diaog</u> dialog, presenting all the available information for the object.

See Also

• See the Object Spy Dialog for more details.

2.3.1.7.1 Accessible (MSAA) Spy

Purpose

The **Accessible Spy** is used to inspect applications that contain Microsoft Active Accessible (MSAA) objects.

- Tree		- Properties		
[ROLE_SYSTEM_CELL.]		State Text		
Screenshot[ROLE_SYSTEM_TEXT,]		A Primary		
select an application to record dialog[ROLE_SYSTEM_GRAPHIC,mk:@MSITStore:C:\User		ChildCount	56	
[ROLE_SYSTEM_TEXT.]	-	ChildId	CHILDID_SELF	
Purpose[ROLE_SYSTEM_TEXT.]		DefaultAction		
The [ROLE_SYSTEM_TEXT.]		Description		
Select an Application to Record[ROLE_SYSTEM_TEXT.]		HWND	[Internet Explorer_Server]	100
(SAR) Dialog appears before [ROLE_SYSTEM_TEXT.]		IndexInParent	0	
Recording[ROLE_SYSTEM_LINK.mk:@MSITStore:C:\Users\Public\Documents\Rapise\Hel		Location RECT	(X=246,Y=108,Width=1101,Height=1966)	
takes place. It queries the user for which program to record, as well as what [ROLE_SYST		Name		*
Recording Library[ROLE_SYSTEM_LINK,mk:@MSITStore:C:\Users\Public\Documents\Ra		Role	ROLE_SYSTEM_CELL	
to use, [ROLE_SYSTEM_TEXT,]		State	0	
(ROLE_SYSTEM_TEXT.)	+	Value		-

Features

The Accessible Spy has the following features:

- The Tree pane lets you view the hierarchy of MSAA objects available in the application
- The Properties pane lets you view the exposed properties of the highlighted MSAA object
- The Learn Object option is displayed when you use the Spy during recording and lets you pick specific objects to learn.

Commands

In addition to viewing the object hierarchy and object properties, you can perform the following tasks:

- Parent This selects the parent object of the one displayed
- Highlight This will attempt to Flash (highlight with a red rectangle) the object selected in the Spy.
- **Refresh** this simply refreshes the Spy view to reflect any changes that might have occurred in the application.
- Default Action this will perform the default action on the selected object in the Spy
- Mouse Click this will perform a simple mouse click on the selected object in the Spy
- Save to File this will save the properties of the currently selected object to a text file.

2.3.1.7.2 Java Spy

Purpose

The Java Spy is used to inspect applications that contain Java (Swing / AWT) objects.

– Java Object			
- Tree	- Properties		
 panel0[java.awt.Panel]	Primary		Т
label0[java.awt.Label]	Handle	790046	1
textArea1[java.awt.TextArea]	IndexInParent	0	
···· list1[java.awt.List]	Name	panel0	
	VmID	463120	
button1[java.awt.Button]	Secondary		
button2[java.awt.Button]	accessibleContext	java.awt.Panel\$AccessibleAWTPanel@df8ff1	
advAwtControls[java.awt.Button]	alignmentX	0.5	
	alignmentY	0.5	
	background	java.awt.Color[r=200,g=220,b=255]	
choice2[java.awt.Choice]	component		
	componentCount	9	
	components	[Ljava.awt.Component;@cf40f5	

Features

The Java Spy has the following features:

- The Tree pane lets you view the hierarchy of Java objects available in the application
- The Properties pane lets you view the exposed properties of the highlighted Java object
- The Learn Object option is displayed when you use the Spy during recording and lets you pick specific <u>objects to learn</u>.

Commands

In addition to viewing the object hierarchy and object properties, you can perform the following tasks:

- Parent This selects the parent object of the one displayed
- Highlight This will attempt to Flash (highlight with a red rectangle) the object selected in the Spy.
- **Refresh** this simply refreshes the Spy view to reflect any changes that might have occurred in the application.
- Save to File this will save the properties of the currently selected object to a text file.

2.3.1.7.3 Mobile Spy

Purpose

The **Mobile Spy** is used to inspect applications running on connected Mobile Devices (e.g. Apple iOS and Android devices).

Get Snapshot		Select Profile android-nexus	7	EAPP +
(Ctrl-G) Connection	Device	Profile	Context	
 (1) android wid (1) android.widget.FrameL (0) android.widget.Line (0) android.widget. 	.1205] ayout[0,33][800,108] neLayout[0,33][800,108] ew[0,33][800,108] get.LinearLayout[11,33][2 get.LinearLayout[715,33][ayout[0,108][800,1205]	30 Title XPath I J Layout 45 Height	0 android.widget.FrameLayout (0).android.widget.FrameLayout[0.0][//android.widget.FrameLayout[1] 1205 800 0 0 7 False false false false false false false false false false	Contractions Contr

The **Mobile Spy** dialog shows a snapshot of the screen displayed on the connected Mobile device as well as the properties of the currently selected object. You can selected the object either by clicking on the screen snapshot or the control hierarchy displayed to the left. The properties displayed will depend on the type of mobile device being tested (iOS vs. Android).

Tree

The spied upon object and its children are displayed here. When you click on an object it will also be highlighted in the **snapshot** view to the right.

Properties

Object fields and field values are displayed here.

Snapshot

This displays a snapshot of what is displayed on the mobile device being tested. The objects in the snapshot are clickable, which allows you to visually select objects from the hierarchy.

Tools

- Get Snapshot (CTRL + G) This will connect to the mobile device and get the latest snapshot from the mobile device and display in the right-hand window.
- Disconnect This option disconnects the Spy from the mobile device and ends the connection.
- Learn Object This option is only displayed in Recording mode and lets you take the currently selected object and add it to the <u>Object Tree</u> for the current test. It can then be used as a scriptable object in the test script.
- **Page Source** This lets you view the source of the mobile device in a text editor such as Notepad. It will show the objects in the treeview represented as an XML document.
- Go to URL This will instruct the mobile device to navigate its built in web browser to a specific URL.
- **Test Locator** This will display the <u>Mobile Test Locator</u> dialog box that lets you try different locators to resolve specific objects in the object hierarchy. It will include options such as using XPath and IDs.
- Select Profile This lets you change the profile of the mobile device you are testing while the Spy dialog is open.

- Edit Profiles This will open up the <u>Mobile Settings</u> dialog box. You cannot be connected to do this.
- Context This will display either 'Discovery Mode' or 'Recording Mode'.

Events Tab

L

The **Mobile Spy** also includes an **Events** ribbon tab that lets you send events to mobile device from Rapise, as if you were actually performing them on the device:

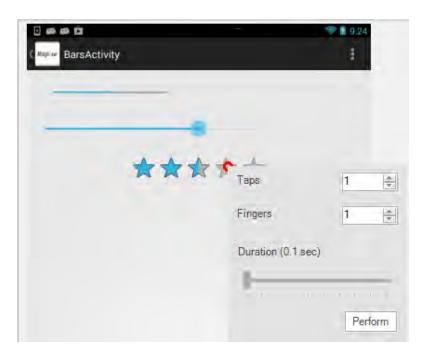
Spy	Events			
Тар	Precise Tap 🔹	Text		Accept Alert
Swipe 🝷	Scroll To	Send Keys		Dismiss Alert
Shake		Execute Script		Change Orientation
Т	ouch		Text	Misc

This dialog lets you perform the following events on the device:

- Tap this will simulate tapping the currently selected object on the device
- Swipe you specify the start and end points of the swipe operation. This is useful for simulating a real swipe on the device in a specific direction at a specific location (e.g. on a progress selector)

and and and and a second secon	_	🤝 🕻 925	
*	****	Fingers Duration (0.1 sec)	1 *

- Shake for devices that support it (e.g. iOS) this simulates shaking the device physically
- **Precise Tap** you specific the specific location on the screen within the bounds of the current object that you will be simulating a tap.



- Scroll To simulates scrolling to the selected object in the device object tree (which may not be visible).
- Text / Send Keys to use this, enter in text in the text box and click 'Send Keys', this sends text to the currently selected object as if you were using the virtual keyboard on the device.
- Accept Alert if you have a popup alert on the device, this simulates accepting it
- Dismiss Alert if you have a popup alert on the device, this simulates dismissing it
- Change Orientation for devices that support it, this simulates changing the orientation of the device from landscape to portrait (or vice-versa)
- Execute Script this is not currently supported and is for future functionality

2.3.1.7.4 Managed (.NET) Spy

Purpose

The **Managed Spy** is used to inspect Microsoft .NET applications that contain .NET framework objects (e.g. using Windows Forms).

	E	Properties		
	Ξ	ManagedSpy Properties	3	
		Component Type		
		Handle	1180844	
-		IsManaged	True	
	Ξ			
			scripti	
		1000000010110110	D ()	
			Default	
		, white open in or		
		The second second	5.1	
			1 0.00	
-				
			E Component Type Handle IsManaged E Misc Accept Button AccessibilityObject	ImagedSpy Properties ComponentType Handle 1180844 IsManaged True AccessibleDefaultActionDescripti AccessibleName AccessibleRole Default AccessibleRole Default ActiveControl AlowDrop Allow Transparency False Anchor Top, Left AutoScale False

Features

The Managed Spy has the following features:

- The Tree pane lets you view the hierarchy of .NET objects available in the application
- The Properties pane lets you view the exposed properties of the highlighted .NET object
- The Learn Object option is displayed when you use the Spy during recording and lets you pick specific <u>objects to learn</u>.

Commands

In addition to viewing the object hierarchy and object properties, you can perform the following tasks:

- Parent This selects the parent object of the one displayed
- Highlight This will attempt to Flash (highlight with a red rectangle) the object selected in the Spy.
- Refresh this simply refreshes the Spy view to reflect any changes that might have occurred in the application.
- Save to File this will save the properties of the currently selected object to a text file.

2.3.1.7.5 UI Automation Spy

Purpose

The **UIAutomation Spy** is used to inspect applications that contain Microsoft UIAutomation objects (e.g. Windows Presentation Framework, Silverlight or Java's Standard Widget Toolkit running on Windows).

Start Tracking (Ctrl-G)			+ -
UIAutomation Object	Properties		
 □ [List] □ listBoxItem0[ListItem] □ listBoxItem1[ListItem] □ listBoxItem2[ListItem] □ listBoxItem3[ListItem] □ listBoxItem4[ListItem] 	Item Type Property Labeled ByProperty Localized Control Type Property Name Property Native Window Handle Property Orientation Property ProcessId Property Runtime Id Property	Not supported Not supported list 1378098 Not supported 8728 System.Int32[]	•
	Primary AutomationID IndexInParent Name NativeHandle SelectionPatternIdentifier CanSelect MultipleProperty	listBox1 18 1378098 False	E

Features

The UIAutomation Spy has the following features:

- The **Tree** pane lets you view the hierarchy of UIAutomation objects available in the application
- The Properties pane lets you view the exposed properties of the highlighted UIAutomation object
- The Learn Object option is displayed when you use the Spy during recording and lets you pick specific objects to learn.

Commands

In addition to viewing the object hierarchy and object properties, you can perform the following tasks:

- Parent This selects the parent object of the one displayed
- Highlight This will attempt to Flash (highlight with a red rectangle) the object selected in the Spy.
- **Refresh** this simply refreshes the Spy view to reflect any changes that might have occurred in the application.
- Save to File this will save the properties of the currently selected object to a text file.

2.3.1.7.6 Web Spy

Purpose

The **Web Spy** is used to inspect web applications running on any of the supported web browsers (currently Internet Explorer, Firefox and Chrome). It allows you to view the hierarchy of objects in the web browser **Document Object Model (DOM)**. In addition it makes the testing of dynamic data-driven web applications easier because it lets you test out dynamic <u>XPath</u> or <u>CSS</u> queries against the web page and verify that the objects return match your expectations.

A ≠ Spy	Rapise Spy	► - ■ ×
Get Snapshot (Ctrl-G) Browser	Locators	Test XPath Learn Test CSS Learn
Ø IE DOM Tree	DOM Element	
<pre>4 dtml xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"></pre>	tagName DIV • Attributes class page • Properties	2c-bced-c921-bbc6-e9decc135fce

The **Web Spy** dialog shows a hierarchical representation of the HTML DOM elements that make up the web application being tested as well as the properties of the currently selected object. You can select the object by clicking on the object hierarchy displayed to the left. The properties displayed are categorized into different types that are described below.

The **Web Spy** also lets you visually highlight an item in the web browser from the object hierarchy and also the reverse - selecting an object in the hierarchy by clicking on its representation in the web browser.

Spy Toolbar

The Web Spy toolbar provides the following tools:

Spy				
Track (Ctrl-T)	*	00	Test XPath Learn	
Get Snapshot	Learn	Options	Test CSS Learn	
(Ctrl-G) Browser	Object	Tools	Locators	~

- Get Snapshot (Ctrl-G): Clicking on this command will refresh the contents of the DOM Tree. This should be done whenever a change is made to the state of the web page in the web browser and you want to view how the DOM objects have been changed after the change.
- Track (Ctrl+T): This tool lets you select items in the web application as rendered by the web browser and have the corresponding object be selected in the DOM tree window. This is useful if you are not sure where an item is located in the DOM tree but you can see it in the browser.

- **Highlighting Mode:** When this is selected, whenever you select an object in DOM Tree, it will highlight the item in the rendered web page with a red square. This allows you to visually see an item in the DOM tree and how it appears to the user.
- Learn: Clicking on this tool lets you take the currently selected object and add it to the <u>Object Tree</u> for the current test. It can then be used as a scriptable object in the test script. When you click on the Learn button, you have the choice (in the dropdown list) of learning the object in terms of either its **XPath** or **CSS** properties.
- Options: Clicking on this brings up the Web Settings dialog box.
- Locators: This section is described separately below in the 'Test Results' section. These tools allow you to try out different XPATH and CSS queries to see which objects match. You can then Learn the results of these queries as new Rapise objects.

DOM Tree

The DOM tree lets you view all of the HTML elements (also known as DOM objects) that make up the web application / web page being tested. The elements are showing in a hierarchical tree representation that mirrors how they are nested on the page. Each element is displayed along with the various attributes (class, id, style, etc.) that are associated with the element:

IE IE DOM Tree [] 4 <html:lang="en" xmlns="http://www.w3.org/1999/xhtml"> > <head>...</head> 4 <head> 5 <head> 6 <head> 6 <head> 6 <head> 6 <head> 6 <head> 6 <head> 7 <head> 6 <head> 7 <head> 8 <head> 9 <head> 10 <head> 11 <head> 12 <head> 13 <head> 14 <head> 15 <head> 16 <head> 17 <head> 18 <head> 19 <head><head>

The DOM elements are shown in gray, with the attribute names being displayed in red and the attribute values in blue.

Sometimes you have more attributes displayed than can be easily red. To make viewing the DOM tree easier, you can use the <u>Web Settings</u> dialog to set a list of attributes that should be excluded from the DOM tree pane.

- When you click on an element in the DOM tree its properties are displayed in the **DOM Element** pane and it's highlighted in the web browser.
- When you right-click on an element it opens a popup menu with the following options:
 - Copy copies node text to clipboard (no attribute truncation)
 - o Highlight highlights the element in the browser
- Double clicking on an element copies it's <u>XPATH</u> and <u>CSS</u> to the ribbon

DOM Element

This pane displays the properties of the currently selected object:

OM Element	
4	
ID	ff4ae447-3791-07c5-a044-6027910dc22e
tagName	TR
Accessibility	
role	rowheader
height	24
width	641
x	694
у	177
CSS	
xpath	/html/body/form[@id='ctl01' and @name='ctl01']/table[@id='tt

The properties that are displayed are grouped into the following categories:

- Primary
 - innerHTML this contains a textual representation of all the HTML content inside this element (if any)
 - o tagName this contains the name of the HTML element in upper case (e.g. TD, TABLE, DIV)
- Attributes all attributes that are not in the Primary or Accessibility section appear here
 - $_{\odot}$ id this contains the ID of the DOM element, if specified in the page
 - \circ style this contains the inline styles defined for the element
 - class this contains the list of CSS classes applied to the element (separated by spaces if more than one)
- Accessibility this contains all of the role or aria-* attributes that are defined in the W3C ARIA accessibility standard
 - \circ role
 - o aria-*
- Properties this contains the computed positional information about the element
 - o height
 - o width
 - οХ

οу

- Selectors
 - css this is the <u>CSS</u> selector that can be used to uniquely locate this element. If you click LEARN using <u>CSS</u>, this is what will be recorded with the object.
 - xpath this is the <u>XPath</u> selector that can be used to uniquely locate this element. If you click LEARN using <u>XPath</u>, this is what will be recorded with the object.

Test Results

In addition to navigating the DOM tree and Learning specific objects, the other main capability of the DOM Spy is the ability to create queries using either <u>XPath</u> or <u>CSS</u> to see which objects match the query and then learn the specific result. For example we want to find all the table cells that have at least some style information specified.

a) Using XPath

If you enter in the XPath query to locate the table cells in the Locators box at the top.

//table//td[@style]	Test XPath Lear	n
	Test CSS Lear	n

Locators

When you click **Test XPath** it will display all of the DOM elements that match the query:

Test Results	
Locator: //table//td[@style], found: 4	

You can now refine the query to only find the items you want to test.

b) Using CSS

If you enter in the <u>CSS</u> selector to locate the table cells in the **Locators** box at the top.

	Test XPath	Learn
table td[style]	Test CSS	Learn

Locators

When you click **Test CSS** it will display all of the DOM elements that match the query:

Test Results
▲ Locator: CSS=table td[style], found: 4

You can now refine the query to only find the items you want to test.

In either case, if you can adjust the query to only match a single element, you can then click the appropriate **Learn** button next to the **Test XPath** or **Test CSS** buttons. That will learn the specified query as a new object that can be scripted against in Rapise. This is very useful if you want to dynamically select an object based on its content rather than a hard-coded ID or position.

In addition, in the test results view, when you click on a result:

- The right-click popup menu is the same as for DOM Tree if type of the result is a DOM Element
- If the result is simple text then only Copy is available, which copies the text
- Clicking on a DOM element in the results list opens it in DOM Element pane and also selects it in the DOM Tree pane

2.3.1.8 Object Manager

Purpose

The **Object Manager** allows you to merge the **object trees** of two different Rapise tests. This can be useful when you have a new test that needs some of the objects from a test that you have already written.

Dbject Manager		
Source		Destination
Demo 1 TC88\Rapise Web Demo 1 T Copy Object to the Right (F5)		on (Ctrl+D) 📑 Copy All to Destination 🝦
Source	ID	Destination

How to Open

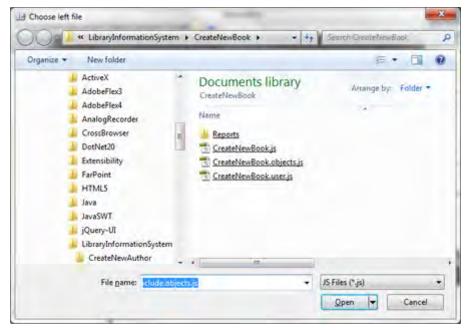
Click on the **Object Mgr** icon in the main Rapise <u>Test Ribbon</u>. This will display the object manager screen for the current test as illustrated in the screenshot above.

Choosing Files to Merge

In the example above we have opened up a test case that has some objects. Now we need to open up another test that also has some objects. To do this, click on the[...] button to the right of the **Destination** text box to open up a Rapise test object file (*.objects.js):

Source		Destination	
C:\SpiraRepository\Rapise Web Demo		•	
 Copy Object to the Right (F5) 	X Remove from Destinatio	on (Ctrl+D) 📑 Copy All to Destination 🤤	
Source	ID	Destination	

This will then bring up the File selection dialog box, where you can choose which other object file to open:



Once you have selected the file, the **Object Manager** dialog will display the list of objects to be merged (see next section).

If you click on the [...] button in left hand side of the dialog box, marked **Source**, you will be able to select a different Rapise test object file (*.objects.js) that you want to copy the objects from.

If you want to make the current test the **Destination** rather than the **Source** (i.e. you want to add objects to the current test rather than exporting from the current test), simply click the blue **Arrow** icon and the current test will be moved to the destination:

Source	🕞 C	:\SpiraRepository\F	Destination Rapise Web Demo 1 TC88\Rapise
 Copy Object to the Right 			Copy All to Destination 🭦
Source	ID		Destination
s had	= Log_In	{}	
{}	= Username_	{}	
. {}	= Password_	{}	
· {}	= ctl00	{} {}	
. {}	= Book_Management		
· {}	= Amsterdam	{}	
· {}	= Log_Out	{}	

Viewing the Objects to Merge

Once you have selected both the source and destination object files, the system will display the dialog that lets you see all the objects defined the source and destination tests. You can now choose which objects to add/delete to/from the destination test:

Source			Destination		
		C:\SpiraRepository\Conference Web Test 1 TC42\Confere			
 Copy Object to the Right 	(F5) 🗙 Remove from Destin	ation (Ctrl+D)	Copy All to Destination		
Source	ID	Destination			
p {}	= Log_In	{}			
» {}	= Username_	{}			
» {}	= Password_	{}			
» {}	= ctl00	{}			
» {}	= Book_Management	{}			
▶ {}	≠ Amsterdam				
» {}	= Log_Out	{}			
Þ	# Hound_of_the_Bas	{}			

For each object in the **object manager** you will see an [>] expand icon in the **left-hand** side. When you click on this icon it will expand the object to display its properties. If the same object is in both the source and destination, you will see the properties of both versions on the left and right hand sides respectively. If it only exists in the source or destination, then it will only show the properties on the appropriate side:

Source SpiraRepository\Rapise Web Demo	1 TC88\Rapi 🗩		Destination nference Web Test 1 TC42\Conf	
 Copy Object to the Right (F5) 	× Remove from Dest	tination (Ctrl+D) 📑 Copy All to Destination 🤿		
Source	ID		Destination	-
· {}	= Log_In	{}		
· {}	= Username_	{}		
· {}	= Password_	{}		
	= ctl00	{}		
· {}	= Book_Management	{}		
4 {}	≠ Amsterdam			
» [1]	≠ locations			
Library Information System	≠ window_text			
IEFrame	≠ window_class			
param:object_name	≠ object_text			
ROLE_SYSTEM_CLIENT	≠ object_role			
Internet Explorer_Server	≠ object_class			
0	≠ version			
UTMI Object	- chiest time.			Ľ

Each object in the source object list will be displayed with one of two icons:

- equals (=) this means that the same object exists in both the source and destination test object files.
- o not-equals (≠) this means that the object only exists in the source file and not in the destination or vice-versa

You can see which file(s) an objects is defined in (source, destination or both) by looking for the {...} icon. If you see this on the left hand side only, this object only exists in the source file, if you see it in the right-hand side, it only exists in the destination. If you see it on both sides then it exists in both the source and destination:

Source C:\SpiraRepository\Rapise Web D	emo 1 TC88\Bani	\SpiraBenository\C	Destination onference Web Test 1 TC42\Con	ferer
 Copy Object to the Right I 				
Source	ID		Destination	
p {]	= Log_In	{}		
» {}	= Username_	{}		
» {}	= Password_	{}		
» {}	= ctl00	{}		
» {}	= Book_Management	{}		
» {}	≠ Amsterdam			
» {}	= Log_Out	{}		
Þ	≠ Hound_of_the_Bas	{}		

Merging the Objects

To add an object from the **source > destination** test (for example the 'Amsterdam' object in this example). select the row in question:

Source : SpiraRepository\Rapise Web I:	Demo 1 TC88\Rapi 🔶 C	Destination SpiraRepository\Conference Web Test 1 TC42\Conf
Copy Object to the Right	(F5) X Remove from Destin	nation (Ctrl+D) 📑 Copy All to Destination 🭦
Source	ID	Destination
p {]	= Log_In	{}
» {}	= Username_	{}
» {}	= Password_	{}
» {}	= ctl00	{}
» {}	= Book_Management	{}
p {}	🗭 Amsterdam	
» {}	= Log_Out	{}
\$	≠ Hound_of_the_Bas	. {}

Then click on the **'Copy Object to the Right (F5)'** icon in the toolbar. This will copy the object from the source to the destination:

Source		Destination
		:\SpiraRepository\Conference Web Test 1 TC42\Conf nation (Ctrl+D) _ Copy All to Destination
Source		Destination
» {]	= Log_In	{}
{}	= Username_	{}
{}	= Password_	{}
{}	= ct(00	{}
	= Book_Management	{}
▶ {}	= Amsterdam	{]
s {}	= Log_Out	{}
6	≠ Hound_of_the_Bas	{}

You can tell that the object has been copied because the **not-equals** (\neq) sign changes to the **equals** option (=).

Conversely, to remove an object (e.g. 'Hound of the Baskervilles') from the destination, simply select the row:

Source)emo 1 TC88\Rani	Destination :\SpiraRepository\Conference Web Test 1 TC42\Conf
		nation (Ctrl+D) 📑 Copy All to Destination 🖕
Source	ID	Destination
» {]	= Log_In	{}
» {}	= Username_	{}
» {}	= Password_	{}
» {}	= ctl00	{}
» {}	= Book_Management	{}
p {}	= Amsterdam	{}
» {}	= Log_Out	{}
¢.	# Hound_of_the_Bas	. {}

Then click on the 'Remove from Destination (Ctrl+D)':

Source SpiraRepository\Rapise Web:		Destination SpiraRepository\Conference Web Test 1 TC42\Conf
Copy Object to the Right	t (F5) 🗙 Remove from Destin	nation (Ctrl+D) 📑 Copy All to Destination 🝦
Source	ID	Destination
p {]	= Log_In	{}
p {}	= Username_	{}
p {}	= Password_	{}
» {}	= ctl00	{}
p. {}	= Book_Management	{}
p {}	= Amsterdam	{}
» {}	= Log_Out	{]

The object will now have been removed from the **destination** object tree.

<u>Warning:</u> All of the changes you make to the objects file are committed immediately, so only delete objects in the destination test that you no longer want to be part of the test.

2.3.2 Playback

Purpose

When you record a test, Rapise translates your actions into a script. When you **playback** the test, the script is executed.

Usage

You can either run your script from the <u>Command Line</u>, or you can play it back while Rapise is open (described below):

- 1. You will first need to <u>open your test</u>. There is no need to have the AUT (Application Under Test) open. Rapise will open the AUT before it begins execution of the test.
- 2. Now, press the play button at the top of the Rapise test ribbon.



Executing

3. During test execution, Rapise displays an execution monitor dialog box that lets the user see the progress of testing playback. The dialog is only shown during test execution and can be turned off in the <u>Options</u> dialog. The following is a screenshot of the test execution monitor.

Test: Button			
	1%		0:00:01
Passed: 1	Failed: 0	Unresolved: 0	
Starting scena	ario: Test		
Pause	Stop (Ctrl+Shift+F12)		Hide

The user can pause or stop the test execution by clicking either the **Pause** or **Stop** button.

4. When Rapise is done executing the test, results will be displayed in a table. The rows with green text are steps that passed; the rows with red text are steps that failed. The following is a screenshot of test results where every step passed:

	#		Туре	Start +	Name	Status	Browser	Comment	Iteration
×	H	i la	9	1-	W.	(A)	W	(A)	+
		+	Assert	11:47:13.659	Username:.DoSetText(["librarian"])	Pass	Internet Explorer HTML	Returned Value: true	0
		F	Assert	11:47:13.862	Password:.DoSetText(["librarian"])	Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:14.096	ctl00\$MainContent\$LoginUser\$LoginButton.DoClick(Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:14.517	Book Management.DoClick([])	Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:15.063	(Create new book) .DoClick([])	Pass	Internet Explorer HTML	Returned Value: true	0
		+	Assert	11:47:15.609	Name:.DoSetText(["The Restaurant at the end of th	Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:15.812	Author:.DoSelect(["Agatha Christie"])	Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:16.015	Genre:.DoSelect(["Science Fiction"])	Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:16.186	ctl00\$MainContent\$btnSubmit.DoClick([])	Pass	Internet Explorer HTML	Returned Value: true	0
		1	Assert	11:47:16.654	Failure in Test	Fail	Internet Explorer HTML		0

See Also

- For more information about the report, see Automated Reporting...
- For information about recording a test, see <u>Recording</u>.
- For instructions on using the Command Line, look <u>HERE</u>.

2.3.2.1 Command Line

Purpose

Rapise test scripts can be run from the command line.

Usage

The form of the command is:

cscript SeSExecutor.js *path_to_sstest_file* [*evals*]

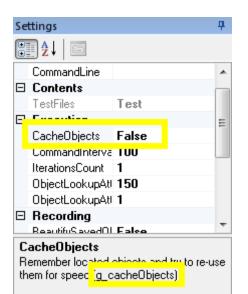
where

path_to_sstest_file is a path to sstest file, e.g. "C:\Program Files\Inflectra\Rapise\Samples \SmarteATM\SmarteATM.sstest"

evals (optional) is a statement like this:

-eval:varname1=value1;varname2=value2;...

varname is a global variable associated with an option in the <u>Settings Dialog</u>. Global variables are prefixed with a **g_.** The global variables under the **Execution** and **Recording** headings can be found by clicking on the corresponding option in the Settings Dialog (see below):



Other variables include:

- g_scriptPath
- g_reportPath
- g_objectsPath
- g_configPath

Exit Code

- 0 indicates a pass
- 1 indicates failure

See Also

Settings Dialog

2.3.2.2 Object Locator

Purpose

Object locators are created during <u>Recording/Learning</u> and used during <u>Playback</u> to identify <u>learned</u> <u>objects</u> and <u>simulated objects</u>. There are four types of locators:

- Location: This locator uses the object's index relative to encapsulating objects for identification. The location is stored as a period separated list of indexes. For instance, 1.2.3 would be "the third object in the second object in the first object." The name, class, and role of the object are also stored.
- LocationPath: This locator remembers name, class, and role property information for the object and all of its encapsulating objects.
- LocationRect: This locator stores screen coordinates.
- Ordinal: This locator creates an array of object name/object class combinations. Each object is assigned an index in the array.

Usage

The locator for each object is specified in **saved_script_objects** in **<scriptname>.objects.js** your test script. Locator information is highlighted in the simulated object example below:

```
Obj10:{"version":0,"object_type":"SeSSimulated","object_name":"regex:.* - Paint",
"object_class":"MSPaintApp","object_role":"ROLE_SYSTEM_WINDOW",
"object_text":"regex:.* - Paint",
"locations":[{"locator_name":"Location","location":{"location":"",
"window_name":"regex:.* - Paint","window_class":"MSPaintApp"}}]
```

Locator Parameters

If a piece of information in the locator matches a piece of object info (object_name, object_class, object_role, object_text) then it is stored in the locator as "param:<*object_info*>". For example:

```
"object_name": "param:object_name",
"object_class": "param:object_class",
"object_role": "param:object_role",
```

Over-riding Locator Parameters

You can over-ride the information used to locate your object at runtime. Normally, to refer to an object, you use the SeS function:

SeS('Obj9')

To override locator parameters, specify the new value in the function call. In the following example, we over-ride the **object_name** parameter for object 9:

```
SeS('Obj9', {object_name: "regex:.*"})
```

You may want to change a parameter value for every locator/object in the program. For instance, perhaps the url of the webpage has changed. Use the global variable **g_locatorparams** as in the following example:

```
function Test()
{
   // Here we use direct parameter overriding
   SeS('Obj1', {url:"http://newaddr/"}).DoAction();
   SeS('Obj2', {url:"http://newaddr/"}).DoAction();
   // And this is equivalent to above
   g_locatorparams["url"]="http://newaddr/";
   SeS("Obj1").DoAction();
   SeS("Obj2").DoAction();
   ...
}
```

See Also
Object Learning

Playback

2.3.3 Automated Reporting

Purpose

Each time you playback a test, Rapise **automatically generates a report** detailing the steps of the test, the data values used, and the outcome of each step.

Usage

Execute your test using the instructions <u>here</u>. When the test is complete, the <u>Report Tab</u> will appear in the Ribbon, and a report file (ending in .trp) will open in the <u>Content View</u>. It will look like this:

	#	Name	Start	Type	Comment	Status	Iteration
					8		-
-	3	Character read successfu	15:32:28.89	Assert	T	Pass	0
	2	Letter size is 44	15:32:28.89	Assert		Fail	0
	2	Character read successfu	15:32:28.90	Assert	e	Pass	0
	2	Letter size is 24	15:32:28.92	Assert	Text = 'e' Font = (Name='Calibri': Size=2	Pass	0
	2	Character read successfu	15:32:28.93	Assert	5	Pass	0
	2	Letter size is 12	15:32:28.93	Assert	Text = 's' Font = {Name='Calibri'; Size=1	Pass	0
	2	Character read successfu	15:32:28.95	Assert	t	Pass	0
	2	Letter size is 72	15:32:28.96	Assert	Text = 't' Font = (Name='Calibri': Size=7	Pass	0
		C:\Program Files	15:32:28.96	Test	Passed:7 Failed:1	Fail	
			0	Test Fa	ail ass:7 Fail:2 Info:0		

The first row (with a white background) is used for <u>Report Filtering</u>. The rows below that each represent a step in the test. The rows with green text represent success; the rows with red text represent failure. You can reposition the columns by dragging and dropping the column names.

The Columns

- #: For displaying icons.
- Name: The test name.
- Start: The time the test step began executing.
- Type: Can be one of the following values: Test; Assert; Message.
- Comment: Assertions and messages have associated comments. They are displayed here.
- Status: Whether the step passed, failed, or was merely informational.

Drag a column header here...

Drag a column header here to group by that column.

Use to order by the values in the chosen column. The result of dragging the **Status** column over looks like this:

Status : Fail (2 Status : Pass (7	and the second se					
	and the second se	-	<u>80</u>	四		1
#	Name	Start	/ Тур	e	Comment	Iteration
Status /						

You can expand each item to see the corresponding report rows:

		Name	Start	Type	Comment	Iteration
	2	11 I	1	B	E.	-
Sta	tus : F	ail (Z items)				
1		Name	Start	Type	Comment	Iteration
•		Letter size is 44	15:32:28.89	Assert		0
4		C:\Program Files\	15:32:28.96	Test	Passed:7 Failed:1	

Drag the Status icon back to undo the sort:

 Name	Start	Status ' ype		Comment	Iteratio
		4	E.		-

See Also

- Report Filtering
- The report output file is specified in the <u>Settings Dialog</u> (Settings > ReportPath).
- The <u>Report tab</u> of the Ribbon is used to alter the report layout.

2.3.3.1 Writing to the Report

Purpose

You can write to individual columns, create columns, and add data to the report.

Usage

Writing to and Creating a Column

Use Tester.PushReportAttribute or Tester.SetReportAttribute to set values in specific rows. Tester.PopReportAttribute reverses the effect of Tester.PushReportAttribute:

PushReportAttribute

Tester.PushReportAttribute(columnName, value);

...some test steps... //the rows corresponding to these steps will have //value in their columnName column

Tester.PushReportAttribute(columnName, value2);

...some test steps... //the rows corresponding to these steps will have //value2 in their columnName column

Tester.PopReportAttribute(columnName); //test steps proceeding this will be back to value

If columnName does not exist, it will be added to the report.

SetReportAttribute

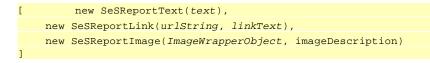
Tester.SetReportAttribute(columnName, value);

If *columnName* does not exist, it will be added to the report. Column *columnName* will be populated with *value* for rows created after this function call (unless specified otherwise).

Adding Data Data must be associated with an Assert row or a Message row.

Tester.Assert(description, expression, data, columnValuePairs) Tester.Message(description, data, columnValuePairs)

- description is a string.
- expression is the Boolean expression that the assertion tests.
- data is an array of data objects. Each data element is written to its own row below the assert/ message row with which it is associated. Data can be text, a link, or an image. The following is an array with text, link, and image data.



columnValuePairs is an object with key/value pairs. Column names are the keys. If the specified column does not exist, it will be created. Ex:
 {requirement: "Req1.2.3", paragraph: "12.5"}

See Also

- Automated Reporting
- The test samples include a sample about reporting (Reporting sstest)

2.3.3.2 Report Filtering

Purpose

Report Filtering lets you specify criteria to filter your view of the <u>test execution report</u>. Rows that do not match your criteria are hidden.

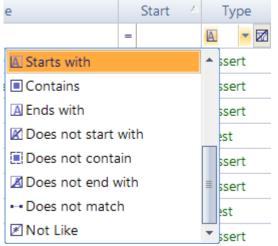
Usage

You can filter the report view while the file is open.	Directly above the first row of the report, there is a
row of filter cells. Each one has a matching criteria	button 🔟, a text-box to specify a filter value, a
drop-down menu with predefined filter values, and	

0	Name	Start	Type	Comment	Status	Iteration
	置	÷	<u>ي</u>	₩.	B	*
3	Submit Transaction.DoAction([])	14:09:32.17	Assert	Returned Value: true	Pass	0
2	Verify that: WindowText=Transaction	14:09:32.48	Assert		Pass	0
2	OK.DoClick([])	14:09:32.87	Assert	Returned Value: true	Pass	0
	Transfer	14:09:32.87	Test	Passed:7 Failed:0	Pass	0
2	Balance.DoAction([])	14:09:33.14	Assert	Returned Value: true	Pass	0
2	OK.DoClick([])	14:09:33.40	Assert	Returned Value: true	Pass	0
	Balance	14:09:33.40	Test	Passed:2 Failed:0	Pass	0
2	Application.DoMenu(["Account:Exit "])	14:09:35.06	Assert	Returned Value: true	Pass	0
	Exit	14:09:35.06	Test	Passed:1 Failed:0	Pass	0
	C:\Program Files	14:09:35.06	Test	Passed:7 Failed:0	Pass	-

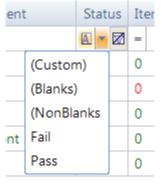
Matching Criteria

Matching criteria determine how to compare the filter string value you input with the values in the report. You can select from 16 matching criteria. Press the button marked **A** above the column you are filtering to see the possible criteria:



Predefined Filter Values

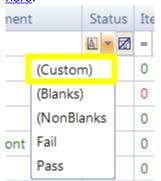
If we expand the filter cell's drop-down menu, we will see a list of predefined filtering options:



- (Custom): This option has to do with the next section Custom Filter Options.
- (Blanks): Matches all rows where the value for this column is blank.
- (NonBlanks): Matches all rows there the value for this column is not blank.
- All other predefined values are copied from cells in the column you are filtering.

Custom Filter Option

To create a filter with multiple matching criteria and filter values, select (Custom) from the filter cell's drop-down menu. The Enter filter criteria for... Dialog will open. Instructions for how to use it are here.



Undo Filtering

To undo filtering for a particular column, press the clear button for that column:

Status	
= Pass 🔻	=
Pass	0

See Also

- Automated Reporting
- Enter filter criteria for... Dialog

2.3.4 Visual Language (RVL)

Rapise includes a scriptless approach to writing automated tests as well as the option to use the built-in <u>JavaScript IDE</u>.

This section gives an overview of the Rapise Visual Language (RVL) option and why you would use it.

For more information on RVL's syntax and constructs, please refer to the separate RVL section of this user manual.

What is Scriptless Testing

Let's imagine we're recording a simple automated web test against the sample <u>http://</u><u>www.libraryinformationsystem.org</u> website that comes with Rapise:



Now in this script, we'll perform the following actions to verify that you can login correctly (the happy path):

- 1. Click on the login button
- 2. Enter your username and password
- 3. Click on the submit button
- 4. Once the home page loads, verify the name of the logged in user at the top-right

If you use the JavaScript <u>scripting option</u>, what will be recorded is a JavaScript test script that looks something like the following:

Objects	🛥 Start Page 📀 Spins Dashboard 👔 Script Testing, J 👔 Script Testing, user, js
V Id 6 60 Object Tree C1Ubersjoden sandmanjDocuments/My Inflictia j Librasj Information System - Google Chro Debrig Information System [Log In - Google Chro Global Vise Functions Viser Variables	<pre>1 2 2 2 2 2 3 2 3 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5</pre>
4 Dijects Files Settings	22 }
Properties	<pre>24 g_load_libraries=["%g_browserLibrary:Chrome HTML%"];</pre>
22 21 III	25 26 27

Now you know that you can drag and drop objects and actions from the object tree on the left-hand side into the test script, but for automation engineers that are not programmers, we found that in many cases the resulting scripts are not easy to understand and the syntax can be fiddly to get right. All you Java, JavaScript, C#, C++, C programmers out there know that you need curly braces, semi-colons at the end of each line, etc. but for others, it's not so obvious.

So to make automated testing and the power of Rapise's object based testing **easier and more accessible**, you can use the alternative RVL methodology.

How Does RVL Compare?

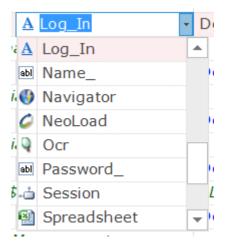
Let's imagine that we perform the exact same set of steps, recording the test script using the Rapise Visual Language approach:

Dbjects ↓ ia n m2		riptless Test +Add Now	1.rvi.xls 📓 Senp	tless Test1 is 👔 Scriptless Ter	st1.user.js				
Object Tree C:\Users\adam.sandmani,Documents\My Ra		Flow	Type	Object	Action	ParamName	ParamType	ParamValue	н
Library Information System Log In - Mozilla Firefox (iii) cti00\$MainContent\$LoginUser\$Logi [cti00\$MainCo (iii) Log_In [Log In]	1 2	Fho™ #	Learned Log	Object In					
 Password [Password:] 	3	•	Action	A Log_In	DoClick				
W Username_[Username:]	4	#	Click on Use	mame:					
Inflectra Library Information System - Mozilla Firefox	-5		Action	WUsername_	DoClick				
A Book_Management [Book Management] (iii) cti00\$MainContent\$btnSubmit [cti00\$MainContent	6	#	Set Text libr	arian in Username:					
Mome [Name]	7		Action	iel Username_	DoSetText	txt	string	librarian	
Library Information System Book Management - Moz	8	#	Set Text libr	arian in Password:					
A _Create_new_book_ [(Create new book)]	9		Action	He Password_	DoSetText	txt	string	librarian	
Global Global	10	#	Click on ct/0	0\$MainContent\$LoginUse	er\$LoginButton				
User Functions	11		Action	@ctl00\$MainConte	DoClick				
- User Variables	12	#	Click on Boo	k Management					
	13		Action	A Book_Management	DoClick				
	14	11	Click on (Cri	eate new book)					
	15		Action	A_Create_new_bo	DoClick				
	16	#	Click on Nan	ne:					
	17		Action	Mame_	DoClick				
Directs Files Settings	18	#	Set Text Add	ims Book in Name:					
roperties	19		Action	Name_	DoSetText	txt	string	Adams Book	
21	20	#	Click on ctl0	0\$MainContent\$btnSubn	nit				
2 X 1 100	21		Action	Ctl00\$MainConte	DoClick				
	22	#	Click on Log	Out					
	23		Action	∆ Log_In	DoClick				
	24								
	Las								

What you'll notice is that each of the recorded actions has now become a series of rows in the grid, with the first column being the type (perform an action, comment, make an assertion that a value matches what was expected, set a variable, be a parameter or output to the current test report):

	<u>_</u>	
	Action	 <u>A</u> Log_In
#	Action	
	Param	n –
#	Output	7
	Variable	n
#	Assert	ľ
	Condition	٦r
#	Assert	е
	Scenario	– M
		• • •

The second column is then used to select the object from the object tree:



Note that you can call custom functions (written in JavaScript), global utility objects as well as any of the recorded objects from the application being tested.

Finally, the remaining columns contain any parameter values, with subsequent rows being used if necessary:

DoSetText	txt	string	Adams Book
:4			

This means that once you have learned the objects during testing, anyone can compose the test just by adding rows to the grid and then just picking the appropriate objects and actions.

Unleash the Power of Excel

Now one the really exciting things about RVL is that it's ultimately stored as an actual Microsoft Excel file (XLSX) file inside the Rapise project. So in addition to doing simple editing within the Rapise visual editor, you also have the option to open the file in Excel natively:

6	. 5	- ¢	:				
F	ile	Home	Insert Page La	iyout Formula	s Data Re	eview View	Developer Ac
Pas	ite 💉 I	Copy 👻 Format Paint	ter В <u>I U</u>		<u>A</u> • = = =	∎ € €	Merge & Center 🝷
		oard		Font	۲ <u>م</u>	Alignment	<u>ل</u>
A1		*	X 🗸	<i>fx</i> Flow			
	А	В	С	D	E	F	G
1	Flow	Туре	Object	Action	Param Name	Param Type	Param Value
2	#	Learned L					
3			Log_In	DoClick			
4	#	Click on U					
5		Action	Username_	DoClick			
6	#	Set Text li					
7			Username_	DoSetText	txt	string	librarian
8	#	Set Text li					
9			_	DoSetText	txt	string	librarian
10	#	Click on ct					
11			ctl00\$MainCor	DoClick			
12	#	Click on B		Declist			
13 14	#	Action Click on (C	Book_Manage	DOCIICK			
14	#	-	Create_new	DoClick			
15	#	Click on N		DOCIICK			
10	π		Name_	DoClick			
18	#	Set Text A		DOCIICK			
19		Action	Name	DoSetText	txt	string	Adams Book
20	#	Click on ct	_				

This opens up many exciting possibilities of being able to use Excel formulas to auto-generate test data from another worksheet, perform validation tests by doing calculations in Excel and comparing them with the results returned from the application. You can now use Excel and Rapise to have a powerful, easy to use, yet incredibly extensive test automation framework.

2.3.5 Scripting

Purpose

There are three reasons to script with Rapise:

- 1. To modify a <u>recorded</u> test to increase coverage, add <u>assert statements</u>, or make the test <u>data-</u><u>driven</u>.
- 2. To extend recording functionality by defining your own objects, actions, and libraries.
- 3. To customize the Rapise Engine.

Usage

Rapise scripts are written in JavaScript (Microsoft JScript). You can run and debug your script using the full featured <u>Internal Debugger</u>. Rapise includes a testing API, with methods for manipulating images, spreadsheets, common GUI widgets, and more. Rapise also uses NodeJS to aid in providing hyperlinks to errors in your test.

See Also

Learn about MS JScript <u>HERE</u>.

2.3.5.1 Understanding the Script

Purpose

When you create a new test in Rapise, four files are created:

- <TestName>.sstest ? the test project file (e.g. MyTest.sstest)
- <TestName>.js ? the test script file (e.g. MyTest.js)
- <TestName>.objects.js ? the file that contains recorded objects. (e.g. MyTest.objects.js)
- <TestName>.user.js ? the file that contains user defined functions. (e.g. MyTest.objects.user.js)

where <TestName> is the name of your Test.

You can have as many javascript files in your test directory as you like, but <*TestName*>.js is the test script (unless you specify otherwise in the <u>Settings Dialog</u>). When you record, your interactions are written to <*TestName*>.js and objects are written to <*TestName*>.objects.js; when you Playback the test, <*TestName*>.js is the script that will run. All Rapise test scripts must have the same basic structure.

Usage

If you are going to modify the script, or create a test script from scratch, you will need to know the test script structure:

Basic Script

The Recording tool creates a Rapise Script with three sections:

1. <TestName>.js: A Test() function

- 2. <TestName>.js: A list of required libraries: g_load_libraries
 g_load_libraries=["Generic"]; // This script will load the Generic library
- 3. <TestName>.objects.js A list of learned objects in saved_script_objects. var saved_script_objects={

//list of objects used in this script ?
};

All Scripts must have the above three sections.

Full script

The following functions are also recognized by Rapise and may be present in the test script. Put these functions either in *<TestName>.js* or *<TestName>.user.js*.

- **TestInit()** : This function is called once before script playback. It should be used to initialize script-wide data (counters, open datasets, etc).
- TestFinish(): This function is called once after test execution. It should be used to release resources (data sets, spreadsheets). TestFinish() is a good place to post-process Reports. It may also be used as an integration point with external test management or bug tracking systems.
- **TestPrepare()** : For advanced users; TestPrepare() is called before recording and before playback. It may be used to properly initialize libraries.

See Also

To specify a different test script, see the <u>Settings Dialog</u>. The test script is specified by **Settings** > **ScriptPath**.

2.3.5.2 Naming Conventions

Purpose

The Rapise engine and API follow some simple naming conventions.

Usage

You will find descriptions of the naming conventions below. Note: italicized text represents placeholders.

- SeS<xxx> ? public functions for user
- o Do<Action> ? action implementations
- o _<somevar> and _<somename> ? private functions and objects
- o g_<varname> ? system global data.

Examples

Here are some examples to clarify the above conventions:

- o SeS("object") gets the object named "object"
- $\circ~\text{DoClick}$ public action function to click on something
- o _mydata private variable called mydata
- o g_publicdata global variable called publicdata

2.3.5.3 Defining Functions

Purpose

The Rapise test script is in Javascript. You may define as many Javascript functions as you would

like to call from your test script.

Usage

There are two ways to maintain additional functions: (1) Inside your test script and (2) in an external file.

Inside your Test Script

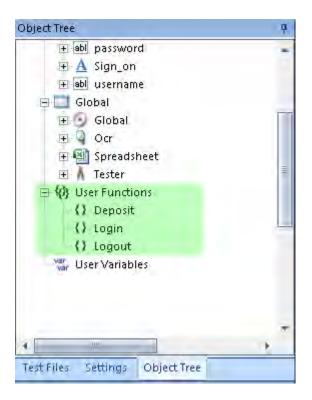
Define the function inside of one of the following functions: **Test()**, **TestInit()**, **TestFinish()**, or **TestPrepare()**. The Script Recorder will erase code placed outside of these functions.

Inside *.user.js File

It is recommended to put all user functions into <testname>.user.js file available in any test from its creation.

Test Files	Ļ
🖃 🔚 Test	
🖻 🦳 Scripts	
🌋 MyTest1.js	
🏽 MyTest1.objects.js	
MyTest1.user.js	
Test Files Settings Object Tree	

This file is automatically attached into every script. All variables and functions defined in it may be used in the test. User-defined functions are also available under the "User Functions" node in the Object Tree:



In an External File

You can define your function in another file and include it.

For example:

```
function Test()
{
    // Withdraw is defined inside the "Test" function
    function Withdraw(amount)
    {
       Log("Start Withdraw of:"+amount);
       // Withdraw logic is here
    }
    Withdraw(12.34);

    // Include "UtilityFunctions.js" to get at function Deposit()
    eval(g_helper.Include(Global.GetFullPath("UtilityFunctions.js")));
    // Deposit is defined in "UtilityFunctions.js"
    Deposit(56.78);
}
```

See Also

 To learn more about what the Script Recorder will change in your test script, see <u>Multiple</u> <u>Recordings</u>.

2.3.5.4 Global Variables

Purpose

Global variables are variables that can be accessed anywhere in the script. There are restrictions (specific to Rapise) as to where they may be placed in the test script. These restrictions do not apply to any additional script files you write and then call from your test script.

Usage

Define your global variables in **TestInit()**. Because Rapise uses javascript, you can initialize global variables inside of functions. See the sample TestInit() below.

```
function TestInit()
{
    number_of_visited_links = 0; //This variable becomes global
    var local_var = 5; //This variable is local for TestInit function
}
```

The keyword var gives variables local scope. A variable initialized without the keyword var will have global scope.

The Script Recorder knows about the following functions: Test(), TestInit(), TestPrepare(), and TestFinish(). Do not declare global variables outside of one of the preceding four functions. The Script Recorder alters the script each time it is run, and may erase your changes.

See Also

- See <u>Making Multiple Recordings</u> for details on what effect the script recorder will have on your test script.
- For details on the structure of the test script, see <u>Understanding the Script</u>.

2.3.5.5 Including other Files

Purpose

The **eval** keyword lets you use external functions and data structures in your test script; eval is a javascript reserved word.

Usage

{

}

See the example below:

function Test()

```
eval(g_helper.Include(Global.GetFullPath("myfunctions.js")));
```

See Also

• Understanding the Script

2.3.5.6 Regular Expressions

Purpose

A regular expression is a sequence of characters that describes how to construct a set of strings. It is

composed of character literals and special characters. Each character literal represents one single character (such as "a", "b", "C", "1"). The special characters can represent a character, many characters, or a choice about how to select characters.

Special Characters:

Char	Description	Examples
?	Combines with whatever character/sub- expression precedes it to represent 0 or 1 occurrences of that character/sub-expression.	a? describes the set: {"", "a"}
*	Combines with whatever character/sub- expression precedes it to represent 0 or more occurrences of that character/sub-expression.	a * describes the set: {"", "a", "aa", "aaa", "aaaa", "aaaaa", "aaaaa",}
+	Combines with whatever character/sub- expression precedes it to represent 1 or more occurrences of that character/sub-expression.	a+ describes the set: {"a", "aa", "aaa", "aaaa", "aaaaa", "aaaaa",}
	Any arbitrary character.	.* describes the set of all possible strings.
Ι	Denotes a choice between two strings	ab ba describes the set: {"ab", "ba"}
0	Denotes a sub-expression.	(abc)?d describes the set: {"abcd","d"}
0	Denotes one character chosen from all the characters with the brackets. You can use a hyphen to denote a range.	[abcde] describes the set: {"a", "b", "c", "d", "e"] [A-Z] describes the set of all one-character, alphabetic, capitalized, strings. {"A", "B", "C", , "Z"}
{n,m}	Quantifier expression. Meaning: "Between n and m occurrences of whatever sub-expression or character precedes."	(abc){1,2} describes the set: {"abc", "abcabc"}
^	The beginning of a string.	^a.* matches all strings that begin with an a.
\$	The end of a string.	.*a\$ matches all strings that end with an a.
١	Precedes a special character to take away any special meaning.	[\\\\$\-\+*] represents the set: {"\", "\$", "-", "+", "*"]

A string and regular expression match if the string is an element of the set described by the regular expression.

Usage

In Rapise, you must prepend regular expressions with the string "regex:". So the regular expression describing

all strings would be: regex: .*

There are three uses for regular expressions in Rapise: (1) in <u>Object Locators</u>, (2) in <u>action overriding code</u>, (3) in <u>Custom Libraries</u>.

2.3.5.7 Assert Statements

Purpose

An assert statement is a special Boolean condition that represents an assumption about program state at a particular point in test execution. When an assert is encountered, the condition is evaluated. A value of **False** indicates a program error. In some languages, execution will halt if an assertion evaluates to **False**. In Rapise, the result is logged to the report with failed status, and execution continues.

Create a Checkpoint

To create a <u>checkpoint</u> using an assertion, you will have to manually alter the test script (another way is to use the <u>Verify Object Properties</u> dialog during <u>Recording</u>):

- 1. Select a location in your script and a subset of application state to check.
- 2. Query for the application state. For images, use the ImageWrapper class provided with Rapise. For object properties, Get<...> methods. For example:

var xx = SeS(?OkButton?).GetX(); // X position of the object

- Save the state. If you are creating an image checkpoint, you will want to save the image to a file. If you are looking at text data, you could use a database, spreadsheet or text file. The SeSSpreadSheet class gives you access to excel spreadsheets.
- 4. **Compare**. Use the **ImageWrapper** class to compare images; use Spreadsheet to read and compare spreadsheet data.
- 5. Write an Assert Statement. Make an appropriate call to Tester.Assert method. Besides a Boolean condition, pass additional data to be placed in the <u>Report</u>.

Read about Tester. Assert syntax in the Rapise Objects documentation part.

See Also

- The test samples include a UsingImageCheckpoint.sstest
- Verifying Object Properties
- Writing to the Report

2.3.5.8 Data Driven Testing

Purpose

Data Driven Testing is an automated testing technique in which test case data is separated from test case logic. Each set of test case data consists of input values and a set of expected output values. The actual output values are compared to the expected output values to determine whether the test

passed.

{

You can perform data-driven testing either using an MS-Excel spreadsheet as the datasource or a relational database.

Using an MS-Excel Spreadsheet

The spreadsheet object is useful for implementing data-driven tests. It allows you to connect to, query, and read an excel spreadsheet from your test script. To create a data-driven test, you will:

1. **Record a test**. The exact inputs you use for the recording will not matter as much as your interactions with the objects. The following excerpt was recorded using <u>www.google.com</u>: <u>function Test()</u>

```
//Set Text Inflectra in q
SeS('Obj1').DoSetText("Inflectra");
//Click on btnG
SeS('Obj2').DoClick();
```

The actions recorded were: (1) Type Inflectra into the search box. (2) Press the Google Search button.

 Parameterize the Test() function. The Test() function has all of the procedural logic for the test. Replace input values with variables. Encapsulate the logic in a nested function with one parameter for each variable you created. As an example, we will parameterize the Test() function we created in step one:

```
function Test()
{
   function Logic(searchterm){ //our new function encapsulates the test logic
        //Set Text using searchterm
        SeS('Obj1').DoSetText(searchterm) //here we changed a hard-coded value into a
variable
        //Click on btnG
        SeS('Obj2').DoClick()
     }
     Logic("Inflectra") //don't forget to call your new function
}
```

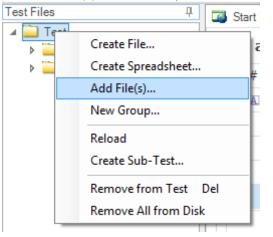
3. Create the test case data. In an excel spreadsheet, create a column for every variable in step two. Add columns for any expected output values you wish to verify. Each row is a test case.

In our google example, we only have one input value (searchterm) and we're not comparing any expected output values, so we will only need one column in our spreadsheet. Save the spreadsheet in the test folder as searchterms.xls:

	A
1	SpiraTest
2	SpiraPlan
3	SpiraTeam
4	Rapise
5	RemoteLaunch
~	

4. Add spreadsheet to the test

Use "Add File(s)..." to add a spreadsheet to the test files:



Attach Spreadsheet object to searchterms.xls Drag the 'searchterms.xls' from files tree into appropriate place in your test source:

Test Files 🕂	🖾 Start Page 🛛 🖉 Searchterms.js*
E — Test ⊕- — Scripts	8 × ···SeS('Obj1').DoSetText 9 × ···//Click on btnG
······철희 searchterms.xls	10 > SeS('Obj2').DoClick() 11 ->)
	12 13 »
	14 × 15 × Logic("Inflectra") //dc
	Output
Test Files Settings Object Tr	

6. Use Spreadsheet to access the test case data.

In our example, we use a spreadsheet object and run the test logic once for every row. function Test()

```
function Logic(searchterm){
    //Set Text searchterm in q
    Ses('Obj1').DoSetText(searchterm)
    //Click on btnG
    Ses('Obj2').DoClick()
  }

Spreadsheet.DoAttach('searchterms.xls', 'Sheet1');

// Go through all rows
while(Spreadsheet.DoSequential())
{
    // Read cell value from column 0
    var term = Spreadsheet.GetCell(0);
    // Pass it into Logic function
```

Logic(term); } }

Using a Relational Database

{

Rapise comes with the Database query global object that allows you to send SQL queries to a database and then iterate through the results. The process for creating such a data-driven test is as follows:

1. **Record a test**. The exact inputs you use for the recording will not matter as much as your interactions with the objects. The following excerpt was recorded using <u>www.google.com</u>: <u>function Test()</u>

```
//Set Text Inflectra in q
SeS('Obj1').DoSetText("Inflectra");
//Click on btnG
SeS('Obj2').DoClick();
```

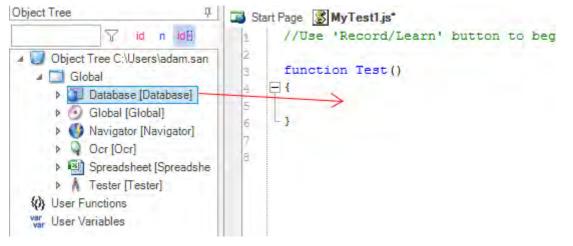
The actions recorded were: (1) Type Inflectra into the search box. (2) Press the Google Search button.

 Parameterize the Test() function. The Test() function has all of the procedural logic for the test. Replace input values with variables. Encapsulate the logic in a nested function with one parameter for each variable you created. As an example, we will parameterize the Test() function we created in step one:

```
function Test()
{
    function Logic(searchterm){ //our new function encapsulates the test logic
        //Set Text using searchterm
        SeS('Obj1').DoSetText(searchterm) //here we changed a hard-coded value into a
variable
        //Click on btnG
        SeS('Obj2').DoClick()
    }
    Logic("Inflectra") //don't forget to call your new function
}
```

3. Use Database to connect the test case data. This assumes that you already have an ODBC or OLE DB compatible relational database that contains the necessary test data.

You drag the 'Database' global object into the script editor:



and then use:

- Database.DoAttach() to make the database connection and specify the SQL query
- Database.GetRowCount() to verify that there is data
- Database.DoSequential() to loop through the dataset row by row
- Database.GetValue() to get that row's data

Here is an example of the code needed to loop through a list of records (taken from the SpiraTest database as an example) and call our **Logic()** parameterized function with the appropriate test data:

```
var success = Database.DoAttach('Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=False;Initial
Catalog=SpiraTest;Data Source=.', 'SELECT * FROM TST_PROJECT');
Tester .Assert('Successfully Connected', success);
var count = Database.GetRowCount();
Tester.Message(count);
//Loop through the rows
while(Database.DoSequential())
{
    var projectId = Database.GetValue("PROJECT_ID");
    var name = Database.GetValue("NAME");
    var description = Database.GetValue("DESCRIPTION");
    Logic(name);
}
```

2.3.5.9 Customizable Engine

Purpose

The source for most of the Rapise implementation is available for you to read and modify. You may find it useful to look at if you decide to create a <u>library</u> customized for your application.

Usage

Unless you specified otherwise, Rapise will be installed at C:\Program Files\Inflectra\Rapise\

The source code is in the Engine directory. You'll find the <u>recording/learning</u> libraries in Engine\Lib. The core logic is in four files: SeSAction.js; SeSBehavior.js; SeSCommon.js; SeSConfig.js.

If you plan to make changes to the Rapise Engine, we recommend you use a version control system capable of reconciling code conflicts, as we do not support user customizations. However, let us know if you feel that your customizations are generally useful; if we decide to integrate them into Rapise, we will support them.

See Also

- <u>Custom Libraries</u>
- <u>Scripting</u>

2.3.5.10 Scenarios

Purpose

Scenarios are a way to create reusable building blocks that can be incorporated into your test scripts. These scenarios can be either included as part of a purely automated test script, or they can be included into a predominantly <u>manual test script</u>.

Creating Scenarios

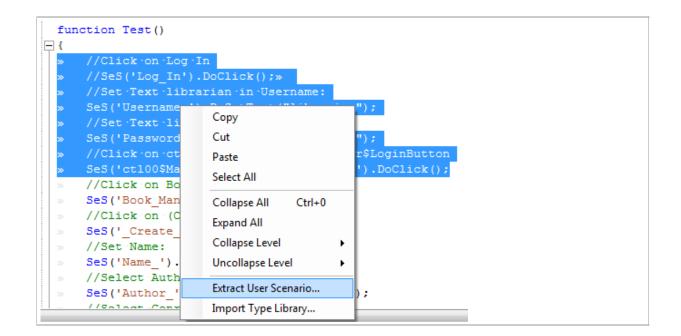
Let's say for example that you have the following Rapise test that was recorded from our sample library information web application:

```
function Test()
ł
      //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$LoginButton
      SeS('ctl00$MainContent$LoginUser$Logi').DoClick();
      //Click on Book Management
      SeS('Book_Management').DoClick();
      //Click on (Create new book)
      SeS('_Create_new_book__').DoClick();
      //Set Name:
      SeS('Name_').DoSetText(g_book_name);
      //Select Author:
      SeS('Author ').DoSelect(g book author);
      //Select Genre:
      SeS('Genre ').DoSelect(g book genre);
      //Click on ctl00$MainContent$btnSubmit
      SeS('ctl00$MainContent$btnSubmit').DoClick();
      //Click on Log Out
      SeS('Log Out').DoClick();
}
```

If we want to break up this monolithic test into individual functions (called scenarios), simply highlight the test you want to extract (for example the Login steps):

	fur	nction Test()
E	- {	
	»	//Click on Log In
	»	<pre>//SeS('Log_In').DoClick();»</pre>
	»	//Set Text librarian in Username:
	»	<pre>SeS('Username_').DoSetText("librarian");</pre>
	»	//Set Text librarian in Password:
1	»	<pre>SeS('Password_').DoSetText("librarian");</pre>
	»	//Click on ot100\$MainContent\$LoginUser\$LoginButton
1	»	<pre>SeS('ctl00\$MainContent\$LoginUser\$Logi').DoClick();</pre>
1	>	//Click on Book Management
:	\gg	<pre>SeS('Book_Management').DoClick();</pre>
1	1	

Then right-click on the section and choose the option to Extract User Scenario:



Now in the dialog box that appears, give the scenario a name (e.g. Login):

Make new Scenario	l.	
New Scenario Name		
Login		
	ОК	Cancel

This will extract the highlighted section into its own scenario:

```
function Test()
{
    //Call scenario Login
    Login();
}
/** @scenario Login*/
function Login()
{
    //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$LoginButton
    SeS('ctl00$MainContent$LoginUser$Logi').DoClick();
}
```

}

Usage in Automated Tests

When you create a new test in Rapise it will contain a MyTest.js file that contains the main test code and a MyTest.user.js file that contains any user-defined functions (called Scenarios). For example in the following test:

```
function Test()
{
    Login();
    CreateBook(g_book_name, g_book_author, g_book_genre);
    Logout();
}
```

The test function calls three scenarios that comprise the main test.

The scenarios themselves are JavaScript functions:

```
/** @scenario Login*/
function Login()
{
      //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$LoginButton
      SeS('ctl00$MainContent$LoginUser$Logi').DoClick();
}
/** @scenario Logout*/
function Logout()
ł
      //Click on Log Out
      SeS('Log Out').DoClick();
}
/** @scenario CreateBook*/
function CreateBook(name, author, genre)
      //Click on Book Management
      SeS('Book_Management').DoClick();
      //Click on (Create new book)
      SeS('_Create_new_book__').DoClick();
      //Set Name:
      Ses('Name ').DoSetText(name);
      //Select Author:
      SeS('Author_').DoSelect(author);
      //Select Genre:
      SeS('Genre ').DoSelect(genre);
      //Click on ctl00$MainContent$btnSubmit
```

```
Ses('ctl00$MainContent$btnSubmit').DoClick();
    //Verify that the Book is added to the grid
    //We need to xpath query the grid to see if any
    //added rows match the item added
    var tr = FindRowByName(name);
    Tester.Assert('Book was added successfully [TS:5]', tr.length != 0);
}
```

If you go to the **Object Tree** you will see these user functions / scenarios displayed:

```
Iser Functions
```

- {} CreateBook
- {} FindRowByName
- {} KillBrowser
- {} Login
- {} Logout

You can then drag and drop those into the test script editor to include in the main test script.

Usage in Manual Tests

When you create a new test in Rapise it will contain a MyTest.js file that contains the main test code and a MyTest.user.js file that contains any user-defined functions (called Scenarios). For example you may have the following scenario defined in the MyTest.user.js file:

You can now include that in a <u>manual test step</u>, by simply making the test step description start with an "@" symbol to denote that it is a scenario:

@Login();

Then when the manual test is executed, that one step will be passed to the scripting engine for automated execution.

Example

If you open the **CreateNewBook** sample (located in C:\Users\Public\Documents\Rapise \Samples\CreateNewBook) you will see a test that has multiple scenarios.

See Also

- Semi-Manual Testing
- Object Tree

2.3.6 Javascript IDE

Purpose

The Javascript IDE includes an editor and a debugger.

Usage

Simply <u>open a script</u> to view the editing features; create a <u>breakpoint</u> and <u>play</u> the script to view the <u>debugging features</u>.

See Also

Learn about MS Jscript <u>HERE</u>.

2.3.6.1 Internal Debugger

Purpose

The Internal Debugger provides <u>Persistent Breakpoints</u>, <u>Control Execution</u>, a <u>Watch View</u>, a <u>Variable/</u> <u>Call Stack View</u>, and <u>Tooltips</u>.

Usage

To use the internal debugger, you must first install Microsoft Script Debugger .

You can choose the Internal Debugger on the Rapise Ribbon (Test tab > Debugging menu).

No Debugging	-
Level 1 (Minimal)	-
Debugging	

The top drop-down menu has four options. Choose the Run with Internal Debugger option.

When you <u>Playback</u> your test script with a breakpoint, the debugging related menus and views will appear:

- The <u>Debugging</u> tab of the Ribbon
- The <u>Watch View</u> and <u>Variable/Call Stack View</u>

The following screenshot shows the placement of Debugging related functionality in Rapise:

R 🚽 🔍 🛞 🤫 🚽 Dequa		Rapise 2.1.3.33 (C:\Users	\adam.sandman\Documents\My Rapise Tests\Web Test 1\Web Test 1.sstest]	1
File Test Edit Depug				
Run 📮 Step Over				
🖬 Step In 🧧 Stop Debugger				
T Step Out 🔇 Break				
Debugger				
bject iree 4	Start Page 🛞 Web Test 1 js			
V id n idB				
🕑 Object Tree C:\Users\adam.san	2	S. S. MARMAN		
Inflectra Library Information		t Steps ####################################		
Library Information System	function Test()			
Global Global Global	Var success I	Database.DoAttach('Prov	rider=SQLOLEDB.1; Integrated Security=SSPI; Persist Securit	ty Info=False; Initial Cata
User Variables		Successfully Connected	', success)	
WE USE Valiables		tabase.GetRowCount();		
	a Tester.Message	(count);		
	11 //Loop through	the rows		
		.DoSequential())		
	13 8 4			
		tId = Database.GetValu		
		Database.GetValue("NI		
		ption = Database.GetVe	<pre>lue("DESCRIPTION");</pre>	
	17 1			
Test Files Settings Object Tree	19			
roperties II	*	th.		
21 m	Variables			
<u></u>	Function/Var		Value	Туре
	¥ Test	C:\Users\adam.sandm	\Documents\My Rapise Tests\Web Test 1\Web Test 1.js:€	
	count	undefined	A REAL PROPERTY OF A READ PROPERTY OF A REAL PROPER	Undefined
	description	undefined		Undefined
	nazie	undefined		Undefined
	Contractor and	undefined		Undefined
	projectId	CHOGETTHEC		Underined
	projectid success.	undefined		Undefined

In the screenshot above, you can see the <u>Debugger</u> buttons available in the ribbon at the top of the screen as well as the <u>Variables</u> and <u>Watch</u> sections in the lower pane.

See Also

• You can use the External Debugger to debug your scripts as well.



Purpose

Tooltips let you view a variable's value during debugging.

Usage

- 1. Put a breakpoint in the script at or near where you wish to investigate
- 2. Mouse over variables as you advance through the script. A small box will popup, displaying the variables' values:



See Also

- Breakpoints
- Internal Debugger

2.3.6.1.2 Control Execution

Purpose

Control Execution allows you to manually direct the execution of the script.

See Also

Usage

<u>Ribbon: Debugger</u>

2.3.6.1.3 Breakpoints

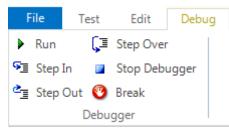
Purpose

Breakpoints stop execution of the test at a specific line in the script. They allow you to investigate program state, and trace execution flow.

Usage

To set a Breakpoint:

- 1. Open the script you would like to debug in the Source Editor.
- 2. Place the cursor at the line where you want a breakpoint.
- 3. Press F9 or the Break button on the Ribbon (Debugger tab).



4. If the Debugger tab is not visible, you can also use the **Toggle Breakpoint** option in the **Edit** tab:

File	Test Options	Edit		
Save	Paste & Cut	UndoRedo	Courier New + 10 +	Toggle Breakpoint
File	Clipboard	History	Font	Debug

- See Also
- <u>Ribbon: Debugger</u>
- Control Execution

2.3.6.2 External Debugger

Purpose

When you enable the External Debugger, the Microsoft Script Debugger is used to debug your script. Rapise provides an Internal Debugger as well.

Usage

You can enable the External Debugger on the Rapise Ribbon (Test tab > Debugging menu).



The top drop-down menu has four options:

- No Debugging
- Run with Internal Debugger: See Internal Debugger for more info.
- Run with External Debugger: Open the Microsoft Debugger to run the script.
- Run External Debugger on Error: Open the Microsoft Debugger only if an error occurs.

When you choose the **Run with External Debugger** option, Microsoft Script Debugger will open as soon as you begin <u>Playback</u> of your script. The debugger will pause on the line

WScript.Echo("SeS Player Starting...")

and display an error message. There is no actual error; you can begin debugging. Note, however, that Rapise is mostly written in javascript, and the Debugger will step through Rapise implementation as well as your test script.

영 Microsoft Script Debugger Ble Edit View Debug Window Help Fie 협 같 일 Edit & 변 은 X 월 Debug 및 또 않 않 약 다 약 중	
😤 Read only: Playerjs (break)	
SmarteUnit is a GUI unit testing engine developed by SmarteSoft, Inc. Copyright (C) 2008 - 2009 SmarteSoft Inc. The contents of this file are subject to the SmarteSoft Public License Versio 100 ("License"), You may obtain a copy of the License at http://www.smartesoft.com/SPL Any use of this file other than as authorize " under the License or copyright law is prohibited.	
Ver g_helper = new ActiveXObject(SeSHelper); g_helper.SetHost(WScript); Singleton. An instance of SeSWrappers Util which is an entry point for all with var g_util=new ActiveXObject(SeSWrappers.Util); Command Window	ndows API access-related features
Running Documents Command Window Playerjs Playerjs B MyScriptHost13 B MyScriptHost4 B ChlistriJavaScriptWSC\DrBoB\Pellegrino B MyScriptHost9 B MyScriptHost9 B MyScriptHost9 B MyScriptHost9	
Ready In 10	1.

See Also

- Internal Debugger
- For instructions on using the Microsoft Script Debugger, try this link: <u>http://msdn.microsoft.com/en-us/library/ms532989.aspx</u>

2.3.6.3 Verbosity Levels

Purpose

The Verbosity Level affects the amount of information written to the Output View.

Usage

The Verbosity Level is set on the Ribbon (Test tab > Debugging menu). See below:

No Debugging	-
Level 1 (Minimal)	-
Debugging	

See Also

<u>Ribbon: Test Tab</u>

2.3.6.4 Syntax Highlighting

Purpose

With **Syntax Highlighting**, words in a program are displayed so as to immediately indicate their function. Reserved words, variables, literals, and comments may be differentiated by color, boldness, underline etc. Syntax Highlighting makes programs easier to read and modify.

Usage

Every javascript file opened in Rapise will display with Syntax Highlighting:

```
function Test()
- {
    var success = Database.DoAttach('Provider=SQLOLEDB.1;Integrated Security=SS
    Tester.Assert('Successfully Connected', success)
    var count = Database.GetRowCount();
    Tester.Message(count);
    //Loop through the rows
    while(Database.DoSequential())
    {
   > var projectId == Database.GetValue("PROJECT ID");
   >> var name = Database.GetValue("NAME");
   » var description = Database.GetValue("DESCRIPTION");
- >>
    }
L }
```

See Also • Source Editor

2.3.6.5 Code Folding

Purpose

Code Folding allows you to hide or show blocks of code. These blocks have syntactic meaning, such as a function body, a class declaration, a loop, or a comment.

Usage

Every javascript file opened in Rapise will display with **hide** and **show** buttons to the top left of their corresponding block. In the following screenshot, hide buttons are highlighted with green boxes; show buttons are highlighted with purple boxes:

See Also <u>Source Editor</u>

2.3.6.6 Syntax Checking

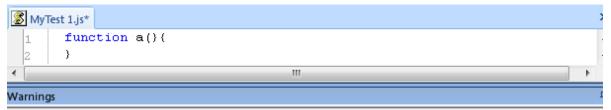
Purpose

An editor performs Syntax Checking if it notifies the user of syntax errors in their program/script.

Usage

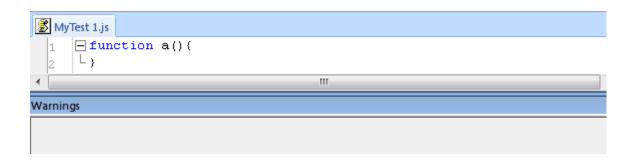
Rapise performs **Syntax Checking** as you type into the <u>Source Editor</u>. Messages regarding syntax errors can be found in the <u>Warning View</u>.

For example, you begin writing a function:



C:\Program Files\Inflectra\Rapise\Bin\Temp\MyTest1\MyTest1.js (1,1): mismatched input ')' expecting RPAREN

We have a typo here. We used ?}? instead of ?)?. Once the error is corrected, the warning view clears automatically:



See Also • Source Editor

2.3.6.7 Code Completion

Purpose

Rapise provides Code Completion for class, method and field names.

Usage

Begin typing a class, method, or field name. Press **CTRL+space** to open a list of possible completions.

Ses('Editbox').Do		
	DoAction()	DaAction()
Edit Box //This is a link SeS('Editbox').Do	 DoAnalogPlay(path, left, t DoClick() DoEnsureVisible() DoLButtonDown() DoLButtonUp() DoLButtonUp() 	User acts on "This is a link label (autLinkLabel)." in window "AUTDotNet2"
//This is a link	DoLDClick(x, y) DoMButtonDown() DoMButtonUp()	.: Change text to 'test'

Advanced

Rapise has built-in code completion logic that lets it suggest the available list of functions for a specific object. However since JavaScript is fundamentally an un-typed language, for the code completion to work, there are some tips and tricks that you can use.

One may define a variable as simple as:

var p;

In this example p is just a variable with undefined type. It may be used as number, string or object. So Rapise has no idea of how to deal with it. So if you type a dot after "p." no code-completion window appears.

Rapise scans for variable definitions when one saves the .js source file. So if anything goes wrong then first thing is to save the file.

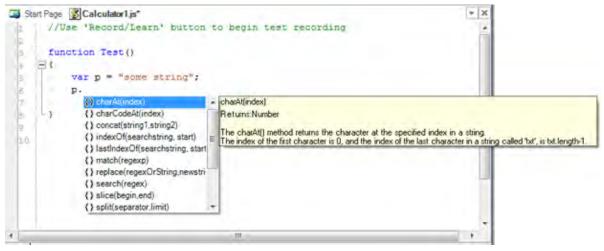
There are several ways of giving Rapise a "hint" about the variable type:

Static Assignement

First, is static assignment. Suppose you specify some constant value when defining a variable:

```
var p="some string";
```

In this case Rapise knows the type of p. So it would assist you when you type a dot "." after p:



Using Comments to Suggest the Type

In some cases variable type is not clear from its definition or assignments is not static:

```
var v1 = input;
var v2;
```

To deal with such cases the code should be instrumented. For example, if we know that input is string and v2 will be used as number then we may explain it to Rapise by placing variable type using special comment: /**<var_type*/ right together with var definition. It should be placed right either between var keyword and variable name or right after an assignment operation (=), if any. I.e.:

```
var v1 = /**string*/input;
var /**number*/v2;
```

So now Rapise will be able to display the list of available methods and properties:

v2.	
{} toExponential(x)	toExponential(x)
<pre>{ } toFixed(x)</pre>	Returns:string
<pre>{ } toPrecision(x)</pre>	The L E second (10) with the second
<pre>{ } toString(radix)</pre>	The toExponential() method converts a number into an exponential notation.
<pre>{ } valueOf()</pre>	
◇→NEGATIVE_INFINITY	
◇→POSITIVE_INFINITY]

Another common case is a function parameter. If you have function that is defined:

```
function my_func(patient_index, patient_name)
{
}
```

The type of parameters patient_index and patient_name are not known, but may be explained in a similar way:

function my_func(/**number*/patient_index, /**string*/patient_name)

So it becomes known to Rapise:

<pre>function my_func(/ { patient index.</pre>		<pre>index, /**string*/patient_name)</pre>
<pre>>> patient_index. }</pre>	() toExponential(x) {} toFixed(x) {} toPrecision(x) {} toString(radix) {} valueOf() → constructor →MAX_VALUE	toExponential(x) Returns:string The toExponential() method converts a number in
	♦→MIN_VALUE ♦>NEGATIVE_INFINITY ♦>POSITIVE_INFINITY	

Code completion for variable names is useful when you have multiple variables or function parameters and need to type them quickly. In this case Alt+Space keyword combination will bring up a list of variables and functions starting with just typed keyword.



2.3.7 Unit Testing

Purpose

Unit Testing involves testing individual units of a piece of software to make sure they act as intended. The units tested are usually functions or class methods.

Usage

There are five ways that Rapise can help you Unit Test:

- 1. Rapise methods support testing objects and methods in <u>DLLs</u>.
- 2. Rapise can test ActiveX objects and their methods through their COM Interface.
- 3. If you choose to write your Unit tests in a third-party tool, Rapise has a Command Line interface where you can access its functionality.
- 4. Test results are written to a TAP file, which allows integration with Unit Testing frameworks.
- 5. Rapise tests can be invoked from Visual Studio MS-Test and NUnit tests.

2.3.7.1 **Visual Studio**

This section describes how to execute Rapise tests from within the Visual Studio Unit MS-Test unit testing framework. It also includes information on using with Visual Studio Test Explorer and Visual Studio Team Services (formerly known as Microsoft Team Foundation Server (TFS)).

Unit Test Mapping

Rapise integrates with Visual Studio at Unit Test level.

Create a Unit Test project in Visual Studio, add a unit test and a test method. In the References section add the DLL:

c:\Program Files (x86)\Inflectra\Rapise\Extensions\UnitTesting\VSUnit\SeSVSUnit\Bin\Release\SeSVSUnit.dll

In a test method specify absolute path to a Rapise test and pass TextContext parameter to Rapise.TestExecute function:

``C# namespace UnitTestProject1 { [TestClass] public class UnitTest1 { public TestContext TestContext { get; set; }

```
[TestMethod, TestCategory("browser")]
    public void CreateNewBook()
        Rapise.TestExecute(@"c:\Demo\Framework\CreateNewBook\CreateNewBook.sstest", TestContext);
    }
1 ```
```

Parameters

}

To pass parameters to Rapise test create .runsettings file.

Each parameter with name starting with g_ will be passed to Rapise via command line.

Here is an example of selecting a browser to use for execution of cross-browser tests:

"xml <?xml version="1.0" encoding="utf-8"?>

Visual Studio Test Explorer

Once Rapise tests are mapped to unit tests one can use Visual Studio Test Explorer to run tests and analyze results.

Test Explorer	• 4 ×	
[t= → 😫 Search	ρ-	
Run All 🚦 Run 🔻 📔 Playlist : All Tests 👻		
UnitTestProject1 (5)		
😢 CreateNewAuthor	24 sec	
😢 TestToFail	2 sec	
🕖 CreateNewBook	18 sec	
Ø EditExistingAuthor	16 sec	
EditExistingBook	18 sec	

TestToFail

Source: UnitTest1.cs line 38

😢 Test Failed - TestToFail

Message: Assert.AreEqual failed. Expected:<0>. Actual:<1>. Test passed: c: \Demo\Framework\NewTest\NewTest.sstest

Elapsed time: 2 sec

Output

StackTrace:

Rapise.TestExecute(String path, TestContext UnitTest1.TestToFail()

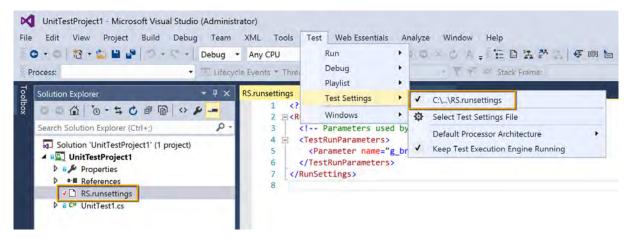


Press Output link (highlighted) to view test run results.

Test Output-TestT	oFail-4-1 + X
Test Name:	TestToFail
Test Outcome:	😣 Failed
Message:	Assert.AreEqual failed. Expected:<0>. Actual:<1>. Test passed: c:\Demo\Framework\NewTest\NewTest.sstest
Attachments	
last.tap	
last.trp	

- last.tap is a test report in Test Anything Protocol format (human readable). Click to open in any Text Viewer/Editor.
- last.trp is a test report in Rapise format. Click to open in Rapise.

One can apply .runsettings file to use for execution:

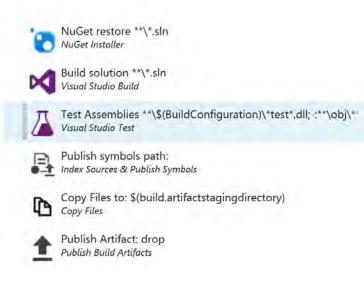


Visual Studio Team Services

In Visual Studio Team Services one can run unit tests after making a build.

М М	yFirstProject		~	Home	Code	Work	Build	& Release	Test
Builds	Releases	Library	Tas	sk Groups	Explorer				
Build	Definitions								
Mine	All Definitio	ins Qu	leued	XAML					
Request	ted by me								
E	Build and	Test in Cl requested 2		and the second second	07.2		sir.	🗸 succeed	ed
E	Build and	Test in IE equested 2					sir.	✓ succeed	ed

Build definition contains predefined steps:



Here is an example configuration of the Test Assemblies step:

Test Assemblies **\\$(BuildC	Configuration)*test*.dll;-:**\obj** 🖋 Versior	n 1.* 🗹
a Execution Options		
Test Assembly	**\\$(BuildConfiguration)*test*.dll;-:**\obj**	o
Test Filter criteria	TestCategory=browser	D.
Run Settings File	UnitTestProject1/UnitTestProject1/RS.runsettings	
Override TestRun Parameters	g_browserLibrary=Chrome HTML	Ø
Code Coverage Enabled		O
Run In Parallel		0

×

Advanced Execution Options

Reporting Options

- Test Assembly field contains a wildcard mask that selects unit tests from matching DLLs only
- In Test Filter criteria one can select tests by TestCategory which is an attribute of a Test Method: C# [TestMethod, TestCategory("browser")] public void CreateNewBook() { Rapise.TestExecute(@"c:\Demo\Framework \CreateNewBook\CreateNewBook.sstest", TestContext); }
- ** Run Settings File** is a link to <u>.runsettings</u> file.
- In Override TestRun Parameters one can override values of the parameters in .runsettings file.

Windows Agent for Test Execution

VSTS can run tests in a hosted environment, but it does not contain Rapise. So most likely you will need to run tests inside your computer network. Download and connect <u>Windows Agent</u>.

One can configure several agent pools to run tests in different environments:

1-1

MyFirstProject 🗸 Home (Code Work	Build & Re	lease	Test	۲
Overview Work Security Alerts Versio	n Control Ag	ient queues	Service	Hooks S	Service
۲ New queue + Download agent	Agents for	pool Defau	ılt		
Manage pools	Agents Role	25			
▲ All Queues	Enabled Name		State	Current St	atus
ත් ^g Default (Default)	AMST	ERDAM	Offline	Idle	
ත් Hosted (Hosted)	HASW	VELL	Offline	Idle	

References

- 1. <u>Rapise</u>
- 2. Visual Studio Test Explorer
- 3. Visual Studio Team Services

2.3.7.2 NUnit

About NUnit Integration

SeSNUnit is a sample of using NUnit. We provide special attribute to help executing Rapise GUI tests from within NUnit tests.

Standard NUnit test looks like this:

```
using System;
using NUnit.Framework;
[TestFixture]
public class MyTests
{
    [Test] - THIS IS AN ATTRIBUTE FOR STANDARD NUNIT Test
    public void MyTest1()
    {
        Assert.AreEqual(1, 2, "Check equality");
    }
}
```

Each test case is a function with a special attribute [Test]. NUnit uses it to find test cases, then collects cases in sets and so on.

Rapise integration makes execution of Rapise tests as simple as execution of normal NUnit tests:

```
using System;
using NUnit.Framework;
using SeSNUnit; - We include Rapise helper class
[TestFixture]
```

```
public class PlayerTesting
{
    - Next line means:
    "this is an NUnit test that executes Cross Browser.sstest from Rapise"
    [SeSNUnitTest(@"T:\Samples\Cross Browser\CrossBrowser.sstest")]
    public void TestIEandFirefox()
    {
        int exitCode = SeSNUnitHelper.TestExecute();
        Assert.AreEqual(0, exitCode);
    }
}
```

Now we use another attribute to mark the test:

```
[SeSNUnitTest(@"<path to .sstest>")]
```

We just mark this test as a wrapper for concrete Rapise test instance.

Installing NUnit

To run this sample you need NUnit. You can download NUnit from this site <u>http://www.nunit.org/</u> <u>index.php?p=download</u>.

Download and install NUnit package (for example, NUnit-2.4.8-net-2.0.msi).

It is installation package file, so only thing that you need to do after download is to double click on that file.

Run NUnit from the start menu:



Figure

Select "OpenProject..." from "File" menu:

NUnit		
File View Test Tools Help		
New Project Ctrl+N Open Project Ctrl+O Close	Fun Stap	
Save Ctrl+S Save Astro		
Reload Project CtXI+P Reload Tests CtrI+R		
Recent Files		
Exit		
Unloaded 1	Errors and Failures Tests Not Run Console.Dut Console.Error Trace	Log

Figure

In opened window find and select "<Rapise Folder>\Extensions\UnitTesting\NUnit\SeSNUnit \SeSSamplesNUnit\bin\Debug\SeSSamplesNUnit.dll".

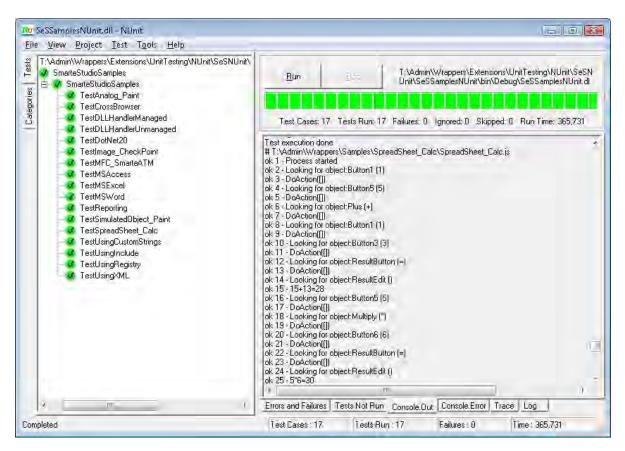
Figure

Press "Run" to start the test

no SeSSamplesNUnit.dll - NUnit Eile View Project Iest Tools Help	
Ele View Project Test Tools Help T: T: Admint/Wrappers/Extensions/UnitTesting/NUInit/SeSNUInit/ SmarteStudioSamples T: SmarteStudioSamples T: TestAnalog_Paint TestDutHandler/Managed TestDutHandler/Managed TestDutHandler/Imanaged TestDutHandler/Imanaged TestDutHandler/Imanaged TestDutHandler/Imanaged TestDotNet20 TestMSAccess TestMSXcress TestMSVord TestSimulatedObject_Paint TestSimulatedObject_Paint TestUsingCustomStrings TestUsingRute TestUsingRute TestUsingRute TestUsingRute TestUsingRute	Bun SmarteStudioSamples
ready	Errors and Failures Tests Not Run Console Out Console.Error Trace Log

Figure

After the test ended NUnit GUI must look like:



Figure

Tree Display

There is a test tree in the left part of a window, it contains tests of current NUnit project. From this tree you can easily determine the test status. Successful tests are colored green, with a check mark. If any tests had failed, they would be marked red, with an X symbol.

Progress Bar

The progress bar shows the progress of the test. It is colored according to the "worst" result obtained: red if there were any failures, yellow if some tests were ignored and green for success.

Result Tabs

The tabs in the right-hand part of the display show the results of running a test. The **Errors and Failures** tab displays the error message and stack trace for both unexpected exceptions and assertion failures. The **Tests Not Run** tab provides a list of all tests that were selected for running but were not run, together with the reason. The **Console.Out**, **Console.Error** and **Trace Output** tabs display text output from the tests.

Visual Studio Templates

Templates are available to help you creating tests in Visual Studio. To install them use run installer file: "<Rapise Folder>\Extensions\UnitTesting\NUnit\SeSNUnit.vsi" and follow the installation process

Visual Studio Conter	nt Installer Content to Install	
Which content item	(s) do you want to	install?
VB SeSNUnit Tes VB SeSNUnit Tes C# SeSNUnit Tes C# SeSNUnit Tes VB SeSNUnit Tes VB SeSNUnit Tes C# SeSNUnit Tes	t Fixture Item t Project t Fixture Item t	A project template for creating a
View files in Window	s Explorer	
Publisher information		
Publisher:	(Unknown)	
Information URL:	(Not Available)	
		Erewrowa Next.> Einrah Cancel

This will add code snippets and project templates described below.

Creating SeSNUnit test

Maybe you will want to write your own SeSNUnit test. We have special template which will help you to do that. In this part we'll explain how to use it.

The only thing you need to do is just to create "SeSNUnitTests" type project. For that open VS2005, on "Start Page" click on "Project..." (in the right side of "Create.") and in opened window from "My Templates" part select "SeSNUnitTests". If you want you can change name of dll. By default it is "SeSNUnitTests1":

New Project						2
Project types:		Templates:				
Project types:		Windows Application Class Library Windows Control Librar Crystal Reports Application ASP.NET Web Service My Templates	Class Library Windows Control Library Crystal Reports Application Device Application ASP.NET Web Service Application My Templates		t	
<no description<="" td=""><td>available></td><td></td><td></td><td></td><td></td><td></td></no>	available>					
Name:	SeSNUnitTest	sl				
Location:	D:\MyDocs\Vi	D:\MyDocs\Visual Studio 2005			ίΨ.	Browse
Solution Name:	SeSNUnitTest	sl	Create directo			
				0	ĸ	Cancel

Figure

In the created project Open "Fixture1.cs" file. All necessary references are already added:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;
using NUnit.Framework;
using SeSNUnit;
```

File also contains Fixture1class with [TestFixture]attribute:

```
[TestFixture]
public class Fixture1
{
    ...
}
```

This is the attribute that marks a class that contains tests and, optionally, setup or teardown methods.

There are a few restrictions on a class that is used as a test fixture.

• It must be a publicly exported type.

- It must not be abstract.
- It must have a default constructor
- It must have no more than one of each of the following method types: SetUp, TearDown, TestFixtureSetUp and TestFixtureTearDown.

If any of these restrictions are violated the class will be shown as a non-runnable test fixture and will turn yellow in the NUnit GUI if you attempt to run it.

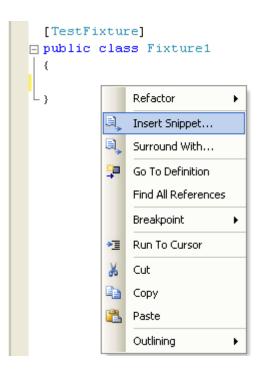
In addition it is advisable that the constructor not have any side effects since NUnit may construct the object multiple times in the course of a session.

In class we have SetUp() and TearDown() methods , and one more test method:

```
[SetUp()]
public void SetUp()
{
    //TODO - Setup your test objects here
}
[TearDown()]
public void TearDown()
{
    //TODO - Tidy up your test objects here
}
[SeSNUnitTest(/*Insert path to .sstest file which must be run.*/)]
public void TestSeS()
{
    int exitCode = SeSNUnitHelper.TestExecute();
    Assert.AreEqual(0, exitCode);
}
```

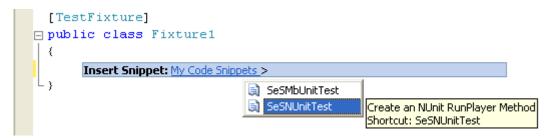
Now you also have a snippet, by which you can easily add "TestSeS" method with

[SeSNUnitTest(@"<path to .sstest>")] attribute.Right click in class body and from
opened context menu select "Insert Snippet...":



Figure

From it select "My Code Snippets" (if your snippets are in "My Code Snippets" folder, otherwise select proper folder), and then "SeSNUnitTest":



Figure

This code will be added:

```
[SeSNUnitTest(/*Insert path to .sstest file which must be run.*/)]
public void TestSeS()
{
    int exitCode = SeSNUnitHelper.TestExecute();
    Assert.AreEqual(0, exitCode);
}
```

You just need to add path of .sstest file to SeSNUnitTest attribute. If you add code via snippet in standard SeSNUnit Test project you will have two "TestSeS" methods, so don't forget to change the name of one of them.

2.3.7.3 DLL Testing

Purpose

You can create objects and invoke methods from both managed and unmanaged dlls.

Usage

Rapise provides API calls to work with managed DLLs. The Windows object **WScript** can be used with unmanaged DLLs.

Managed DLLs

- Util.InvokeMember: Invoke a class method in a managed DLL.
- Util.CreateClassInstance: Creates an instance of a class in a managed DLL.
- Util.SetFieldValue: Sets a field value in an object created with CreateClassInstance.

Unmanaged DLLs

 WScript.CreateObject("DynamicWrapper"): Create a DynamicWrapper object. The Register and ShellExecute methods of the DynamicWrapper object can be used to invoke DLL methods as in the following example:

```
var UserWrap = WScript.CreateObject("DynamicWrapper");
UserWrap.Register("shell32.dll", "ShellExecute", "I=hssssl", "f=s", "r=l");
UserWrap.Register("USER32.DLL", "MessageBoxA", "I=HsSu", "f=s", "R=l");
UserWrap.MessageBoxA( null, "" + elapsed, "Time Elapsed:", 0x30 );
```

Test Samples

There is a **Samples** folder in your Rapise directory. There are two test samples that illustrate working with DLLs:

- UsingDLLHandlerManaged
- UsingDLLHandlerUnManaged

See Also

For more information on the WScript object, see: <u>http://msdn.microsoft.com/en-us/library/at5ydy31(VS.85).aspx</u>

2.3.7.4 COM Testing Support

Purpose

Microsoft's **Component Object Model** (COM) is a standard for communication between separately engineered software components (<u>source</u>). Any object with a COM interface can be created and used remotely.

Usage

Creating a COM Object

You can create a COM object using Windows' ActiveXObject class. Once the object is created, method invocation is the same as with any other object in your program. The methods available will depend on the object's COM interface. The following example shows how to create an instance of the Word application and open a file.

var doc = new ActiveXObject("Word.Application"); doc.Documents.Open(wordFileName);

Test Samples

There are several test samples that show how to Unit Test application modules via COM interface:

- UsingMSWord
- UsingMSExcel
- UsingMSAccess

See Also

- Learn more about COM <u>HERE</u>.
- Learn more about ActiveXObject <u>HERE</u>.

2.3.7.5 Custom Strings

Purpose

Custom Strings allow you to associate meta data with your test. Each custom string has a **name** and a **value**. The value can be retrieved using the name.

Usage

Adding a Custom String

- 1. Open the NameValue Collection Editor dialog. Instructions are HERE.
- 2. Press the Add button.
- 3. Fill in a name and value for the custom string.
- 4. Press OK. The dialog will close.

```
Retrieving a Custom String value
Use the GetCustomString() method to retrieve a custom string's value. See the example below:
var factory = new ActiveXObject("Rapise.Test.Test");
var test = factory.LoadFromFile( Global.GetFullPath("UsingCustomStrings.sstest"));
var BugID = test.GetCustomString("BugID");
var TestID = test.GetCustomString("TestID");
```

See Also

- NameValue Collection Editor Dialog
- There is a <u>sample test</u> called UsingCustomStrings.

2.3.7.6 TAP Results

Purpose

Rapise supports the **Test Anything Protocol** (TAP). TAP specifies communication between unit tests and testing frameworks, such as <u>Visual Studio MS-Test</u> or <u>NUnit</u>.

Usage

The results of a Rapise test are saved to a TAP file in the same directory as the test. Tap files have a .tap extension.

TAP's general format is:

```
1..N
ok 1 Description # Directive
# Diagnostic
....
ok 47 Description
ok 48 Description
more tests....
```

For example, a test file's output might look like:

```
1..4
ok 1 - Input file opened
not ok 2 - First line of the input valid
ok 3 - Read the rest of the file
not ok 4 - Summarized correctly # TODO Not written yet
```

Example

An example Rapise .TAP file looks like the following:

```
# Simple IE Popup Example
# Starting scenario: Test
ok 1 - Open popup.DoClick([])
ok 2 - Click me.DoClick([])
ok 3 - Close me.DoClick([])
ok 4 - Simple IE Popup Example
```

See Also

- More information about tap is available at the TAP wiki: <u>www.testanything.org</u>
- <u>Visual Studio Unit Testing (MS-Test)</u>
- <u>NUnit</u>

2.3.8 Web Service Testing

What is a Web Service?

A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

Web Services allows different applications to talk to each other and share data and services among themselves. Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, Web services is used to make the application platform and technology independent.

What types of Web Service are There?

There are two broad classes of web service:

- SOAP These web services make use of the Web Service Definition Language (WDSL) and communicate using HTTP POST requests. They are essentially a serialization of RPC object calls into XML that can then be passed to the web service. The XML passed to the SOAP web services needs to match the format specified in the WSDL. SOAP web services are fully selfdescripting, so most clients do not directly work with the SOAP XML language, but instead use a client-side proxy generator that creates client object representations of the web service (e.g. Java, .NET objects). The web service consumers interact with these language-specific representations of the SOAP web service.
- 2. <u>REST</u> A RESTful web API (also called a RESTful web service) is a web API implemented using HTTP and REST principles. Unlike SOAP-based web services, there is no "official" standard for RESTful web APIs. This is because REST is an architectural style, unlike SOAP, which is a protocol. Typically REST web services expose their operations as a series of unique "resources" which correspond to a specific URL. Each of the standard HTTP methods (POST, GET, PUT and DELETE) then maps into the four basic CRUD (Create, Read, Update and Delete) operations on each resource. REST web services can use different data serialization methods (XML, JSON, RSS, etc.).

Why do we Test Web Services?

The purpose of **Web Service Testing** is to verify that all of the Application Programming Interfaces (APIs) exposed by your application operate as expected. In some ways they are similar to <u>unit tests</u> in that they test specific pieces of code rather than user interface objects.

Unlike simple unit tests however, web services tests will normally need to be developed for each of the supported versions of the API so that when a new version of a product comes out, you can regression test the latest version of the API and all previous versions. This ensures that legacy clients using the older version of the API don't need to make any changes.

Also, unlike unit tests, web services are being called across a network using the HTTP/HTTPS protocol rather than simply calling code that is resident on the same system as the test script. In that sense, they are similar to testing web sites.

Finally, in situations where you have an AJAX web application, as well as testing the front-end user interface using the appropriate UI library, you may need to test the web service that is providing the data to the user interface at the same time. In these situations you have a hybrid, web user interface and web service test.

Testing Web Services with Rapise

Rapise contains a built-in web service module that can currently test the following types of web service:

- <u>REST Web Services</u> Rapise contains a built-in <u>REST definition builder</u> and object library that allows you to prototype out your REST web service requests, inspect the returned HTTP headers and HTTP response body and then covert into a parameterized set of Rapise objects that can be scripted against in the main Rapise <u>JavaScript editor</u>. It also includes built-in support for verifying the data returned as Rapise checkpoints.
- SOAP Web Services Rapise contains a built-in SOAP request tester and object library that allows you to prototype out your SOAP web service requests, inspect the returned HTTP headers and SOAP response body and then covert into a parameterized set of Rapise objects that can be scripted against in the main Rapise <u>JavaScript editor</u>. It also includes built-in support for verifying the data returned as Rapise checkpoints.

2.3.8.1 Testing REST Web Services

What is REST and what is a RESTful web service?

REpresentational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a web API design model that offers greater simplicity over other web service protocols such as SOAP and XML-RPC.

A RESTful web API (also called a RESTful web service) is a web API implemented using HTTP and REST principles. Unlike SOAP-based web services, there is no "official" standard for RESTful web APIs. This is because REST is an architectural style, unlike SOAP, which is a protocol.

How does Rapise test REST web services?

Creating a REST web service test in Rapise consists of the following steps:

- 1. Using the <u>REST definition builder</u> to create the various REST web service requests and verify that they return the expected data in the expected format.
- 2. Parameterizing these REST web service requests into reusable templates and saving as Rapise learned objects.
- 3. Generating the test script in Javascript that uses the learned Rapise web service objects.

Rapise REST Definition Builder

When you add a web service to your Rapise test project, you get a new REST definition file (.rest) that will store all of your prototyped requests against a specific REST web service. The various REST requests are then created in the REST definition builder:

ES7		3 LibraryInfom	Library InformationSystem. e 🔄 Start Page 📓 WebServicesREST p.					
* 2 4 X .		REST Request						
Get_Session		Name	Get_Books					
		Nome						
- Get_Books		Method	GET - http	//www.libraryinformationsystem.org/services/rests	rvice svc/book?session_id+(session_id)	- Send Record		
- Insert_Boo	k	Credentials	Ibrarian 🔀					
Update_Bo		Headers	-					
- Delete_Box - UpdateIma		Pleaders	0			1		
Library Informa			Name	Value	Operations			
- Get_Sessi	09		Accept	application/json				
- Get_Books			Content-Type	application/json	8			
- Get_BookE - Insert_Boo	k	Parameters	(*)					
- Update_Bo			Name	Value	Operations			
- Updateima			(session id)	12345	Insert in URL X			
bjects Files PES operties		Body	\odot					
Misc		Response Header						
Body			Name		Value	•		
Crédential Headers	System Net NetworkCrede (Collection)	Status Code		200 OK				
Method	GET	Content-Lengt	b	3586				
Name Parameters	Get_Books (Collection)	Cache-Control		private				
Url	http://www.libraryinformatic			application/jron, charsel+uth-8				
		Dete		Tha. 22 Dec 2016 19:40:50 GMT				
		Set-Cookia						
					ASP NET_Session/a-SwkbuShomm/zS2045eytbc; path-/; Http://www.aspinal.com/aspinal-2004/			
		Server		Microsoft-IIS/8.0	Microsoft-HS(8, 0			
ome		X-AspNet-Van	noir	4 0 30319				
		Y.D.		A CU MET				
		Response Body	Ferroame Headers Outs	ut Warnings Errors Find Results				

Each REST request can then include the following items:

- Method the type of HTTP request being made (GET, POST, PUT, DELETE, etc.)
- URL the URL of the web service request with any parameter tokens included (e.g. {session_id} in our example above)
- Credentials Any HTTP Basic Authentication Headers
- Headers Any other HTTP headers (both standard and custom)
- **Parameters** Any parameters that have been defined in the URL that will be called from the Rapise test script.
- **Body** The body of the request (for POST and PUT requests). This can be in any text-serialized format such as XML or JSON.

When you execute the request, it will return back the HTTP response headers and if it recognizes the MIME content-type as either XML or JSON, it will format it to make it more readable by the tester:

Start Page	WebServicesREST is	LibraryInformationSyst	fen mei			
REST Request						
Name:	Get_Books					
Method:	GET v http://	www.libraryinformationsy	stem.org/services/rest	service_svc/book?session_id=(session_id)	* Send	Record
Credentials:	librarian 💌					
Headers.	\odot					
	Name		Value	Operations		
	Accept	application/xml				
	Content-Type	application/json		×		
Parameters:	\odot					
	Name		Value	Operations		
	(session_id)	12345		Insert in URL X		
Body	Auto Raw XML J	SON				^
isponse Body						
Auto Raw xml<br CArray CArray CArray	version="1.0" end	tp://schemas.dz	atacontract.or	rg/2004/07/Inflectra.SampleApplications.LibraryInformationSystem.Web.DataObjec	ts" xmlns	:i="http:
	Authors	and the second second				
Aesponse Body	Response Headers Output	Warnings Errors Fin	d Results			

Once you have finished with your prototyping of the web service test operations, you can then save the request definitions and use the 'Update Object Tree' option to populate the main Rapise <u>Object Tree</u>.

Web Service Object Recognition

Each of the REST web service requests that has been prototyped in the REST definition editor is converted by Rapise into a scriptable object:

Dbjects
√ id n idB
Object Tree C:\Users\Public\Documents\Rapise\Samples\WebServicesRES
LibraryInformationSystem
 LibraryInformationSystem_Delete_Book [Delete_Book]
• - LibraryInformationSystem_Get_BookById [Get_BookById]
 LibraryInformationSystem_Get_Books [Get_Books]
LibraryInformationSystem_Get_Session [Get_Session]
LibraryInformationSystem_Insert_Book [Insert_Book]
=♦ DoExecute
≡∳ DoVerify
♦→ GetRequestHeaders
٠ • • • • • • •
Objects Files REST Settings

Each of the REST service objects in the tree has operations designed to let you call the method and access the returned body either in its raw text format, or if it's a web service that returns data in JSON format, it will be able to send/receive data as native JavaScript objects.

Rapise provides you with access to the following attributes of the HTTP request before/after the request has been executed:

- Request:
 - o Method
 - \circ Url
 - o Headers (inc. authentication)
 - o Body
- Response:
 - o Headers
 - o Body

Generating Rapise REST Test Scripts

Once all the REST operations have been defined and saved as Rapise learned objects, you can call the REST operations from within your Rapise test scripts.

The easiest way to do this is to click on the **Record** button in the REST definition editor (next to the **Send** button) which will add the request to list of recorded steps:

Get_Books GET		Create Script
Insert_Book POS	T	Remove Step
		Clean
	Script Steps	

Usually you need to verify the data returned as well as call the REST method. To do this, go to the **Verify** text box underneath the Body section:

Get_Books			
GET v http://www.libraryinformationsystem.org/services/restservice.svc/book?session_id=(session_id)	v (Send Record	Verify
librarian 🔀			
\odot			
\odot			
\odot			
response[14]			
4 0			
 Author 			
Age=125			
Id=3			
Name=Arthur Conan Doyle			
AuthorId=3			
DateAdded=1/18/2015 2:24:12 PM			
DateAddedIso=1/18/2015 7:24:12 PM			
▶ Genre			
Genreld=2			

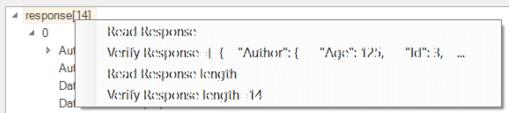
If you select the overall array **response[14]** and click the main '**Verify**' button next to the Record button, the system will automatically add a verification step that verifies all of the values. To try this, click the **Verify** button. This will add a bold verification step to the recorded script:

Get_Session GET
*Get_Books GET

You will see a script step recorded with a verification test added (it's shown in bold with an asterisk*):

However, in many cases you only want to verify certain properties. For example, we might want to just verify that 14 books are returned, and that the first book has the right name.

To do this, right-click on the **response**[14] entry to display the verification content menu:



Choose the option 'Verify Response length=14'. This adds the following step to the recorded script:



Now we want to verify the name of the first book returned. To do that, expand the "0" index entry and then right-click on the "Name" property returned:

respo	nse[14]
4 0	
Þ	Author
	AuthorId=3
	DateAdded=1/18/2015 2:24:12 PM
	DateAddedIso=1/18/2015 7:24:12 PM
Þ	Genre
	Genreld=2
	ld=1
	IsOutOfPrint=False
	Name=Hound of the Baskervilles
▶ 1	Read Response[0].Name
<u> </u>	Verify Response[0].Name Hound of the Baskervilles

Choose the option to **Verify Repsonse[0].Name = Hound of the Baskervilles**. This will add a verification step for this specific property:

✓ *Get_Books GET	-
Verify Response length=14	
Verify Response[0].Name=Houn	-

Once you are ready, click the 'Create Script' and the test script will be created for you:

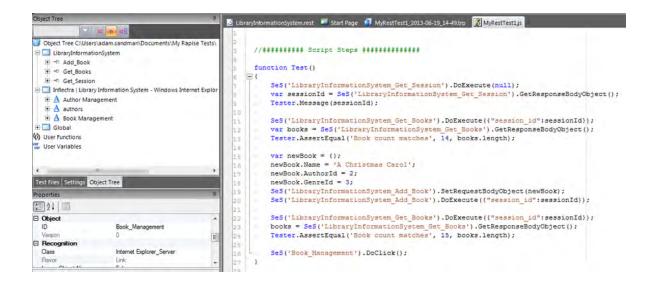
```
//First get the session id so that the site can track our changes
var LibraryInformationSystem_Get_Session = SeS('LibraryInformationSystem_Get_Session');
LibraryInformationSystem_Get_Session.DoExecute();
Tester.AssertEqual ("Error Executing Web Service', LibraryInformationSystem Get Session.GetResponseIsErrorStatus(), false)
var sessionId = LibraryInformationSystem_Get_Session.GetResponseBodyObject();
Tester.Message('Unique session ID: ' + sessionId);
// Specify 'session id' as a session-level parameter - so it will be added to every ongoing call
Session.SetParameter('session id', sessionId);
//Get the list of books
LibraryInformationSystem_Get_Books = SeS('LibraryInformationSystem_Get_Books');
LibraryInformationSystem_Get_Books.DoExecute();
LibraryInformationSystem_Get_Books.DoVerify('Tests that we have 14 books', "length", 14);
//Create a new book and get its new ID
var book = {};
book.Name = 'A Christmas Carol';
book.AuthorId = 2;
book.GenreId = 3;
var LibraryInformationSystem_Insert_Book = SeS('LibraryInformationSystem_Insert_Book');
LibraryInformationSystem_Insert_Book.SetRequestBodyObject(book);
LibraryInformationSystem_Insert_Book.DoExecute();
var bookId = LibraryInformationSystem Insert Book.GetResponseBodyObject("Id");
Tester.Message('New Book ID: ' + bookId);
```

As well as simply calling the **DoExecute()** method of each REST web service object to call the previously defined operation, you can use the various properties on the REST service object to send through specific parameter values, add/remove headers, change the authenticated user, change the request body as well as inspect all of the attributes in the request and response.

This allows you unparalleled control over the web service request, with the ability to debug and diagnose web service issues in addition to being able to quickly call the learned operations.

Since the REST objects are just like any other Rapise object, you can have hybrid test scripts that call web service methods and also test GUI objects. This is very useful when you want to test how the user

interface changes in response to specific web service API interactions, or when you have a user interface that connects to the sever using a web service (for example with a JSON-based AJAX web user interface).



Once you have created your REST web service test, you can use the standard <u>Playback</u> functionality in Rapise to execute your test and display the report:

#	Name	Start	Туре	Status	Comment	Iteration
		=				=
	Starting scenario: Test	14:49:03.725	Message	Info		
	Get_Session.DoExecute([null])	14:49:04.334	Assert	Pass	Returned Value: true	0
	c3d8dcd4-6125-427d-939a-0dd181b3cce1	14:49:04.334	Message	Info		0
	Get_Books.DoExecute([{"session_id":"c3d8dcd4-6125-4	14:49:05.051	Assert	Pass	Returned Value: true	0
	Book count matches	14:49:05.051	Assert	Pass		0
	Add_Book.DoExecute([{"session_id":"c3d8dcd4-6125-4	14:49:05.379	Assert	Pass	Returned Value: true	0
	Get_Books.DoExecute([{"session_id":"c3d8dcd4-6125-4	14:49:05.597	Assert	Pass	Returned Value: true	0
	Book count matches	14:49:05.597	Assert	Pass		0
•	MyRestTest1	14:49:05.597	Test	Pass	Passed:6 Failed:0	

2.3.8.2 Testing SOAP Web Services

What is SOAP and what is a SOAP web service?

SOAP is the Simple Object Access Protocol, and allows you to make API calls over HTTP/HTTPS using specially formatted XML. SOAP web servicesmake use of the Web Service Definition Language (WDSL) and communicate using HTTP POST requests. They are essentially a serialization of RPC object calls into XML that can then be passed to the web service. The XML passed to the SOAP web services needs to match the format specified in the WSDL.

SOAP web services are fully self-descripting, so most clients do not directly work with the SOAP XML

language, but instead use a client-side proxy generator that creates client object representations of the web service (e.g. Java, .NET objects). The web service consumers interact with these language-specific representations of the SOAP web service. However when these SOAP calls fail you need a way of testing them that includes being able to inspect the raw SOAP XML that is actually being sent.

How does Rapise test SOAP web services?

Creating a SOAP web service test in Rapise consists of the following steps:

- 1. Using the <u>SOAP test studio</u> to create the various SOAP web service test actions and verify that they return the expected data in the expected format.
- 2. Saving each of these SOAP API functions as Rapise learned objects.
- 3. Generating the test script in Javascript that uses the learned Rapise web service objects.

Rapise SOAP Testing Studio

When you add a SOAP web service to your Rapise test project, you get a new SOAP definition file (.soap) that will store all of the test invocations against a specific SOAP web service:

SOAP	Start Page - WebServicesSOAP.son	
	Invoke Request/Response	
✓ BasicHttpBinding_ISoapService	Input	Volue
Author, RetrieveByld Book, Deleto Book, Jaset Book, Inset Book, RetrieveBylde(Range Book, RetrieveByld(Book, Uploate Book, Uploate Connection_Authenticate Connection_Disconnect Connection_Disconset Connection_Disconset Connection_Disconset	 	Value True Type System Boolean
Author, RetrieveByld Book, Delinte Book, Jinset Book, Inset Book, RetrieveByDateRange Book, RetrieveByDateRange	• Output	Value Invoko
Objects Files SOAP Settings		
Properties		
24		
	Response Body	
	Auto Raw XML JSON	
	1 <7xml version="1.0" encod 2 = <s:envelope xmlns:s="http
3 = <s:Endy>
4 = <s:Fault>
5 < faultcode>s:Client</td><td>://schemas.xmlsoap.org/soap/envelope/"></s:envelope>	
	Response Body Response Headers Output Wa	amings Errors Find Results

The SOAP test studio (illustrated above) works by connecting to the WSDL location that you specify in the **Endpoint** tab of the <u>SOAP Ribbon</u>:

http://www.libraryi	Get WSDL		
Custom Endpoint	http://www.libraryinformationsystem.org/Services		
	Endpoint		

When you enter in the URL to your SOAP Web Service WSDL file and click **Get WSDL**, Rapise will download the WSDL file and display the list of available methods in the SOAP explorer:

SOAP		
▲ 🔄 WebServicesSOAP		
BasicHttpBinding_ISoapService		
=le Author_Retrieve		
=le Author_RetrieveById		
=♦ Book_Delete		
=level Book_Insert		
= Book_Retrieve		
Book_RetrieveByDateRange		
Sook_RetrieveById		
= Book_Update		
Book_UploadImage		
Connection_Authenticate		
Connection_Disconnect		
BasicHttpBinding_ISoapService1		
author_Retrieve		
Author_RetrieveById		
= Book_Delete		
=i Book_Insert		
Book_Retrieve		
Book_RetrieveByDateRange		
Book_RetrieveById		
Objects Files SOAP Settings		

Clicking on one of the available methods (e.g. Connection_Authenticate) will display that method in the main SOAP editor. Normally you will start using the **Invoke** tab of the SOAP editor:

Start Page - WebServicesSOAP.sos			- 2
Invoke Request/Response			
Input	Value		
 	Value Inhuit Type	librarian Falas System String	
Output	Value Invoke Record		
O Headers	Vetue Typě	True System Booleats	
O Body O Body Odean Connection, AuthenticateResult = Troe OB Boolean Connection, AuthenticateResultSpecifi Boolean Connection, AuthenticateResultSpecifi			

This is where you can tell Rapise to invoke the method, pass any expected parameters and view the response from the service.

If the SOAP method expects input parameters, they will be displayed in the "Body" section of the Input in a treeview. You can expand the tree and fill in the various values. In the example above, we have passed the login and password as parameters.

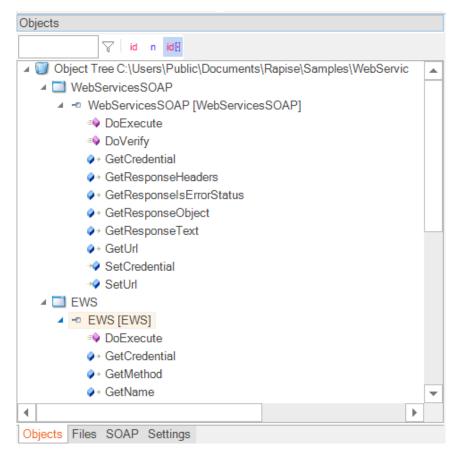
When you click the **Invoke** button, Rapise will send the SOAP request to the API and display the returned output in the Output section. In this case we get the value "True" back, indicating that our authentication request was successful.

You can also click on the **Request/Response** tab to view the raw SOAP XML that was sent to and from the server. This is very useful when debugging a service that does not work as expected.

Start Page - W	ebServicesSOAP.son			
Regum/Proor AllowAutoRedire	False	Request Auto Raw XML ISON		
AllowWriteStream BasicAuthPassw BasicAuthUserN		xml version="1.0" encoding="utf-8"? <soap:envelope http:="" www.inflectra.c<br="" xmlns:soap="http://schemas.xmlsoap.org/soa</td><td>p/envelope/" xmlns:xsi="http:.~</td></tr><tr><td>HttpProxy</td><td>text/xml; charset-utf-8</td><td></td><td></td></tr><tr><td>Method
Pipelined</td><td>POST
False</td><td></td><td></td></tr><tr><td>PreAuthenticate
SendChunked</td><td>False
False</td><td>5</td><td>Y</td></tr><tr><td>SOAPAction</td><td>">100000</soap:envelope>		Send
Uri UseCookieConta UseDefaultCrede		ResponseCode: 200 (OK) Content-Length 296 Cache-Control.private Content-Type text/xmi.charset=utf-8 Date: Thu, 22 Dec 2016 20:59:14 GMT Server: Microsoft=IIS/8.0 X-AspNet-Version.4.0.30319 X-Powered-By:ASP.NET		
		<s:envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:body></s:body></s:envelope>		

Web Service Object Recognition

Unlike the REST web service testing editor, each SOAP endpoint will generate a single Rapise SOAP object in the object tree:



These two objects (EWS and WebServicesSOAP) map to the two .SOAP files in the Rapise project. Each of these objects can be used for all of the web service requests in that file:

```
var WebServicesSOAP=SeS('WebServicesSOAP');
    "librarian", "password": "librarian"}, {} );
    WebServicesSOAP.DoExecute('Book Retrieve');
    WebServicesSOAP.DoVerify('"Body.result.length" Response',
"Body.result.length", 14);
    "Author":
{"Name": ""}, "AuthorId": 2, "AuthorIdSpecified": true, "DateAddedIso":
"2016-10-02T20:00:00", "Genre": {"Name": "" }, "GenreId": 3,
                                "IdSpecified": false,
"GenreIdSpecified": true, "Id": 0,
                    "IsOutOfPrintSpecified": false, "Name": "A
"IsOutOfPrint": false,
Christmas Carol" }} );
    Log("Resp: "+WebServicesSOAP.RequestProperties.requestPayLoad);
```

In the example above, the same object "WebServicesSOAP" is being used with DoExecute with different SOAP methods passed as the first parameter (Connection_Authenticate, Book_Retrieve, Book_Insert).

Generating Rapise SOAP Test Scripts

To save time, Rapise can generate the test script code automatically for you (instead of having to write it by hand).

To do that, after you have verified that a particular method works as expected, when you click the **Invoke** command, then click the **Record** button to record the step, and if you want to verify the data

returned, also click on the Verify button. That will generate the following:

Connection_Authenticate {"userName":"librarian","password":"librarian"}	🛅 Create Script
Book_Retrieve {}	🐰 Remove Step
Connection_Disconnect {}	🗙 Clean
Script Steps	

The steps recorded will be in **bold** type if they include a verification step, and will be in normal type if they are simply invoked with no verification.

When you click the 'Create Script' button, the following will be generated:

```
WebServicesSOAP.DoExecute('Connection_Authenticate',
{"userName":"librarian","password":"librarian"});
    Tester.Assert('Connection_Authenticate Response',
WebServicesSOAP.GetResponseObject(), {"Body":
    {"Connection_AuthenticateResult"
    true,"Connection_AuthenticateResultSpecified":true},"Headers":{}});
    WebServicesSOAP.DoExecute('Book_Retrieve', {});
    Tester.Assert('Book_Retrieve Response',
WebServicesSOAP.GetResponseObject(), { ... });
    WebServicesSOAP.DoExecute('Connection_Disconnect', {});
```

Playback of SOAP Tests

Once you have created your SOAP web service tests, you can use the standard <u>Playback</u> functionality in Rapise to execute your test and display the report:

)rag a	column hea	ader here to gr	oup by that co	lumn.				
#	Type	Start	*	Name	Status	Comment	Iteration	
* 🖪		-				(
	Assert	13:36:57.357	WebServicesS	GOAP.DoVerify(["\"Body.result.Author.Name\" R	Pass	Returned Value: true	0	
	Assert	13:36:57.368	"Body.result.C	Senre.Name" Response	Pass		0	
	Assert	13:36:57.480	WebServicesS	OAP.DoVerify(["\"Body.result.Genre.Name\" Re	Pass	Returned Value: true	0	
	Assert	13:36:57.486	"Body.result.N	Name" Response	Pass		0	
	Assert	13:36:57.589	WebServicesS	OAP.DoVerify(["\"Body.result.Name\" Respons	Pass	Returned Value: true	0	
	Assert	13:36:57.789	WebServicesS	SOAP.DoExecute(["Book_Delete",{"id":15,"IdSpe	Pass	Returned Value: true	0	
	Assert	13:36:58.009	WebServicesS	OAP.DoExecute(["Book_Retrieve"])	Pass	Returned Value: true	0	
	Assert	13:36:58.011	"Body.result.le	ength" Response	Pass		0	
	Assert	13:36:58.121	WebServicesS	GOAP.DoVerify(["\"Body.result.length\" Respons	Pass	Returned Value: true	0	
	Assert	13:36:58.840	WebServicesS	GOAP.DoExecute(["Book_UploadImage",("id":1,	Pass	Returned Value: true	0	
	Assert	13:36:58.846	"Body.Book_U	IploadImageResult" Response	Pass		0	
	Assert	13:36:58.967	WebServicesS	SOAP.DoVerify(["\"Body.Book_UploadImageRes	Pass	Returned Value: true	0	
	Assert	13:36:59.196	WebServicesS	SOAP.DoExecute(["Connection_Authenticate",{"	Pass	Returned Value: true	0	
	Assert	13:36:59.203	"Body.Connec	tion_AuthenticateResult" Response	Pass		0	
	Assert	13:36:59.309	WebServicesS	OAP.DoVerify(["\"Body.Connection_Authenticat	Pass	Returned Value: true	0	
,	Test	13:36:59,314	WebServicesS	SOAP	Pass	Passed:32 Failed:0		

2.3.9 Mobile Testing

Purpose

Rapise lets you record and play automated tests against native applications on a variety of mobile devices using either <u>Apple iOS</u> or <u>Android</u>. Rapise gives you the flexibility to test your applications on

either real or simulated devices.

Usage

Since the process for testing mobile devices depends heavily on the platform being used, we have split the guide into two separate sections:

- Mobile Testing using Apple iOS
- Mobile Testing using Android

Examples

You can find the mobile sample tests and sample Applications (called AUTAndroid for Android and AUTiOS for Apple iOS) in your Rapise installation at the following locations:

Sample Mobile Tests:

- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AppAndroid (testing a native Android App)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\WebAndroid (testing a Chrome web app)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AppiOS (testing a native iOS App)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\WebiOS (testing a Safari web app)

Sample Applications

- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTAndroid (for iOS)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTiOS (for Android)

(we supply the sample applications as both a compiled binaries and an projects with appropriate source code)

See Also

- <u>Technologies Mobile Testing</u>, for instructions on preparing your environment for mobile testing, including instructions for installing the necessary prerequisites and configuring the various third-party components that Rapise uses to connect to the device.
- Mobile Testing Tutorial for a simple introduction to mobile device testing.
- <u>Mobile Settings Dialog</u> for information on setting up the different mobile profiles for the mobile devices you will be testing
- <u>Mobile Object Spy</u> for information on how Rapise connects to the device and lets you view the objects in the application being tested

2.3.9.1 Apple iOS

Purpose

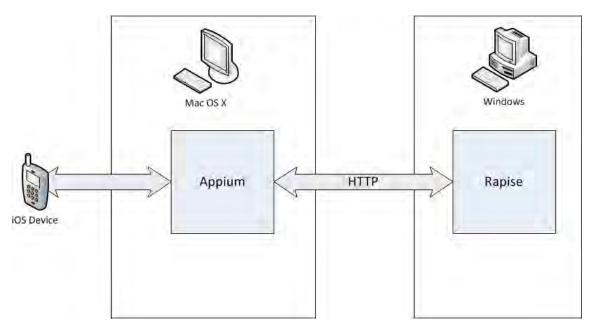
Rapise lets you record and play automated tests on real iOS devices (iPad and iPhone) as well as test applications using the iOS simulator that ships with XCode. No jailbreaking needed! With Rapise you can record on one device and playback on multiple.

Prerequisites

This section assumes that you have already installed and configured all of the necessary components.

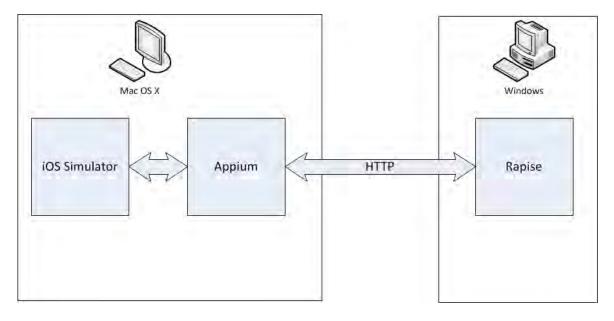
For details on this, please refer to the <u>Technologies - Mobile Testing</u> and <u>Mobile Testing - iOS Setup</u> sections that describes the necessary steps for both physical and simulated devices.

Since Rapise runs on Windows computers (PC) and iOS devices (both real and simulated) can only be tested on an Apple Macintosh (Mac) computer, it is necessary that you install **Appium** and **Apple Xcode** onto the Mac and connect to Appium over the network from Rapise running on your PC.



For Physical iOS devices the architecture looks like:

For simulated iOS devices (using the Xcode iOS Simulator) the architecture looks like:



1) Configure the Mobile Profile

To begin mobile testing, when you create the new test, make sure you choose the mobile methodology

option	"Mobile:	Mobile	Support":
--------	----------	--------	-----------

Create or Open Tes	t			
Server:	http://d	loctor/spirateam/		
Project:				
Library Information S	ystem			
Test Cases:			New Folder	New Test Case
Ability to e Ability to c	create new book edit existing book create new author Case (Ability to report of			E
Test that 1	the system lets you rep dit existing author	ort		
Test that Test Steps		Expected Result	Actual F	Result
Test that i Ability to e Test Steps Step D	edit existing author		Actual F	Result
Test that i Ability to e Test Steps Step D Mobile: Mobile Suppo	edit existing author Description	Expected Result	Actual F	Result
Test that i Ability to e Test Steps Step D	edit evisting author Description ort	Expected Result	Actual F	Result

Once you have entered the name for the new test (with the mobile methodology selected) you will be asked to choose the mobile profile. Rapise ships with several default profiles, for now select the one that is closed to the device you want to test (you can always change it later):

Specify Mobile Profile:	iPhone4	

When you click the [OK] button, Rapise will create a new mobile test with this profile selected.

Now you need to modify the profile so that it correctly matches the type of device you are testing and also so that it correctly points to the <u>Appium</u> server that you are using to host the mobile devices. Click on Options > Tools > Mobile Settings to bring up the <u>Mobile Settings</u> dialog box:

Select Profile: iPhone4	✓ Save Test Uni Duplicate Create Rename Delete
4 Connection	
Uri	http://adams-imac.corp.inflectra.com:4723/wd/hub
Details	
 Primary Capabilities 	(5)
app	/Users/elise.brooks/Library/Developer/Xcode/DerivedData/AUTiOS-gv
deviceName	Goshen iphone iOS
platformName platformVersion	71.1
udid	54cfb1c05a4438eb95e322a7ed8df3e45978654
_Primary Capabilities Desired capabilities are a set of keys and values we are interested in starting up. There are also v	(i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session arious capabilities which can modify the behavior of the server during automation.
	Close

The example screeenshot above is for an iPhone 4 physical device running iOS 7.1.1. For any iOS device (real or simulated) you will need to provide the following:

- Uri this is the URL to your Appium server. We shall discuss this shortly
- **app** this needs to the path (on the Mac running Appium) to the Application being tested on the device (e.g. /Users/user.name/Library/Developer/Xcode/DerivedData/AUTiOS-gvdyymxgyzrfgqdfvfylapawjoyd/Build/Products/Debug-iphonesimulator/AUTiOS.app)
- **deviceName** this needs to match the name of the device being connected
- platformName this needs to be set to 'iOS'
- platformVersion this needs to be set to the correct version of iOS that the device is running

In addition, for physical devices only, you need to specify:

• **udid** - The unique device identifier of the connected physical device (leave blank for simulated devices)

Once you have entered in the information and saved the profile, make sure that Appium is running on the Mac (see the <u>Technologies - Mobile Testing topic</u> for details) and then click the **[Test URL]** button to verify the connection with Appium:



Now you can start testing your mobile iOS application.

2) Using the Mobile Spy

The Mobile Spy will let you view an application running on the mobile device, take a snapshot of its screen and then interactively inspect the objects in the application being tested. This is a useful first step to make sure that Rapise recognizes the application and has access to the objects in the user interface.

To start the Mobile Spy, open the Spy icon on the main <u>Test ribbon</u> and select the Mobile option and the Mobile Spy will be displayed in **Discovery Mode**. Now click the **[Get Snapshot]** button to display the application specified in the <u>mobile profile</u> on the screen:

Get Snapshot	Page Source Go To Url Control Test Locator	Select Profi	Lange and the second		Discovery Mode Select Context		
(Ctrl-G) Connection	Device		Profile		Context		
 (0) UIAApplication[0.20][320.480] (0) UIAV/indow[0.0][320.480] (1) UIAW/indow[0.0][320.480] 		Aut Id Ind Tay Tex Titl		AUTIOS Q UIAApplic AUTIOS (0) UIAAp	t.localTarget()frontMostApp sation uplication[0,20][320,480] slication[1]	••••• Verizon 🗟 Master - Deta Table view with ite	3:04 PM AUTIOS ail Table View
		Wie X Y	ight äth operties t el me	460 320 0 20 AUTIOS AUTIOS		Advanced Co	Label, Picker, Switch, Slider, Imag

In the example above, we are displaying the sample iOS application that comes with Rapise (AUTiOS).

If you click on one objects in the user interface, it will be highlighted in Red and the tree hierarchy on the left will expand to show the properties of that object:

			ect Profile iPhone4		Discovery Mode		
			dit Profiles Select Context		-		
Connection Device Device (0) UIAApplication[0.20][320.480] (0) UIAAvidav0[0.0][320.480] (0) UIAAvidav10.0][320.480] (1) UIATable/Cell[0.10320.180] (0) UIATable/Cell[0.108][320.152] (0) UIATable/Cell[0.152][320.196] (2) UIATable/Cell[0.152][320.196] (2) UIATable/Cell[0.480][320.524]			Profile Cont Calculated Automation UIATarget.localTarget() frontMon Id Basic Controls, Button, TextField IndexinParent 0 TagName UIAStaticText Text Basic Controls, Button, TextField Title (0) UIAStaticText(1) (108)[20,15 XPath //UIAStaticText(1) (108)[20,15]		ntrols, Button, TextField, Lab	●●●○○ Verizon 훅	AUTIOS
					aticText[0,108][320,152]	Master - De Table view with i	etail Table View
			Layout Height Width X	44 320 0		Basic Contr Button, TextField	rols J. Label, Picker, Switch, Slider, Image
						Advanced Controls WebView, ScrollView, TextView, Search, TabBar, Page	
			label name value		ntrols, Button, TextField, Lab ntrols, Button, TextField, Lab		

If you want to view the contents of the Spy as a text file, just click the **'Page Source'** button and you will see the contents of the Spy properties window as a text file. If you want to perform an action on the application (e.g. click on the selected item), switch the ribbon to the 'Events' view:

1	Touch		Text	Misc
Shake		Execute Script		Change Orientation
Swipe 🝷	Scroll To	Send Keys		Dismiss Alert
Тар	Precise Tap 🔻	Text		Accept Alert
Spy	Events			

For a full list of the available events and what they mean, please refer to the main Mobile Spy Topic.

Assuming that you can see your application in the Spy and that the objects can be inspected (similar to that shown above) you can now begin the process of testing your mobile application.

Click on **Disconnect** to end your Spy session and close the Rapise Spy dialog. You will now will be returned back to your test script.

3) Recording and Playing a Test

With the new Rapise mobile test script open, click on the **Record/Learn** button in Rapise and that will display the <u>recording activity dialog</u>:

Object	Action	Data	Comment		
Verify (Ctrl+1)	Learn (C	Ctrl+2) 😳 S	Spy(Ctrl+5)	Pick Object	Resume

Now click on the [Pick Object] button and the Rapise Spy will be displayed in Recording Mode:

Disconnect	Page Source	Select Profile	iPhone4 +	Recording N	lode				
Get Snapshot (Ctrl-G)	Learn Go To Url Object Strest Locator	Edit Profiles		Select Context					
Connection	Device		Profile		Context				
(0) UIAApplication[0.20][320.480]		Calculated Automation Id IndexInParent TagName	UIATarget.localTarg AUTIOS 0 UIAApplication	jet().frontMostApp	•••∘∘ Verizon 🤶	12:58 PM			
		Text Title XPath	AUTIOS (0) UIAApplication(0 //UIAApplication[1]		Master - Detail Table View				
			Width X		460 320 0	320 0		Basic Controls Button, TextField, Label, Picker, Switch, Slider, Image	
	Y Prop hint		20		Advanced Co WebView, ScrollVie	ontrols w, TextView, Search, TabBar, Pagel			
		label name value	AUTIOS AUTIOS						

We now want to record a click on one of the menu options, simply highlight one of the menu entries:

Get Snapshot (Ctrl-G)	Learn Object States	Select Profile Edit Profiles	iPhone4 •	Recording M Select Context	Context		
Connection Device • (0) UIAApplication[0.20](320.480] • (0) UIAVindow(0.0][320.480] • (0) UIATableView[0.0][320.480] • (1) UIATableView[0.0][320.480] • (0) UIATableView[0.0][320.480] • (0) UIATableView[0.0][320.182] • (0) UIATableView[0.0][320.182] • (0) UIATableView[0.0][320.182] • (0) UIATableView[0.0][320.152] • (2) UIATableView[0.0][320.152] • (Calculated Automation Id IndexInParent TagName Text Title XPath	UIATarget local Targ Basic Controls, Butt 0 UIAStatic Text Basic Controls, Butt (0) UIAStatic Text (0, //UIAApplication[1]	get() frontMostApp on, TextField, Lab on, TextField, Lab 108][320,152]	•••••• Verizon ຈ Master - Deta Table view with iten		*
 	24]	▲ Layout Height Width	44 320 0		Basic Control Button, TextField, L	S abel, Picker, Switch, Slider,	Image
		Y Properties hint label name value	108 Basic Controls, Butt Basic Controls, Butt	and the second	Advanced Co WebView, ScrollVie	ntrols w, TextView, Search, TabBa	r, Pagel

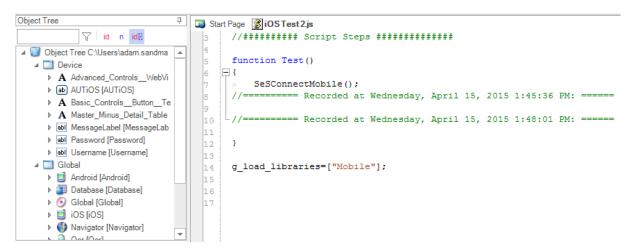
Now click the **[Learn Object]** button and the object will be added to the Rapise <u>object tree</u>. Now **on the device itself** click on the menu entry to go to the next screen, then in Rapise click **Get Snapshot** to get the updated screen:

Spy							
Get Snapshot (Ctrl-G)	Learn Qbject Test Locator	Select Profile iPh		Recording N Select Context	•		
Connection ☐ (0) UIAApplication(0,20)[320,480] ⊕ (0) UIAWindow(0,0)[320,480] ⊕ (1) UIAWindow(0,0)[320,480]	Device	Calculated Automation Id IndexInParent TagName Text Tide XPath Layout Height Width X Y Properties hint label name value	ofile. UIATarget.localTar AUTIOS 0 UIAApplication AUTIOS (0) UIAApplication[1 460 320 0 20 AUTIOS AUTIOS	get().frontMostApp 0,20][320,480]	Context Contex	1:40 PM Basic	Send
						lercury Venus Earth	-

Now click on some of the objects and choose **Learn** to add them to the <u>object tree</u>. Once you are finished, click on the **Disconnect** button. You will see the events in the recording activity dialog:

#	Object	Action	Data	Comment
11	Basic Contro	Leam		Learned Basic Controls, Button, TextField, Label, Picker, S
12	MessageLabel	Leam		Learned MessageLabel
213	Usemame	Leam		Learned Username
14	Password	Leam		Learned Password
15	AUTIOS	Leam		Leamed AUTIOS
		Learn (C		Pick Object
	y (Car(+1)	Lisen (Ch		
	y (Cat+1)	Pastu (S	Sectors	← Cancel Finish (Ctrl+3)

Now click on the Finish button and you will be taken back to the test script with the iOS objects listed:



Now that we have the objects, we can drag them into the test script editor and write the following script:

This will click on the first menu entry, then enter a username and password and then finally return back

to the main menu.

Now to playback the test simply click **Play** in the Rapise test ribbon and the test will play back in the mobile device:

aya	corumninea	der nere to grou	p by that column.			
#	Туре	Start	Name	Status	Comment	Iteration
* 🖾		-	A	A	A	-
	Message	13:54:06.008	Starting scenario: Test	Info		
	Assert	13:54:51.382	Basic Controls, Button, TextField, Label, Picker, Switch, Slider,	Pass	Returned Value: true	0
	Assert	13:55:06.386	Username.DoSetText(["test user"])	Pass	Returned Value: false	0
	Assert	13:55:21.415	Password.DoSetText(["test pwd"])	Pass	Returned Value: false	0
	Assert	13:55:36.582	AUTIOS.DoClick([])	Pass	Returned Value: true	0
•	Test	13:55:36.587	iOS Test 2	Pass	Passed:4 Failed:0	
			a Test Pass			

This is the report of the test being executed.

Example

You can find the iOS sample tests and sample Application (called AUTIOS) in your Rapise installation at the following locations:

Sample iOS Tests:

- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AppiOS (testing a native App)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\WebiOS (testing a web app)

Sample Application (AUTiOS)

C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTiOS

(we supply the sample application as both a compiled binary and an Xcode project with Objective C source code)

See Also

- <u>Technologies Mobile Testing</u>, for instructions on preparing your environment for mobile testing, including instructions for installing the necessary prerequisites and configuring the various third-party components that Rapise uses to connect to the device.
- <u>Mobile Testing iOS Setup</u> the steps for setting up XCode for developing and deploying iOS applications

2.3.9.2 Android

Purpose

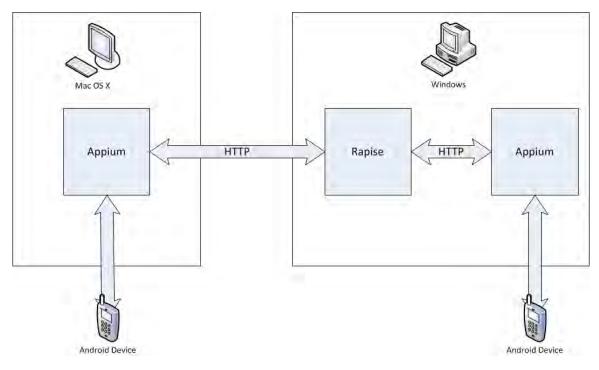
Rapise lets you record and play automated tests on real Android devices (e.g. Nexus, Galaxy) as well as test applications using the Android simulator. With Rapise you have the powerful interactive Object Spy that makes it easy to record on one device and playback on multiple.

Prerequisites

This section assumes that you have already installed and configured all of the necessary components. For details on this, please refer to the <u>Technologies - Mobile Testing</u> section that describes the necessary steps for both physical and simulated devices.

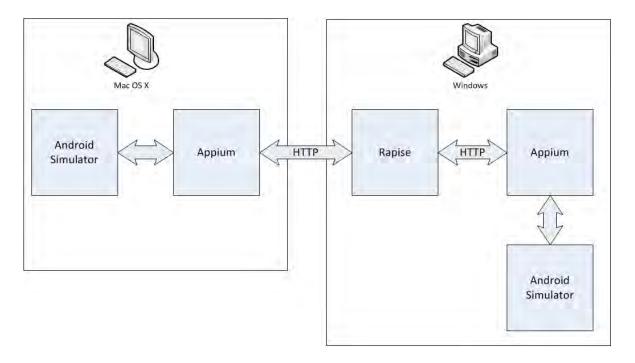
Rapise runs on Windows computers (PC) and Android devices (both real and simulated) can be tested on either an Apple Macintosh (Mac) computer or a PC:

- If using a Mac, it is necessary that you install Appium and Android Studio onto the Mac and connect to Appium over the network from Rapise running on your PC.
- If using a PC, you can either install Appium and Android Studio onto a separate PC or you can simply use the same PC that is running Rapise. The only difference will be whether the URL used to connect to Appium is a localhost URL or one pointing to the other PC.



For Physical Android devices the architecture looks like:

For simulated Android devices (using the Android Virtual Device Manager) the architecture looks like:



1) Configure the Mobile Profile

To begin mobile testing, when you <u>create the new test</u>, make sure you choose the mobile methodology option "Mobile: Mobile Support":

Create or Open 1	Test			
Server:	http://d	doctor/spirateam/		
Project:				
Library Information	n System			
Test Cases:			New Folder	New Test Case
Ability Ability Ability New T Test th	to create new book to edit existing book to create new author fest Case (Ability to report nat the system lets you rep to edit existing author			E
Test Steps				
Test Steps Step	Description	Expected Result	Actual F	Result
Step		Expected Result Specify methodology:	Actual F	Result
Step Mobile: Mobile Sup	pport.		Actual F	Result
Step	pport pport		Actual F	Result

Once you have entered the name for the new test (with the mobile methodology selected) you will be asked to choose the mobile profile. Rapise ships with several default profiles, for now select the one that is closed to the device you want to test (you can always change it later):

obile Test Setup	and the second second second	
Specify Mobile Profile:	android-nexus7	•
		ОК

When you click the **[OK]** button, Rapise will create a new mobile test with this profile selected.

Now you need to modify the profile so that it correctly matches the type of device you are testing and also so that it correctly points to the <u>Appium</u> server that you are using to host the mobile devices. Click on Options > Tools > Mobile Settings to bring up the <u>Mobile Settings</u> dialog box:

n nabilities	http://127.0.0.1:4723/wd/hub	[
abilities		1
abilities		
	(6)	
	C:\Users\Public\Documents\Rapise\Samples\L	JsingMobile\AUT\AUT
Name		
ame	Nexus 7	
Name	Android	
Version	4.4.2	
	015d24bc5d4ffe05	
abilities	(9)	
nch		
onName		
bview		
n	abilities ch in Name	015d24bc5d4ffe05 abilities (9) ch nName

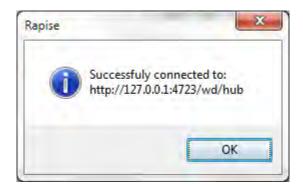
The example screeenshot above is for an Android Nexus7 physical device running Android 4.4.2. For any Android device (real or simulated) you will need to provide the following:

- Uri this is the URL to your Appium server. We shall discuss this shortly
- app this needs to the path (on the Mac/PC running Appium) to the Application being tested on the device (e.g. C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT \AUTAndroid\bin\AUTAndroid.apk). If running on the same PC as Rapise, then this path should be already correct.
- deviceName this needs to match the name of the device being connected
- platformName this needs to be set to 'Android'
- platformVersion this needs to be set to the correct version of Android that the device is running

In addition, for physical devices only, you need to specify:

• **udid** - The unique device identifier of the connected physical device (leave blank for simulated devices)

Once you have entered in the information and saved the profile, make sure that Appium is running on the Mac/PC (see the <u>Technologies - Mobile Testing topic</u> for details) and then click the **[Test URL]** button to verify the connection with Appium:



Now you can start testing your mobile Android application.

2) Using the Mobile Spy

The Mobile Spy will let you view an application running on the mobile device, take a snapshot of its screen and then interactively inspect the objects in the application being tested. This is a useful first step to make sure that Rapise recognizes the application and has access to the objects in the user interface.

To start the Mobile Spy, open the Spy icon on the main <u>Test ribbon</u> and select the Mobile option and the Mobile Spy will be displayed in **Discovery Mode**. Now click the **[Get Snapshot]** button to display the application specified in the <u>mobile profile</u> on the screen:

Disconnect			rofile android-nexus	7 🚽	Discovery Mode		
Get Snapshot (Ctrl-G) Connection	 Go To Url Test Locator Device 	Edit Pro	Profile		Select Context NATIVE	APP +	
(0) android.widget.FrameLayout[0	.0][800,1205]		Calculated			0 û	💎 🖬 2:44
 (0) android.view.View[0,0][800 (0) android.widget.LinearL 			Automation Id			AUTAndroid	
 (0) android.widget.Fra (0) android.view.V 	meLayout[0,33][800,108] iew[0,33][800,108]		IndexInParent TagName	0 android.w	idget.FrameLayout	Login	
	lget.LinearLayout[11,33][2 lget.LinearLayout[715,33]]		Text Title	(D) and mi	d.widget.FrameLayout(0,0)[DropDownSelect	
 (1) android.widget.FrameL 			XPath		.widget.FrameLayout[1]	Buttons	
 (0) android.widget.Line 			Layout			Buttons	
	LinearLayout[21,129][779		Height	1205		Bars	
P (U) android.wid	lget.ListView[21,129][779,	40	Width X	800		Views	
			Ŷ	0			
		1.4	Properties				
			checkable	False			
			checked	false			
			clickable enabled	false true			
			focusable	false			
			focused	false			
			scrollable	false			
			selected	false			

In the example above, we are displaying the sample Android application that comes with Rapise (AUTAndroid).

If you click on one objects in the user interface, it will be highlighted in Red and the tree hierarchy on the left will expand to show the properties of that object:

Get Snapshot (Ctrl-G)	 Page Source Go To Url Test Locator 	-	lect Profile android-nexus7	×	Discovery Mo Select Context	_	APP +	
Connection	Device		Profile		C	ontext		
Connection Device Device Device (0.0)[800,1205] (0) android view View(0.0)[800,1205] (0) android vieget LinearLayout[0.33][800,108] (0) android view.View(0.33][800,108] (0) android view.View[0.33][800,108] (0) android view.View[0.33][800,108]			Calculated Automation Id IndexInParent TagName Text		idget.TextView		a a a AUTAndroid	¢≗244 :
 (0) android.widget.Li (1) android.widget.Li 			Title	Drop Down Select (1) android.widget. Text View [21, 1941]		21,1941	DropDownSelect	
(1) android.widget.FrameLayout			XPath		widget.FrameLayou		Buttons	
	rLayout[21,129][779,45	3] '9,193]	Layout Height Width X Y	64 758 21 194			Bars Views	
(3) android.widg	et. TextView[21,259][77 et. TextView[21,324][77 et. TextView[21,389][77	9,388]	Properties checkable checked clickable enabled focusable focused acrollable selected text	false false true true false false false false false false	nSelect			

If you want to view the contents of the Spy as a text file, just click the **'Page Source'** button and you will see the contents of the Spy properties window as a text file. If you want to perform an action on the application (e.g. click on the selected item), switch the ribbon to the 'Events' view:

Spy	Events			
Тар	Precise Tap 💌	Text	Accept Alert	
Swipe 🔻	Scroll To	Send Keys	Dismiss Alert	
Shake		Execute Script	Change Orienta	ation
	Touch	Text	Misc	

For a full list of the available events and what they mean, please refer to the main Mobile Spy Topic.

Assuming that you can see your application in the Spy and that the objects can be inspected (similar to that shown above) you can now begin the process of testing your mobile application.

Click on **Disconnect** to end your Spy session and close the Rapise Spy dialog. You will now will be returned back to your test script.

3) Recording and Playing a Test

With the new Rapise mobile test script open, click on the **Record/Learn** button in Rapise and that will display the <u>recording activity dialog</u>:

t Object	Action	Data	Comment		
Verify (Ctrl+1)	Learn (C	trl+2)	Spy(Ctrl+5)	Pick Object	Resume

Now click on the [Pick Object] button and the Rapise Spy will be displayed in Recording Mode:

Spy Events				Rapise Spy		
Get Snapshot (Ctrl-G) Connection	Learn Object Test Locator Device	Select Profile Edit Profiles	android-nexus7	Recording M Select Context	Iode NATIVE_APP v Context	
(1) android.widge(2) android.widge(3) android.widge	0,33][800.108] 0,108][800.1205] out[0,108][800,1205] Layout[21,129][779,453]	 Calculated Automation Id IndexInParent TagName Text Title XPath Layout Height Width X Y Properties checkable checkable checkable checkable scrollable aelected text 		TextView et. TextView[21, 129][t. FrameLayout[1]/anc	AUTAndroid Login DropDownSelect Buttons Bars Views	₩ 8.04

We now want to record a click on one of the menu options, simply highlight one of the menu entries (e.g. "Login"):

Disconnect	Page Source	Select Profile	android-nexus7 -	Recording N	lode	
Get Snapshot (Ctrl-G) Connection	Learn Go To Url Object Test Locator Device	Edit Profiles	Profile	Select Context	NATIVE_APP -	
android.widget.FrameLayout(0.0)[800 (0) android.view.View(0.0)[900.1205]	.1205] 0.33][800.108] 0.108][800.1205] out[0.108][800,1205] Layout[21.129][779.453]	 Calculated Automation Id IndexInParent TagName Text Title XPath Layout Height Width X Y Properties checkable checkable checkable checkable dickable enabled focusable focusable focusable scrollable aelected text 	android Jid/Aext 1 0 android widget. Tex Login (0) android widget. T //android widget. F 64 758 21 129 false false false false false false false false false false false false false false false false false false	tView FextView[21,129][AutAndroid Login DropDownSelect Buttons Bars Views	****

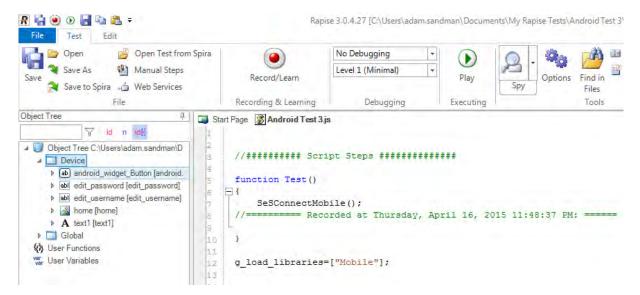
Now click the **[Learn Object]** button and the object will be added to the Rapise <u>object tree</u>. Now **on the device itself** click on the menu entry to go to the next screen, then in Rapise click **Get Snapshot** to get the updated screen:

Spy Events						
Tap Precise Tap ▼ Text Swipe ▼ Scroll To Send Keys Shake Execute Script Touch Send Keys android widget FrameLayout[0.0][800,1205] (0) android widget LinearLayout[0.03][80,108] 4 (1) android widget LinearLayout[0.108][800,1205] (0) android widget LinearLayout[0.108][800,1205] 4 (0) android widget TextView[21,129][779,158] [(1) android widget TextView[21,129][779,158] [(1) android widget Endit™ext[21,128][779,245] (2) android widget Endit™ext[21,126][779,303] (4) android widget Endit™ext[21,303][123,367] (5) android.widget TextView[21,367][779,396]		Text	com.inflec 1 android.wi (1) android	Accept Alert Dismiss Alert Change Orientation Misc ra.autandroid.ild/edit_usen dget.EditTexd widget.EditTexd[21.158][7 widget.FrameLayout[1]/ane	Login	
		focused scrollable selected text	true false false			

Now click on some of the objects and choose **Learn** to add them to the <u>object tree</u>. Once you are finished, click on the **Disconnect** button. You will see the events in the recording activity dialog:

#	Object	Action	Data	Comment	
2	lext1	Leam	Login	Learned text1	
13	edit_usemame	Leam		Learned edit_usemame	
14	edit_password	Leam		Learned edit_password	
15	android.widg	Learn	Login	Learned android.widget.Button	
₿6	home	Leam		Leamed home	
Ven	iy (0:(i=1)	Learn (C		Pick Object	Pause
Arel	25 (214-1)		Graduag	✓ Cancel	Finish (Ctrl+3)

Now click on the **Finish** button and you will be taken back to the test script with the Android objects listed:



Now that we have the objects, we can drag them into the test script editor and write the following script:

```
SeS('home').DoAction();
}
g_load_libraries=["Mobile"];
```

This will click on the first menu entry, then enter a username and password and then finally return back to the main menu.

Now to playback the test simply click **Play** in the Rapise test ribbon and the test will play back in the mobile device:

#	Туре	Start	 Name 	Status	Comment	Iteration
*	A	-	A	A	•	-
	Message	23:50:41.088	Starting scenario: Test	Info		
	Assert	23:50:56.250	text1.DoClick([])	Pass	Returned Value: true	0
	Assert	23:51:05.477	edit_username.DoSetText(["test user"])	Pass	Returned Value: true	0
	Assert	23:51:14.211	edit_password.DoSetText(["test pwd"])	Pass	Returned Value: true	0
	Assert	23:51:18.253	android.widget.Button.DoClick([])	Pass	Returned Value: true	0
	Assert	23:51:19.129	home.DoAction([])	Pass	Returned Value: true	0
•	Test	23:51:19.131	Android Test 3	Pass	Passed:5 Failed:0	

This is the report of the test being executed.

Example

You can find the Android sample tests and sample Application (called AUTAndroid) in your Rapise installation at the following locations:

Sample Android Tests:

- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AppAndroid (testing a native App)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\WebAndroid (testing a web app)

Sample Application (AUTAndroid)

• C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTAndroid

(we supply the sample application as both a compiled .apk binary and an Android Studio Java project with source code)

See Also

• <u>Technologies - Mobile Testing</u>, for instructions on preparing your environment for mobile testing, including instructions for installing the necessary prerequisites and configuring the various third-party components that Rapise uses to connect to the device.

2.3.10 Manual Testing

Purpose

Manual testing is used for situations where automated testing does not make sense. This may be due to

technical reasons (the application being tested does not have an API that lets tools such as Rapise interact with them) or economic (this part of the application is rarely used and the user interface is changing in each release).

However Rapise can help **accelerate and optimize** your manual testing as well. Rapise lets you rapidly create manual tests 5x faster than creating them by hand. It integrates with Spira for test management, so you still have a central repository of version-controlled test cases, but Rapise allows you to <u>edit them</u> <u>offline</u> when you have no connection to Spira and also to <u>execute them from within Rapise</u>.

Usage

To start manual testing, simply create your test as normal using the <u>New Test</u> dialog box. Then once the test has been created, click on the "Manual Steps" icon in the Test ribbon and then you will be taken to the <u>Manual Editor</u> with the <u>Manual Test Ribbon</u> Visible:

File	Test	Manual			
Save	Reload Save Local	Add Step Remove Step	 View in Browser New Incident 	Release Release	Kecord Screenshots Whole Window Manual With Cursor
	Spira	Editor	External Tools	Playback	Recording

From here you can start creating your new manual test using the <u>Manual Recorder</u>, then edit the created test steps in the <u>Manual Editor</u>. Finally you can <u>save the test to Spira</u> and then play it back using the <u>Manual Playback</u> and <u>Incident Logging</u> screens.

In addition to being used for manual testing, the test step editor lets you view the test steps that define the test scenario so that when you automate the test case, you can easily tie back specific <u>verification</u> <u>points</u> with test steps in <u>Spira</u>.

Finally you can also have the best of manual and automated testing in the same test script, using <u>semi-</u><u>manual</u> testing. That allows you to automate some of the repetitive tasks in a primarily manual test case.

Example

For a full tutorial using the manual playback, refer to the Exploratory Testing tutorial.

In addition, a working sample of manual testing is available from <u>Spira</u>, simply connect to the sample "Library Information System" project and open the 'Ability to Create New Book (TC2)' test case in the "Functional Tests" folder of the project. That will then display the sample manual test within Rapise:

🚳 Start Page	Nanual Steps.mt		v
 Ability to c 	reate new book		
StepId	Description	Expected Result	Sample Data
🐿 Step 1	Call		
[TS:1]			
🕐 Step 2	User clicks link to create book	User taken to first screen in wizard	
[TS:2]			
🕅 Step 3	User enters books name and author, then clicks Next	User taken to next screen in wizard	Macbeth, William Shakespeare
[TS:3]			
🕐 Step 4	User chooses book's genre and sub-genre from list	User sees screen displaying all entered	Play, Tragedy
[TS:4]		information	
🕐 Step 5	User clicks submit button	Confirmation screen is displayed	
[TS:5]			

See Also

- Manual Recording
- <u>Manual Playback</u>
- Exploratory Testing Tutorial
- Dialogs, Views and Menus
 - o Manual Ribbon
 - o Manual Test Editor
 - o Manual Playback
 - o Incident Logging

2.3.10.1 Manual Recording

Purpose

As described in the main <u>Manual Testing</u> topic, sometimes it is not possible to automate the testing of a specific application, however Rapise is also a powerful manual test generation system that can help you **create test cases 5x faster** than simply creating test cases by hand step by step.

This section describes how you can record a set of steps automatically by **simply using the application being tested**. Unlike an automated test however, Rapise will store a human-readable description of what was performed along with a <u>screenshot</u>, rather than actual computer code that can be played back by a computer.

Step 1 - Creating a New Test

To start manual testing, simply create your test as normal using the <u>New Test</u> dialog box. Then once the test has been created, click on the "Manual Steps" icon in the Test ribbon and then you will be taken to the <u>Manual Editor</u> with the <u>Manual Test Ribbon</u> Visible:

File	Test	Manual			
Save	Reload Save Loca	Add Step	S View in Browser	Release Execute Manual	✓ Record Screenshots ✓ Record Screenshots ✓ Whole Window Manual With Cursor
	Spira	Editor	External Tools	Playback	Recording

The test step list will initially be empty:

🖾 Start Page	📓 MS-Paint Test TC34.js 🎕 Manual Steps.rmt		
MS-Paint 1	Test		
0	Description	Expected Result	Sample Data
StepId	Description	Expected Result	Sample Data
Stepia	Description	Expected Result	Sample Data

Step 2 - Recording Some Steps

Now you should open up the application you want to record from. In this example we shall be testing the built-in **Microsoft Paint** application. This is a good candidate for manual testing as a lot of the functionality is hard to test automatically since there is a simple drawing canvas rather than discrete buttons and data elements to test.

To make sure that we have screenshots recorded, whilst keeping the size of the screenshots reasonable, use the following recording options:

100	1	Record Screenshots
		Whole Window
Record Manual	1	With Cursor
	F	lecording

Now click the '**Record Manual**' button and choose MS-Paint from the list of running applications in <u>Select Application to Record</u> dialog and then click '**Select'** to start recording.

As you click through the application, the recording will display the list of steps and actions being captured:

#	Object	Action	Data	Comment
11	Application	LClick	37,11	User clicks at: 37, 11 in 'Application menu'
12	Application	LClick	42,12	User clicks at: 42, 12 in 'Application menu'
13	New	LClick	44,13	User clicks at: 44, 13 in 'New'
14	Pencil	LClick	15,9	User clicks at: 15, 9 in 'Pencil'
15	Text	LClick	14,16	User clicks at: 14, 16 in 'Text'
16	Simulated	LClick	156,256	User clicks at: 156, 256 in "
17	Text	SetText	This is some	: Change text to 'This is some text'
8	Bold	LClick	11.14	User clicks at: 11, 14 in 'Bold'
Verif	y (Ctrl+1)	Learn (Ctr	1+2)	Spy(Ctrl+5) Pick Object Resume
Analo	og (Ctrl+4)		_Simulated	✓ Cancel Finish (Ctrl+3)

In this example, we created a new canvas, chose the Pencil tool, created a drawing using the pencil, entered some text and then made it bold:

Home	Text Tools View Text	Untitled - Paint		
V ot	libri	A Opaque A Transparent Background	colors	Edit colors
M				
		$\langle \rangle$	\sum	
	This is some t	ext •		a:
	0	Δ		
		0		

When you click **Finish** to complete the recording, Rapise will now display the list of populated manual test steps with the embedded screen captures:

MS-Paint	est		
StepId	Description	Expected Result	Sample Data
Step 1	User clicks at: 37, 11 in 'Application menu'		SeS('Application_menu').DoLClick(37, 11);
Step 2	User clicks at: 42, 12 in 'Application menu'		SeS('Application_menu').DoLClick(42. 12);
1 Step 3	User clicks at: 44, 13 in 'New'	l k₩	SeS('New').DoLClick(44, 13);
Step 4	User clicks at: 15, 9 in 'Pencil'	<u>₩</u>	SeS('Pencil').DoLClick(15, 9);
Step 5	User clicks at: 14, 16 in 'Text'	A	SeS('Text'). <u>DoLClick(14, 16);</u>
Step 6	User clicks at: 156, 256 in "	37.2705	SeS('Simulated').DoLClick(156, 256);

You will notice that the description of each test step will use the form "User [action] at [coordinates] in '[object name]'" and the expected result will include the screenshot of what the user was doing. In addition, the sample data will contains the equivalent Rapise automation code for reference. This can be useful later if you decide to automate this test.

Step 3 - Editing the Steps

Typically you may want to **add some additional steps** (e.g. we added a line to describe the process of starting up MS Paint), **delete any duplicate/unnecessary steps** and **reword them** so that they make the most sense to the tester. In our example we used the <u>manual editing</u> screen to update the steps as follows:

MS-Paint	Manual Steps.rmt~ est		
StepId	Description	Expected Result	Sample Data
Step 1	User starts up the MS-Paint Application	The application loads with a blank canvas	
Step 2	User clicks the main 'Application menu'		SeS('Application_menu').DoLClick(42, 12);
Step 3 *	User clicks the 'New' entry	Lew .	SeS('New').DoLClick(44, 13);
Step 4	User clicks on 'Pencil'		SeS('Pencil') DoLClick(15, 9);
Step 5	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);
C Step 6	User clicks at: 156, 256 in the canvas	72700	SeS('Simulated').DoLClick(156, 256);

Click **Save** to make sure the updates are all saved locally. Now before you can <u>execute these tests</u>, you will need to Save them to <u>Spira</u> (our web-based test management system).

Step 4 - Saving to Spira

Click on the option to **Save to Spira**, you will be asked to confirm the creation of the document folder in Spira that will hold the test files:

apise will create a new	folder in the selected Spira project to store th	he test script and ass	ociated files:
acal Repository Path	C:\SpiraRepository\MS-Paint Test TC34		
ocal nepository Fain.	C. Opiral repository (WO-Faillt Test TC34		
Remote Folder:	Root Folder/MS-Paint Test TC34		

Click on '**Create'** and then the manual test will be saved to Spira. You will see that this process adds the unique Spira test step IDs to each step. They are displayed using the format [TS:xxx]. This special token [TS:xxx] can be used in Tester.Assert commands to relate specific verification points with test steps during automated testing.

	Manual Steps.mt			- >
MS-Paint StepId	Description	Expected Result	Sample Data	-
[TS:47]		Law Law		-
(TS:48]	User clicks on 'Pencil'	<u>×</u>	SeS('Pencil').DoLClick(15, 9);	
(TS:49)	User clicks the 'Text' tool	4	SeS('Text').DoLClick(14, 16);	1
1 Step 6 [TS:50]	User clicks at: 156, 256 in the canvas	19200 FR IF	SeS('Simulated').DoLClick(156, 256);	
(1) Step 7 [TS:51]	Enters text 'This is some text '	This is some text	SeS('Text1'').DoSetText("This is some text/r");	
Step 8	User clicks on the 'Bold' button		SeS('Bold').DoLClick(11, 14);	-

Now that the test has been saved in Spira, you can click on the 'View in Browser' option to see how the test steps look inside Spira.

Ins	ert St	tep Insert L	<u>ink Delete Copy Refresh.</u>	- Show/hide columns 🔻	Edit Parameters			
	U	Step #	Description	Expected Result	Sample Data	Execution Status	ID	► Edit
		1 Step 1	User starts up the MS- Paint Application	The application loads with a blank canvas		Not Run	TS000045	► Edi
	U	Step 2	User clicks the main 'Application menu'		SeS('Application_menu').DoLClick(42, 12);	Not Run	TS000046	► Edi
	U	1 Step 3	User clicks the 'New' entry	Lew	SeS('New').DoLClick(44, 13);	Not Run	TS000047	► Edi
	U	External Step 4	User clicks on 'Pencil'	1	SeS('Pencil').DoLClick(15, 9);	Not Run	TS000048	► Edi
	U	E Step 5	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);	Not Run	TS000049	► Edit
	U	Step 6	User clicks at: 156, 256 in the canvas		SeS('Simulated').DoLClick(156, 256);	Not Run	TS000050	► Edit
	U	🕲 <u>Step 7</u>	Enters text 'This is some text '	This is some text	SeS('Text1').DoSetText("This is some text\r");	Not Run	TS000051	► Edi
	U	Step 8	User clicks on the 'Bold' button	R	SeS('Bold').DoLClick(11, 14);	Not Run	TS000052	► Edi

Now this test case is ready for manual playback.

See Also

- Manual Testing
- Manual Playback

2.3.10.2 Manual Playback

Purpose

As described in the main <u>Manual Testing</u> topic, sometimes it is not possible to automate the testing of a specific application, however Rapise is also a powerful manual testing tool that lets you execute manual test cases stored in <u>SpiraTest</u>.

The advantage of using Rapise to execute the manual tests (instead of just using SpiraTest itself) is that Rapise can display the <u>execution window</u> as a small minimizable dialog box that gets rid of the need to have two screens (one to display the test and one to test the application). Also Rapise provides superior <u>image manipulation tools</u> over those available in a web application.

Step 1 - Open the Manual Test

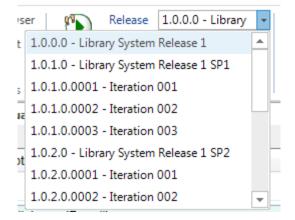
Using the MS-Paint example manual test that <u>we created previously</u>, open up the test in Rapise. Click on the 'Manual Steps' icon in the <u>Test ribbon</u> and you should see the list of test steps:

Start Page	Manual Steps.mt			-
StepId	Description	Expected Result	Sample Data	
[TS:47]		La Sew		
Step 4 [TS:48]	User clicks on 'Pencil'	*	SeS('Pencil').DoLClick(15, 9);	
Step 5 [TS:49]	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16);	
(TS:50)	User clicks at: 156, 256 in the canvas			1
(TS:51)	Enters text 'This is some text '	This is some text	SeS('Text1').DoSetText("This is some text\r");	ĺ
Step 8	User clicks on the 'Bold' button	8	SeS('Bold').DoLClick(11, 14);	-

Now that we have the test opened, we can start the playback

Step 2 - Executing the Manual Test

Choose the Release from the list of those available in the project:



Then click on the 'Execute' icon to start manual test execution. That will bring up the <u>manual playback</u> screen:

282 Rapise User Manual

Executing MS-Paint Test	-	
NS-Paint Test		×
Image: Big in the second se		
User starts up the MS-Paint Application		
	Control Data	
Expected Result: The application loads with a blank canvas	Sample Data:	
Actual Result- Please enter the actual test result if it differs, then click the appropriat	e button:	
[II] B I U A Arial ▼ A ³ 10 ▼ A ▼ ♥ ▼ ■ ■ ■ II II = III		
Add Attachment		

On this screen, we shall follow through the steps listed in the test case. This involves opening up MS Paint, creating a new canvas, adding some lines using the pencil and then adding some text using the text tool. As you perform these steps, click on the **Pass** button to indicate that each step has passed. You can also minimize the manual playback screen by clicking the > | button.

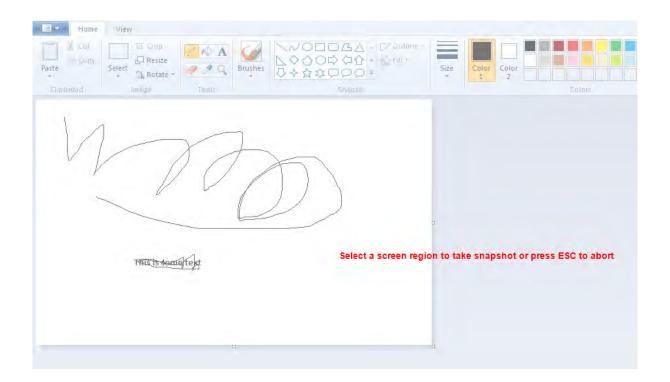
Once you get to Step 7, we shall pretend that MS Paint failed to display the text correctly. Enter in the Actual Result a message to that effect:

Executing MS-Paint Test		
MS-Paint Test		>
Image: Big and the state of the state o		
Enters text 'This is some text '		
Expected Result:	Sample Data:	
This is some text	SeS('Text1').DoSetText("This is some text\r");	
Actual Result- Please enter the actual test result if it differs, then click the appropriate b The text did not display correctly.	utton:	
🖾 B I U A Arial - A ^a 10 - A v 🥙 - 🖃 = 🗄 🗄 ∞		j,
Add Attachment Log Incident		

Next we shall attach a screenshot of what actually happened and log a test failure and associated incident / defect.

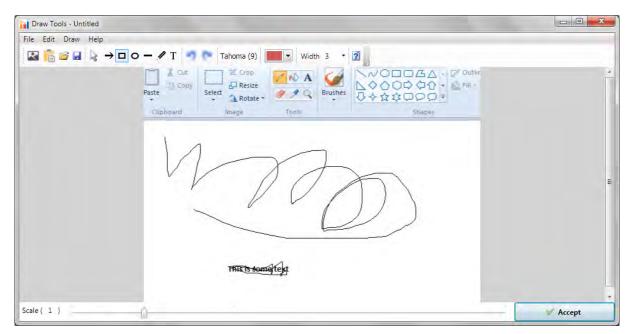
Step 3 - Capturing and Annotating a Screenshot

Click on the **Image icon** in the rich text editor associated with the **Actual Result** text box. That will bring up the <u>Drawing Tools</u> screen that asks you to draw a rectangle to select a portion of the current screen to capture:



If the MS Paint application is not in the foreground, just click ESC on your keyboard to abort, rearrange your windows and then try again.

Once you have selected the rectangle, the drawing tools will display your selected image in the image editor:



You can now use the annotation tools to add labels, text and other items to explain the issue that you found:

Draw Tools - Untitled *	
File Edit Draw Help Image: Imag	
Image	
MAD	ja R
THIS IS sourced to be	
Scale (1)	🖌 Accept

In the example above, we added a red ellipse, arrow and text to mark the issue that was seen in MS-Paint. Once you are happy with your image, click **Accept** and the image will be included in the test Actual Result:

🛐 Executing MS-Paint Test		
MS-Paint Test		2
Image: Step 7 Step 7 8 Pass 4 Blocked 6 Caution 1 Fail X		
Enters text 'This is some text '		
Expected Result:	Sample Data:	
This is some text	SeS('Text1').DoSetText("This is some textir");	
Actual Result- Please enter the actual test result if it differs, then click the appropriat	te button:	
THIE'S some tel	Wittong	
□ B I U A Arial • A ^a 10 • A • ♥ • ■ ■ □ □ □ □ □		a.
Add Attachment Log Incident		

Now we can log an incident that is associated with this test failure.

Step 4 - Logging the Incident / Defect

New Incident *	1 manual and the second s		
🕞 Save			
🕙 Details / Descrip	tion		×
Name: The text di	d not display correctly in MS Paint		
Description:			
BIU AAr	al 🔹 🖌 🗛 10 🔹 📥 🗷 🖽 🖽		
	Pro-		
Туре:		Status: New	
Type: Detected By:		•	
Detected By:			
Detected By: Priority:	Fred Bloggs	•	
Detected By: Priority:	Fred Bloggs 2 - High 1.0.0.0 - Library System Release 1	Severity: None	

Click on the 'Log Incident' button to display the new incident entry screen:

Choose the **type** of incident, enter the **name**, **description**, **priority**, **detected release** and any other required fields as defined by the workflow in the project that you are connected to. Once you have entered in the various fields, click the **'Save'** icon in the top left.

This will return you to the <u>manual execution</u> screen with the **Incident ID** [IN:xxx] and **name** displayed at the bottom. Now click on the '**Fail**' button and the test case will be marked as failed:

NS-Paint Test		>
Image: Big I with a start of the start	Finish	
User clicks on the 'Bold' button		
Expected Result:	Sample Data:	
R	SeS('Bold').DoLClick(11, 14);	
Actual Result- Please enter the actual test result if it differs, then click the ap	propriate button:	
🖾 B I U A Arial 🔻 Að 10 🕶 🛕 🔹 🗐 🗄 🗄	2	ja
Add Attachment		
[IN: 71] Did not display the correct text		

Finally, click on the Finish button and the results will be posted to Spira.

Step 5 - Viewing the Results

Now to view the results in Spira, click on the <u>Spira Dashboard</u> item in the main Rapise <u>Test ribbon</u>. Then under the 'My Created' test cases, click on the link for the test case you execute. That will bring up the test case in Spira. Now click on the 'Failed' hyperlink in Spira and the new test run will be displayed:

Test Run: 🗎	NS-Paint Test [TR:000041]			
Overview *	Attachments # Incidents #			
▼ Details				
Release #:	1.0.0.0 - Library System Release 1	Estimated Duration:	hours	
Tester Name:*	Fred Bloggs	Actual Duration:	0.05 hours	
Test Set:		Execution Date:	3/31/2015 1:54:11 PM	
Test Case #:*	TC000034	Execution Status:*	Failed	
Build:	None 🖤	Test Run Type:*	Manual	
Web Browser:	Please Select V	Operating System:	Please Select	▼
Notes:	Font			

If you scroll down, you can see the individual test steps that were executed, with the associated actual result (including the captured screenshot):

Test Steps						
D	Test Step Description	Expected Result	Sample Data	Test #/ Step #	Actual Result	Execution Statu
RS000084	User starts up the MS- Paint Application	The application loads with a blank canvas		TC000034 / TS000045		Passed
R\$000085	User clicks the main 'Application menu'	10 N	SeS ('Application_menu').DoLClick (42, 12);	<u>TC000034</u> / <u>TS000046</u>		Passed
RS000086	User clicks the 'New' entry	- Com	SeS('New').DoLClick(44, 13);	TC000034 / TS000047		Passed
RS000087	User clicks on 'Pencil'	2N	SeS('Pencil').DoLClick(15, 9);	TC000034 / TS000048		Passed
RS000088	User clicks the 'Text' tool	A	SeS('Text').DoLClick(14, 16):	TC000034 / TS000049		Passed
RS000089	User clicks at: 156, 256 in the canvas		SeS('Simulated').DoLClick (158, 258);	TC000034 / TS000050		Passed
RS00000	Enters text "This is some text"	This is some text	Ges((Text)), DoGetText("This is some text");	1000034 / 15000051	Failed with the text being lifegible.	Falled
RS000091	User clicks on the 'Bold' button	R.	SeS('Bold').DoLClick(11, 14);	TC000034 / TS000052		Not Run

If you click on the **Incidents** tab, you can also see the new incident that was logged, linked to this test run:

Overview *	Attachments *	Incidents #								
Display List of Incidents: > Refresh Apply Filter Clear Filter Show/hide columns V										
Displaying 1 - 1 out of 1 incident(s) linked to this test run. Filtering results by Test Run #. (Clear Filters)										
🗸 🕘 Name 🔺 🖲	'	Type ▲▼	Status ▲▼	Priority ▲▼	Detected By ▲▼	Creation Date ▲▼	Owner ▲▼	Progress	ID ∆ ▼	Edit
		Any 🛛 🔻	- Any 🔻	Any 🔻	- Any 🔻		Any 🛛 🔻	' Any 🔻		► Edit
Did not	display the correct text	Incident	New	2 - High	Fred Bloggs	31-Mar-2015			IN:000071	► Edit
Show 15 V rows per page 1 S of 1 N										

Congratulations! You have now successfully executed a manual test using Rapise.

See Also

- Manual Testing
- Manual Recording

2.3.10.3 Semi-Manual Testing

Purpose

This is a useful technique when you want to have a predominantly manual test (executed by a tester) that has some steps that are automated by Rapise. These could be some of the initial setup tasks (e.g. logging in, starting the application) or just tasks that are well suited to automation.

Usage

Create your manual test either using the <u>recorder</u> or the <u>manual test editor</u>. You can also just open up a test already created in <u>Spira</u>.

Next, inside Rapise, create a <u>test scenario</u> (function) that contains the necessary login. In this example we shall simply automate the launching of MS-Paint.

Create a function in the MyTest.user.js file with the following code:

```
function LaunchMsPaint()
{
    Global.DoLaunch('C:\\Windows\\system32\\mspaint.exe');
}
```

Now go to the Manual Steps section of Rapise by clicking on the Manual Steps icon in the test ribbon:

Inside the first test step (for example), change the **Description** to the following:

```
@LaunchMsPaint();
//User starts up the MS-Paint Application
```

This will be contained within the actual test step itself:

StepId	Description	Expected Result
1 [TS:45]	@LaunchMsPaint(); //User starts up the MS-Paint Application	The application loads with a blank canvas
1 Step 2 [TS:46]	User clicks the main 'Application menu'	

Now, when you execute the test (using the normal **Execute** button on the main <u>Test ribbon</u> (<u>not</u> the Execute Manual icon on the <u>Manual Steps ribbon</u>) what happens is that Rapise will execute the main Test() function that contains:

```
function Test()
{
    Global.DoPlayManual();
}
```

```
g_load_libraries=["Generic"];
```

this instructs Rapise to use the <u>manual playback</u> system. However when it gets to the first step, it will see the ampersand symbol (@) that denotes that this is actually an automated scenario and then call the following code:

//User starts up the MS-Paint Application
LaunchMsPaint();

Once the scenario has completed, Rapise will then return back to the manual test playback.

See Also

- Manual Playback
- Test Scenarios

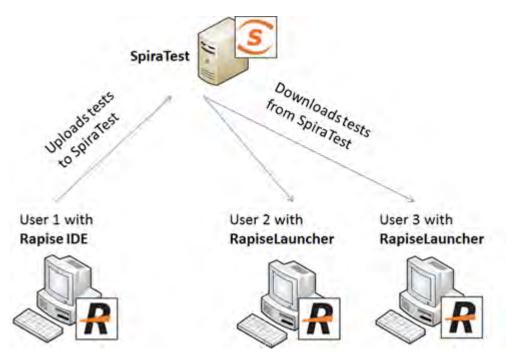
2.3.11 SpiraTest Integration

For more details on using SpiraTest with Rapise, please refer to the separate "Using SpiraTest with Rapise" guide.

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. *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.

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.

Connecting to SpiraTest

The first thing you need to do is to configure the connection to SpiraTest. To do this, click on the <u>Options ribbon</u> in Rapise:

File	Test	Options					
- Carlor - C	🔍 Build	d OCR Font [Database	sé	Selenium Settings	<u>\$</u>	Java Settings
Settings	🕤 Rese	t Layout		ø	Web Spy Settings	L.	Mobile Settings
bettings	🖲 Spira	a Connection	n Settings	٩	Browser Settings	G	NeoLoad Integration Settings
	G	eneral			Web Testing		Other Testing

Click on the "Spira Connection Settings" button in the General tab to bring up the dialog box that lets

you configure the connection to SpiraTest:

URL:	http://doctor/spirateam
User Name:	administrator
Password:	
Default Repository Path:	C:\Temp\RapiseTests
Token (read only):	

Enter the URL, login and password that you use to connect to SpiraTest and then click the "Test" button to verify that the connection information is correct.

- The "Default Repository Path" is a folder that used to store local copies of the non-absolute repositories.
- The **Token** is the identifier of the current machine that Rapise is installed on. It needs to match the 'Token' name of the corresponding 'Automation Host' in SpiraTest.

You need to be running SpiraTest / SpiraTeam v4.0 or later to use the integration with Rapise.

Creating a Rapise test from a SpiraTest test case

To create a new Rapise test based on the manual test steps already defined in a SpiraTest test case,

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:

Create or Open Test				
Server:	http://doc	ctor/spirateam		
Project:				
Fest Cases:			New Folder	New Test Case
Fest Steps				
Test Steps Step Descri	ption	Expected Resul	t Actual	Result
	ption	Expected Resul	t Actual	Result
	ption	Expected Resul	t Actual	Result
		Expected Resul Specify methodology:	t Actual	Result
			t Actual	Result
Basic: Manual Scripting M	ode			

- Select the project that has our new test case. The list of test case folders will be displayed.
 You can create a new folder by clicking the **New Folder** button
 Expand the folders until you can see the desired test case:

Create or Open Tes	st			
Server:	http://do	ctor/spirateam		
Project:				
Library Information S	öystem			
Test Cases:			New Folder	New Test Case
Ability to	create new book edit existing book create new author edit existing author reassign book to differen	tauthor		E
Repression T	the same of the second s			-
Test Steps	the same of the second s	Expected Result	Actual F	Result
Test Steps	Description		Actual F	
Basic: Manual Script	Description	Expected Result		

Now either create a new test case by using the **New Test Case** button or simply click on a test case that you previously created in Spira. In either case you will see its test steps displayed underneath (if there are any):

Serve	er.	http://doctor/spira	lean			
Projec						
Libr	ary Informati	on System				
Test (Cases:			New Folder	New Test Case	
	- 전 Abilit 전 Abilit	y to edit existing book y to create new author y to edit existing author y to reassign book to different author				m
Test :						+
	Rearessi		Expected Result	Actual Re	esult	*
	Regressi Steps	nn Tests			esult	+
	Regressi Steps	Description		n w	esult	* * *
	Steps Step 1	Description User opens up Internet Explorer a	The browser loads the logi	n w	esult	+ + +
	Steps Step 1	Description User opens up Internet Explorer a User logs in to application	The browser loads the logi User taken to main menu s	n w		+ × +
Test :	Steps Step 1 2	Description User opens up Internet Explorer a User logs in to application	The browser loads the logi User taken to main menu s	n w		
Test :	Steps Step 1 2	Description User opens up Internet Explorer a User logs in to application Specify	The browser loads the logi User taken to main menu s III methodology:	n w	} ,	

Once you are satisfied that this is the correct test case, choose the desired **methodology (Web, Mobile or Basic)** and then click the **Create from Spira** button. Rapise will now create a local test folder and files based on this Spira test case.

Saving a Test to SpiraTest

To save the a Rapise test into SpiraTest you need to make sure that the following has been setup first:

- 1. You have a project created in SpiraTest to store the Rapise tests in. The Rapise tests will be stored in a repository located inside the **Planning > Documents** section of the project.
- 2. The user you will be connecting to SpiraTest with has the permissions to create new document folders.
- 3. You have created the Test Case in SpiraTest that the Rapise test will be associated with. This is important because without being associated to a SpiraTest Test Case, you will not be able to schedule and execute the tests using SpiraTest and RapiseLauncher.
- 4. You have created an AutomationEngine in SpiraTest that has the token name "Rapise". This will be used to identify Rapise automation scripts inside SpiraTest.

Once you have setup SpiraTest accordingly, click on the **Save to Spira** icon in the File section of the Rapise Test ribbon:

File Test Options					
Gen Save Save to Spira	 Start Page Spira Dashboard Test Manual Steps Functions L Web Services 	Record Spy		About Convert to NeoLoad Activation Spira Properties Object Mar	Find in Help Files
File	Shortcuts	Record and Execute	Debugging	Tools	Help

That will bring up the Save to SpiraTest dialog box:

Server:	http://localhost/spira		Change	Refresh
Project:				
Library Information System				
Test Case Folders:				
 Library Information System Functional Tests Regression Tests Scenario Tests Common Tests 				
Name		D	Repository Path	

The first thing you will need to do is choose the SpiraTest project from the dropdown list. This will then update the list of test case folders displayed in the top pane of the dialog box.

Once you have chosen the desired project, you need to expand the test case folders in SpiraTest and choose the existing Test Case that you want to attach the Rapise test to:

Server: http://loca	Ihost/spira	Change	Refresh
Project:			
Library Information System			
Test Case Folders:			
 Functional Tests Regression Tests Scenario Tests Exception Scenario Tests Common Tests 			
Name	ID	Repository Path	
Name Maility to create new book	ID TC2	Repository Path Click to view	
Ability to create new book	TC2	Click to view	
Ability to create new book Ability to edit existing book	TC2 TC3	Click to view Click to view	
Ability to create new book Ability to edit existing book Ability to create new author	TC2 TC3 TC4	Click to view Click to view Click to view	
Ability to create new book Ability to edit existing book Ability to create new author Ability to edit existing author	TC2 TC3 TC4 TC5	Click to view Click to view Click to view	

When you expand the folders to display the list of contained test cases, it will display the name and ID of the test case together with an icon that indicates the type of test case:

- 1. 🔄 Manual test case that has no automation script attached. (Repository Path will also be blank)
- 2. 4 Test case that has an existing Rapise test attached.
- 3. 🖆 Test case that has a non-Rapise automation script attached.

If you are adding a new Rapise test, choose a test case that has icon (1) and doeesn't have an associated Repository path. If you are updating an existing test, choose a test case that has icon (2) and the matching Repository path.

Note: test cases with icon type (3) cannot be used with Rapise for adding or updating scripts.

Once you have chosen the appropriate test case, click the [Save/Synchronize] button. That will bring up the **Create New Repository** dialog box:

Rapise will create a new	folder in the selected Spira project to	store the test script and associated files:
Local Repository Path:	C:\Users\adam.sandman\Document	ts\My Rapise Tests\Spira New Test1
Remote Folder:	Root Folder/Spira New Test 1	

This dialog box will let you know where the Rapise script will be stored in SpiraTest and also the location of the repository local directory used to store the 'working copy' of the Rapise test. Click [Create] to confirm.

Files To			
Selected	File Name	Action	Path
1	Spira New Test1.js	Upload	C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1
1	Spira New Test1.objects.js	Upload	C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1
>>>>>	Spira New Test1.sstest	Upload	C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1
D.3.	Spira New Test1.sstest.us_	Upload	C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1
	Spira New Test1.user.js	Upload	C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1
1	Repository json	Download	C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1

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:

The system will display the message that it's saving the files to the server:

Uploading to SpiraTest Server	×
Uploading	
	_
	Cancel

If an error occurs during the save, a message box will be displayed, otherwise the dialog box will simply close.

Opening a Test from SpiraTest

To open a Rapise test from SpiraTest you need to make sure that the following has been setup first:

- 1. You have already configured the connection to the SpiraTest service (see the instructions at the top of this page).
- 2. The user you will be connecting to SpiraTest with has the permission to view the project that the tests are being stored in.

Once you have setup SpiraTest accordingly, click on the **Open Test from Spira** icon in the File section of the Rapise Test ribbon:

File	Test Options										
	🍃 Open	🖾 Start Page	🛐 Spira Dashboard		0		No Debugging	+	About	💪 Convert to NeoLoad	
Save	X Save to Spira	📓 Test	Manual Steps	Record	X	Play	Level 1 (Minimal)	•	? Activation	Spira Properties	Find in Help
	👙 Open Test from Spira	Functions	👍 Web Services	necord	Spy				Object Mgr		Files
	File	5	Shortcuts	Record	d and Ex	ecute	Debugging			Tools	Help

That will bring up the Open Test from SpiraTest dialog box:

Server	http://localhost/spira		Change	e Refresh
Project:				
Library Information System Test Case Folders:				1
 Library Information System Functional Tests Regression Tests Scenario Tests Common Tests 				
Name		D	Repository Path	

The first thing you will need to do is choose the SpiraTest project from the dropdown list. Once you have done that, the system will display the list of test case folders in this project.

Once you have chosen the project, you need to expand the test case folders in SpiraTest and choose the existing Test Case that you want to open:

Server ht	tp://localhost/spira	Change Réfre	sh
Project:			
Library Information System			
Test Case Folders:			
 Library Information System Functional Tests Regression Tests Scenario Tests Common Tests 			
Name	ID.	Repository Path	
Ability to create new book	TC2	CreateNewBook/CreateNewBook.ss	
Ability to create new book.	TC2 TC3	CreateNewBook\CreateNewBook.ss_ Click to view	
Ability to create new book Ability to edit existing book Ability to create new author	TC2 TC3 TC4	CreateNewBook/CreateNewBook.ss Click to view Click to view	
Ability to create new book	TC2 TC3 TC4 TC5	CreateNewBook\CreateNewBook.ss_ Click to view	

When you expand the folders to display the list of contained test cases, it will display the name of the associated Rapise test script associated with it (to the right). Choose a test case that has the matching Rapise test case listed to the right of it (in the Repository Path column).

Note: Only test cases that have an attached Rapise test script will be displayed in this view.

Once you have chosen the appropriate test case, click the [Load/Synchronize] button to load the test case:

Files Io	Upload:			
Selected	File Name	Action	Path	
1	Spira New Test1.user.js	Download	C:\Temp\RapiseTests\Spira New Test1	
1	Spira New Test1.sstest	Download	C:\Temp\RapiseTests\Spira New Test1	
1	Spira New Test1.objects.js	Download	C:\Temp\RapiseTests\Spira New Test1	
111	Spira New Test1.js	Download	C:\Temp\RapiseTests\Spira New Test1	
1	Repository.json	Download	C:\Temp\RapiseTests\Spira New Test1	

A dialog box will be displayed that lists all the files on the server which will be downloaded from SpiraTest. You can change which files are to be downloaded. Once you are happy with the list of files being checked-out, click the [OK] button:

The system will display the message that it's downloading the files from the server. If an error occurs during the download, a message box will be displayed, otherwise the dialog box will simply close.

Viewing the SpiraTest Properties of a Test

To see which SpiraTest **project** and **test case** the current Rapise test is associated with, click on the **Spira Properties** icon in the Tools section of the Rapise Test ribbon. This will bring up the Spira Properties dialog box:

Spira Properties	
Spira Project:	Library Information System
Test Folder:	Functional Tests
Test Case:	Ability to create new book
	> Unlink from Test Case

This will display the name of the current Rapise test together with the name of the SpiraTest project, test folder and test case that this test is associated with.

If you would to save the current Rapise test into a new SpiraTest project or if you want to save it against a new test case in the same project, you must first unlink the test. To do this click on the **Unlink from Test Case** button. This will tell Rapise to remove the stored SpiraTest information from the .sstest file so that it can be associated with a new project and/or test case in SpiraTest.

Warning: This operation cannot be undone so please make sure you really want to unlink the current test.

Using the Spira Dashboard

In addition to using the ribbon options described in this page, you can interact with SpiraTest using the <u>Spira Dashboard</u> that is available from the <u>Start Page</u>. This provides a convenient way of interacting with SpiraTest, allowing you to quickly create, save and open test cases from SpiraTest.

Using RapiseLauncher

RapiseLauncher is a separate application that installs with Rapise. It allows you to remotely schedule the automated tests in SpiraTest and have RapiseLauncher automatically invoke the tests according to the schedule. Details on using SpiraTest with RapiseLauncher to remotely schedule and execute tests is described in the separate **"Using SpiraTest with Rapise"** guide. This guide can be found in the Rapise program files folder. Click on Start > Programs > Inflectra > Rapise in Windows and you will see the shortcut for the guide.

2.3.12 Checkpoints

Purpose

A **Checkpoint** is defined by two things: (1) a location in the test execution path and (2) a subset of AUT state. Each time the checkpoint executes, the AUT state is compared to a predefined value. Discrepancies are noted, and may show a regression in program behavior.

Usage

A checkpoint can be added in two ways:

- (1) during recording, with the Verify Object Properties dialog, or
- (2) by manually adding an Assertion to the test script.

See Also

<u>Recording</u>

2.3.13 Tests and Sub-Tests

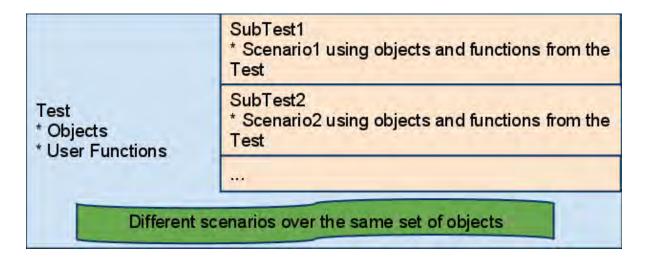
The concept of Sub-Test is an organic way to organize the whole work with Tests in organic way. By having sub-tests one may meet one of the following goals:

- 1. Create multiple test scenarios working with same set of Objects and Functions.
- 2. Organize different test scenarios into a single workspace.
- 3. Use Sub-test to make cross-browser tests

We will consider each of described goals separately. The test containing the sub-test(s) we will call **base** or **parent** test.

Make Multiple Test Scenarios with the Same Set of Objects

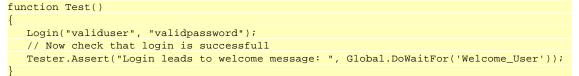
In this case 'parent' test contains all learned objects and user-defined functions.



For example, the parent test may have objects "User Name", "Password", "Sign On". And function

```
function Login(username, password)
{
    ...
}
```

SubTest1 may be used to check login with valid Credentials, SubTest1.js looks like:



SubTest2 may be used to check login with invalid Credentials (i.e. it is a fail-test). **SubTest2.js** looks like:

```
function Test()
{
   Login("invaliduser", "invvalidpassword");
   // Now check that login is successfull
   Tester.Assert("Login leads to invalid user object: ",
Global.DoWaitFor('Invalid_User'));
}
```

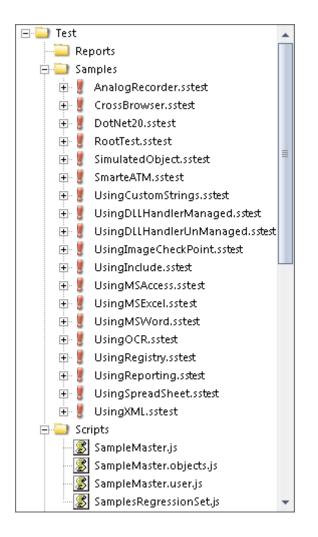
Function Login and objects Welcome_User and Invalid_User are defined in Test. The subtests are just implementing various scenarios for the same set of objects.

Organize different tests into a single workspace.

Each test has its own objects, functions and scenarios.

Test * No own Object or User Functions * Used to organize different tests into a single workspace	Test1 * Objects * User Functions * Scenario Test2 * Objects * User Functions * Scenario
Single workspace	for independent tests

The usage of such an approach is well demonstrated by example. We created a test called 'SampleMaster' and put all Rapise samples into it by using **Add File** context menu in the the <u>Test Tree</u> dialog. Finally the Files tree looks like:



All tests in this tree are independent. We use the Sample Master to manage all the tests from a single environment.

Using Sub-Tests for a Cross-browser testing

See 'Cross Browser'.

Sub-Test Features

• Sub-test may have its own nested sub-tests. For example, in the parent test contains reference to 'CrossBrowser' subtest having 'IE' and 'Firefox' subtests inside:

🖃 🔜 Test
» 🦲 Reports
🖻 🚄 Samples
🕀 🐰 AnalogRecorder.sstest
😑 🚪 CrossBrowser.sstest
🕀 🤜 Reports
🕀 🦲 Scripts
🕀 🚪 Firefox.sstest
🕀 🚪 IE.sstest

• Sub-test options are available from the 'Tag' property in the 'Properties window:

Test Files	4		
🖃 🦳 Test			
🖨 🦳 Samples	=		
🕀 📒 AnalogRi	ecorder.sstest		
🕀 📒 CrossBro	wser.sstest		
🕀 📒 DotNet2().sstest 🔹		
Test Files Object Tree	e Settings		
Properties	Д		
<u></u>			
Advanced	A		
🖂 Tag	T:\Admin\Wrappers\Sample		
BeautifySavedObje			
CacheObjects	False		
CommandInterval	1000		
CommandLine			
CustomStrings	(Collection)		
IterationsCount	1		
Metadata	T:\Admin\Wrappers\Sample		
Methodology	Basic 📃		
ObjectLookupAtter	500		
ObjectLookupAtter	10		
ObjectsPath	CrossBrowser.objects.js		
⊕ Options ■			

• The following options are available in the context menu fore each of the sub-tests:

Play
Record
Save
Show Objects
Delete

- Play: Execute selected sub-test
- Record: Start recording into selected sub-test
- Save: Save options of a sub-test
- Show Objects: Show objects form a sub-test in the Object Tree
- Delete: Remove reference to a sub-test from its parent test

2.3.14 NeoLoad Integration

This section describes the integration between Inflectra's Rapise functional testing tool and Neotys' NeoLoad performance testing tool.

The purpose of this integration is to satisfy the following two use cases:

- 1. The tester has written a test script in Rapise that tests the user interface of the application (by clicking on buttons and performing other UI tasks) but now wants to load test the server behind the user interface to make sure it can withstand the load of 10,000+ similar users making requests.
 - The existing Rapise test is not suitable because it would mean launching 10,000+ instances of the application or web browser on a single client machine, or having 10,000 instances of the application run on different machines at the same time.
 - The solution is to use a protocol-based load testing tool such as NeoLoad that sends the HTTP/HTTPS requests directly to the server
 - However it is laborious and time consuming to manually record the entire script in NeoLoad from scratch.
 - The solution is to <u>automatically convert the Rapise script to NeoLoad</u> by playing the Rapise script at the same time as NeoLoad records the generated HTTP/HTTPS traffic.
- 2. The tester wants to measure the <u>speed of the user interface</u> that a user would experience whilst performing a protocol-level level load test
 - o How long does each transaction take whilst 1000 VUs are hitting the same system
 - The user interface could be web, mobile or thick client (e.g. ERP system)

2.3.14.1 Convert Functional to Load Test

This aspect of the Rapise-NeoLoad integration describes the process for taking an existing test script written in Rapise and converting it seamlessly into a performance scenario in the NeoLoad load testing system. This feature allows you to convert Rapise tests for HTTP/HTTPS based applications into protocol-based NeoLoad scripts that can be executed by a large number of **virtual users (VUs)** that simulate a load on the application being tested.

Prerequisites

In order to use the integration with NeoLoad, you need to have the following:

- Rapise 5.0 or above
- NeoLoad 5.1 or above

Both tools must be installed on the same Windows host.

NeoLoad must have a license with Recording API enabled. The default trial version does not have this

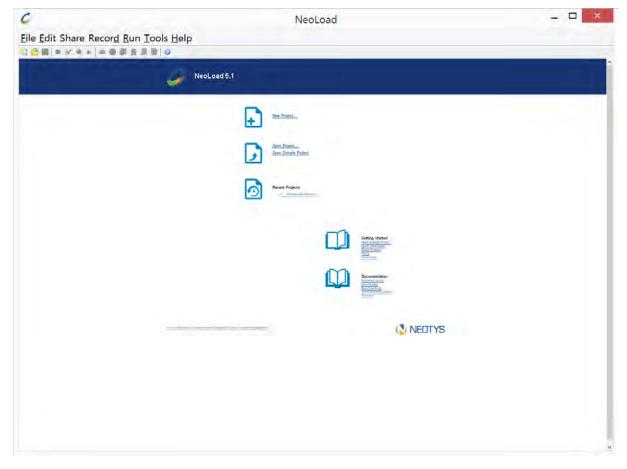
capability.

The Rapise test needs to be testing an application that uses HTTP/HTTPs or other NeoLoad supported protocols.

The Steps for Converting a Rapise Test

Launch NeoLoad

Open NeoLoad via the Start menu or using a command line: "c:\Program Files (x86)\NeoLoad 5.1\bin\NeoLoadGUI.exe"



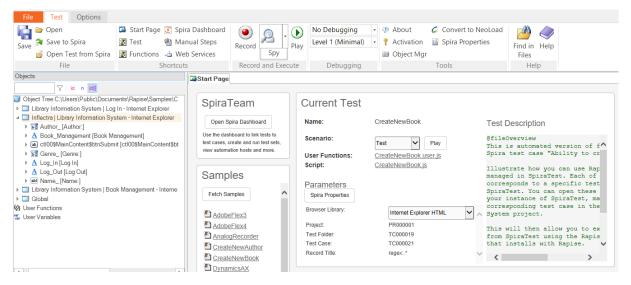
Create or Open Existing Load Project Use menu File > New or File > Open.

C	NeoLoad - [DemoProject]	- 🗆 ×
Eile Edit Share Record E	un Iools Help	
S ≧ ↓ ● ¥ ☆ ► ∞ @ ∅	A A B A	
	of Design 🔤 Runtime 🚿 Results	
Virtual Users @ Population		
Quick search The Servers Wintual User Profiles Chared Containers	Recording Record a Virtual User profile • Start Recording Flags	
	Flag requests:	
	 as variable extractor sued in request whose definition contains: whose recorded response contains: with response validations with variable extractors defined on: requests parameters is link/form extractor with parameters is a POST Form disabled whose type is: Adobe Flex/AMF 	
Actions Delay Choop C While O IfThenElse O Variable Modifier Container Container O TrvCatch Sop Stop K Fork	with SLA Profile: Any	

Open a Test Script in Rapise to be Converted

Inside Rapise, open the script you want to convert. Make sure the application it is testing is a web-based

application using the HTTP/HTTPS protocols:



Ensure that the Internet Explorer HTML library is set in the test parameters.

Sett	ings		
	At I		
۵	Advanced		\mathbf{A}
	CommandLine		
	EntryPoint	Test	
۸	TestParams	5 items	
	Browser	Internet Explorer HTML 🗸	
	Record Title	regex:.*	
	ServerProjectId	5	
	ServerTestCase	39	
	ServerTestCase	113	
۵	Execution		
	CacheObiects	False	Υ.
Ree	wear		
Tes	st Files Settings 0	bject Tree	
_			_

Open Conversion Dialog

On the Rapise <u>Test Ribbon</u> > Tools tab, press the **'Convert to NeoLoad'** button:

🦻 About	🔢 Object Mgr
P Activation	🥔 Convert to NeoLoad
Libraries	📑 Spira Properties
	Tools

This will display the <u>NeoLoad convertor dialog</u>. On this dialog box, set the following parameters:

Convert Current Test to Neo	Load	_		×
Virtual User				
VirtualUser				
Base Container				
Actions				•
Container/Business Transact	tion			
Detect dynamic parameter	S			
	Cano	el	Conv	vert

In this field, you need to enter the name of the virtual user to create in NeoLoad:

- The default value is "VirtualUser"
- If the name is already used, then it is automatically renamed using "_X" suffix, with X an integer incremented.
- If the name has invalid characters then they will be escaped as an underscore (_).

Base Container

This specifies the base container where we want to start the recording (Init / Actions / End)

• The default value is Actions.

Container/Business Transaction

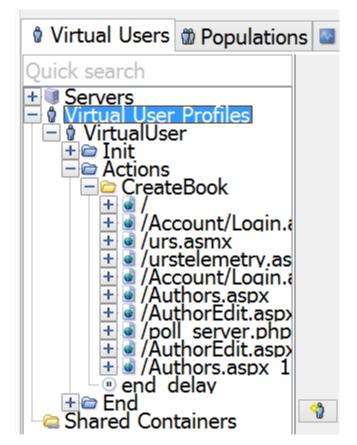
This is used to specify the current recording container in NeoLoad. It is just based on a single level. There is no way to specify a tree of containers.

- The default is no container.
- If the name is already used then it will be made unique by adding _1, _2, etc.
- If the name is empty then no container will be used.

Detect Dynamic Parameters

When you check this box, Rapise tells NeoLoad to scan the protocol traffic to look for known dynamic parameters (e.g. Session IDs, ASP.NET ViewState) that change on each HTTP request and need to be parameterized by NeoLoad to ensure the performance scripts are robust and well-defined (v.s. having a hardcoded Session ID).

In the NeoLoad tree it looks this way:



Press Convert Button

After pressing Convert button Rapise will launch the test and NeoLoad will start capturing network

traffic. When test playback is finished you'll see the dialog:

Ra	pise ×
Convert	ion successfull
	OK

Troubleshooting

During conversion you may get a few error messages. This section provides a list of common messages

and their solution.

Unable to connect to the remote server

	Rapise	×
8	Unable to connect to the remote server No connection could be made because the target machine actively refused it 127.0.0.1:7400	
	ОК	

If you get this message, check that NeoLoad is running. Usually this message means that NeoLoad is not

started or another application is using port 7400, preventing NeoLoad binding to it.

No project is open

Rapise	×
NL-RECORDING-ILLEGAL-STATE-FOR-OPERATION(No project is open.)	
ОК	

If you get this message, check that a project is opened in NeoLoad.

NeoLoad is not ready for recording

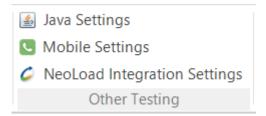
Rapise	×
NeoLoad is not ready for recording. Current status is RECORDING	
ОК	

If you get this message, it means that NeoLoad is in the recording state. To fix the issue, you need to stop the existing recording in NeoLoad.

NeoLoad Integration Settings

There are a few global options for the Rapise - NeoLoad integration. To reach them open up the

Rapise Options ribbon:



In the 'Other Testing' tab, press the "NeoLoad Integration Settings..." button:

NeoLoad Int	tegration Settings $-\Box \times$
2 ∎ 2 ↓ □	
▲ NeoLoad Proxy Address	127.0.0.1:8090
NeoLoad REST Service URL	http://localhost:7400/Recording/v1/Ser
NeoLoad Proxy Address Rapise automatically changes IE pr convertion.	roxy to this value during test script

This is the <u>NeoLoad Settings</u> dialog.

Normally there is no any reason to change these settings, but for completeness they are described below:

- NeoLoad Data Exchange URL this is the URL to the NeoLoad data exchange API
- NeoLoad Proxy Address this is the IP address and prt of the NeoLoad HTTP proxy
- NeoLoad REST Service URL this is the URL to the NeoLoad recording service REST API

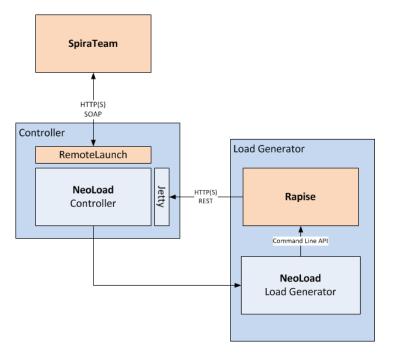
2.3.14.2 Client Performance Monitoring

Concept and Architecture

The purpose of this integration is to satisfy the following use case:

- The tester wants to measure the speed of the user interface that a user would experience whilst performing a protocol-level level load test
 - o How long does each transaction take whilst 1000 VUs are hitting the same system
 - The user interface could be web, mobile or thick client (e.g. ERP system)

The following describes the technical architecture that would need to be in place:



The process flow for the above architecture would be as follows:

- 1. User schedules the performance test in SpiraTest
- 2. RemoteLaunch connected to the NeoLoad controller and SpiraTest initiates the start of the testing
- 3. The NeoLoad controller sends commands to the NeoLoad load generators to start the performance scenario
- 4. During the performance scenario, NeoLoad calls Rapise through its command-line to start a specific test
- 5. At specific points in the Rapise function test (which is grouped into transactions), timing code in the Rapise script (see next section) will call the NeoLoad REST API to report back the timing for the transaction block.
- The NeoLoad controller will need to correlate each of the blocks with the appropriate transaction in the performance scenario
- 6. At the end of the performance test, RemoteLaunch will read the entire performance test and send back the results to SpiraTest.

NeoLoad Global Object

Rapise has a global object named NeoLoad which is capable of sending client performance metrics to the NeoLoad controller.

- 🔺 🛄 Global
 - Android [Android]
 - Database [Database]
 - Global [Global]
 - IOS [iOS]
 - Navigator [Navigator]
 - 🔺 🥥 NeoLoad [NeoLoad]
 - CaptureBrowserPerformance
 - CaptureHostPerformance
 - EapturePerformanceValue
 - CreateClient
 - EndTransaction
 - GetNetworkInterfaceNames
 - StartTransaction
 - Q Ocr [Ocr]
 - Spreadsheet [Spreadsheet]
 - Tester [Tester]
 - WebDriver [WebDriver]

NeoLoad will display the received data in a both tree and graph views.

NeoLoad Tree View

Image: Second state Image: Second state Image: Second state Image: Second state </th
Quick search
C
 easy library TC113 LG Performance A Memory Network Bytes Sent Network Bytes Received Page Load Page Load Inttp://libraryinformationsystem.org/App Themes/DefaultTheme/DefaultTheme.css Duration http://libraryinformationsystem.org/WebResource.axd d=9 D5u0u-RawwI0gM2OcJvZIO2s4 c http://libraryinformationsystem.org/WebResource.axd d=CK2inkCyL7XaDv3zPBZMPWeZRX9f http://libraryinformationsystem.org/Images/SpiraTestLogo.gif http://libraryinformationsystem.org/Images/Inflectra%20Logo-Medium.gif http://libraryinformationsystem.org/Account/Login.aspx http://libraryinformationsystem.org/Account/Login.aspx Transactions T Login T Login T Login Duration Duration Duration Duration

The root of the tree is the name of a script that was executed by Rapise.

On the first level we have three groups:

- LG Performance this is a folder of a load generator essential metrics captured by Rapise: CPU, Memory, Network Sent/Received bytes.
- Page Load this folder contains nodes for each page and nested resource. Duration values are
 obtained using Window.performance structure. See also <u>http://www.w3.org/TR/navigationtiming/#processing-model</u>
- Transactions this folder contains counters for transactions defined by a script.

Graph View

Each leaf node can be dragged to the graph view for visualization:

Graph board	× Graph board_1	2101											
		asy literary TC1134 (3 Pertinmania)	DPU.			1000			wasy title	wyYottatus Performance	Memory		
		V-V-		M	-	10,000 7,500 5,000 2,500 8	00.03.16		0.00	2019/28	10 00 25	0000	
0.0000	010100 000 00	European street	00 00000 (0004	10 00 04 M	Swite of		bullet, re	~	DO BO YS	CLASSE C COM	10.07.25	90.02	A.
	wasy like	ey YC11310 Performance/Network	Bytes Dard						easy iterary TC1	OIS O Performance Network	Bytes Received		
100 100 200						3.000 2.900 1.900	/	/					
20	a3.10 00.03.14	D0.09.20 Elapsed time	10 01 21		00.09.36	-	(0.83 ×1		23-18	D) 05.20 Explaint time	20.03.25	00.0000	810
	#1472	itrary TCA13/Transactional? Home C	Durgelien.						sasythra	yTC113/Transactions/T Log	ND # MIN		
100						1.800							
1.000						0.000							
2.000						2,800							
00.00030	0010100 0001100	ob da ba Bi gana di ba	00.0000 00.000	00040)	0010400	5 0	00100 001010	0 000100	00157:00	(oratino or Especifica	000000000000000000000000000000000000000	10.0=01 0	olokato (olda
	easylibrary 10143	Page Loaditito (Etraryinformation	system.org/Duration				any library TC113/Page	Loadhtp /librarym	for mations yeters on	jihttp://bitraryinformationsy	ten orgilizo, The mesilitets	ATThe me Default The me us	n Duration
1290 1006 1100 1007 100 1000 1000 1000 1000						18 18 18 05	/						
0 00 00 00	00/01/00 00:00 30	Elapsed time	ộc ộa bở 00 đã độ	00:04:00	0010430	0.5	30.0230 10	06103 30	010 (010	12/00 00.02/30 Ellapsed time	010100 000	2.00 001.04.00	30,0+3\$
	eary	brary TC 113/Transactions/T Logout/	Duration						wany library TC11	Wiransactiona/T Create Nev	AutoriOurstion		
(400						10.000							
1000						5.000							
500						2.000							
	ci - colarcial - colocciali	do do po El ameditime	(0.000 DC 60 00	0033430	00.06.00	00000		00.01.00	00-02-00	nó czas co co n	ob chore	000+00 000+04	co co co

Performance Filters

Performance results obtained from different runs can be filtered using result filter on the toolbar.

Filters: Ø All Ø All Ø All All ...

You can open the Results Filter dialogs by pressing '...' button.

$\boldsymbol{\boldsymbol{\omega}}$	Result Filter	×
Result Filter		
Select the te	st result filters to apply.	
Platform	All	~
Client Software	All	~
Location	All	~
Script	All	~
	Close	

- Platform is a combination of hardware OS. Hardware is passed by a user script, OS is determined by Rapise automatically. E.g. hardware1 Microsoft Windows 8.1 Pro.
- Client Software is Rapise.
- Location is supplied by a script. E.g. location1.
- Script is a combination of script name and time stamp. E.g.

Script	All	~
	🥃 easy library TC113_1460648451876	~
	<i>a</i> easy library TC113_1460648364186	
	aeasy library TC113 1460648276751	

NeoLoad Object API Reference

The methods of the global NeoLoad object are described in **object reference section** of this help file.

0	Rapis	e User Manual - HtmlHelp - Viewer 📃 📃	×		
<u>File View</u>					
🗢 🔿 💽 🔂 🗇 🤣 💈) 🔊				
Contents Index Search	NeoLoad	Top Previous Net	1		
Company & Copyright Company	Global object. Use it to perfi object, it is always automat	rm actions not related to a particular object. You do not need to record or learn this cally available in any test.			
E Global	Action Summary		I		
🖹 Navigator	Action	Description	il		
E NeoLoad	CaptureBrowserPerformanc	Captures duration of loading a page and resources.	11		
SeSReport Exceptio	CaptureHostPerformance	Sends host performance metrics to NeoLoad: CPU, Memory, Network, Battery.			
SeSReportImage	CapturePerformanceValue	Sends performance value to NeoLoad.	1		
SeSReportLink	CreateClient	Connects to NeoLoad.			
🖹 Spreadsheet	EndTransaction	Marks end of a transaction and sends performance metrics to NeoLoad.			
Tester	GetNetworkInterfaceNames	Gets names of network interfaces.			
E VebUnver	StartTransaction	Marks start of a transaction.			
🗄 🍫 HTML	Action Detail				
	CaptureBrowserPerfe Captures duration of loading Parameters:				
	Name Ty	pe Description	ť		
	uri	Node name for NeoLoad entry tree.			
< >	CaptureHostPerforma Sends host performance me	ince() Irrics to NeoLoad: CPU, Memory, Network, Battery.	•		

How to Use

In order to send performance metrics to NeoLoad you need to add a sequence of calls to your script.

First, connect to the NeoLoad. When this call is executed NeoLoad must be running a script, otherwise an error will be returned.

NeoLoad.CreateClient("hardwarel", "location1", "Broadcom 802.11ac Network

```
Adapter _2");
```

Let's look at the following block:

```
NeoLoad.CaptureHostPerformance();
NeoLoad.StartTransaction('T Home');
NeoLoad.CaptureBrowserPerformance(SeS('Log_In').GetPageURL());
//Click on Log In
SeS('Log_In').DoClick();
NeoLoad.CaptureBrowserPerformance(SeS('Username_').GetPageURL());
NeoLoad.EndTransaction('T Home');
```

- CaptureHostPerformance call sends immediate values of CPU, Memory and Network load to the NeoLoad controller.
- StartTransaction/EndTransaction calls define a transaction with name 'T Home'. Upon EndTransaction call the duration of it is sent to the NeoLoad controller.
- CalculateBrowserPerformance call sends page load performance metrics to the NeoLoad controller.

There is also a general call that can send any user-defined metric to the NeoLoad controller:

```
NeoLoad.CapturePerformanceValue("First Level Folder", "Transaction1|
Duration", "milliseconds", 120);
```

Components of the second parameters can be separated by |. It allows to add more levels to the tree.

2.4 Dialogs, Views, and Menus

This section details the Rapise GUI. Each subsection describes the function of a particular Dialog, View, or Menu. The purpose and consequences of all buttons, options, lists, and check boxes are listed.

2.4.1 Add Web Service Dialog

Screenshot		
Add New Web Service	2	\times
	This wizard will create a new web service definition file inside the current test:	
●REST ○SOAP		
LibraryInformationSystem	rest	
C:\Users\adam.sandman\	Documents\My Rapise Tests\MyRestTest1	
	Create Cancel	

Purpose

Adds a new REST or SOAP web service to your Rapise test. It adds the web service as a .rest or .soap file that is added to the "Services" folder of the "Test Files" section:

How to Open

Click on the "Web Services" icon in the Rapise Test ribbon tab.

2.4.2 Browser Settings Dialog

Purpose

This dialog box displays the list of native (not using <u>Selenium</u>) web browsers that have been configured for use by Rapise and lets you create a new profile, modify a profile or make a new profile based on an existing one.

This feature is useful if you have several different versions of a web browser on the same machine (e.g. Firefox Latest vs. Firefox ESR, or different versions of Chrome portable) or if you want to be able to run a browser with different command-line options (e.g. IE normal, IE safe mode, IE emulating a specific version)

Screenshot

owser Settings		
Select Profile: Firefox HTML	Save Test Duplicate Create Rename	Delete
::::::::::::::::::::::::::::::::::::::		
4	a second s	
Browser Library	Firefox HTML	
Browser Path		
CMD Line Parameters		
Default Click Mode	auto	
Browser Library		
Type of browser.		
	r	Close
		Close

How to Open

You can open this dialog box from the main Rapise Options ribbon.

Menu Options

This dialog box has the following menu options:

- Select Profile This dropdown list lets you select a different web browser profile to be displayed in the dialog.
- Test This button will test the connection from Rapise to the specified browser
- Duplicate This button will create a new browser profile based on the currently viewed one.
- Create This button will create a new empty browser profile that you can edit.
- Rename This button will change the name of the current browser profile being edited.
- Delete This button will delete the currently displayed browser profile. There is no undo, so be careful!

Profile Options

This section has various settings, each of which are described below:

- Browser Library The type of browser being used, currently can be:
 - o Internet Explorer HTML
 - Chrome HTML
 - Firefox HTML
- **Browser Path** The path to the location of the web browser executable (e.g. chrome.exe, iexplore.exe, firefox.exe) on the computer
- CMD Line Parameters Any command-line parameters to pass to the web browser (e.g. -extoff for IE safe mode)
- Default Click Mode Species the default 'click mode' for tests using this web browser profile

 auto This tries to locate an element on the screen, moves the mouse over it and then sends the
 appropriate DOM 'click' event (the default)
 - ${\scriptstyle \circ}$ event Just sends the DOM click event to the element with no prior mouse-move
 - $\,\circ\,$ click Simulates an actual click on the element rather than sending a DOM click event

2.4.3 Create New Test Dialog

Purpose

Create a new Rapise test. You have the option of either connecting to <u>Spira</u> and storing the new test in our central test management system or simply saving the new test locally.

How to Open

Simply click on the File tab of Rapise and click New Test (Create New Test) on the File menu.

(a) Creating in Spira

By default Rapise will ask you to save the new test into the Spira test management system:

Create or Open Test				
Server:	http://doctor/spi	irateam		
Project:				-15
				1
Test Cases:			New Folder	New Test Case
Test Steps				
Test Steps Step Descri	otion	Expected Result	Actual F	Result
Step Descri	Specif	Expected Result	Actual F	Result
	Specif		Actual F	Result
Step Descri	Specif	fy methodology:		

Assuming that you have already <u>configured the connection to Spira</u>, first you need to select the project in Spira. That will then display the test case folders and test cases in Spira:

	en Test			
Server:	http:/	//doctor/spirateam		
Project:				
Library Inform	ation System			
Test Cases:			New Folder	New Test Case
· · · · · · · · · · · · · · · · · · ·	ility to create new book ility to edit existing book ility to create new author ility to edit existing author			E
Ab Beare	ility to reassign book to diffe ssion Tests	erent author		-
Ab Beare		erent author Expected Result	Actual R	esult
Test Steps Step	ssinn Tests		Actual R	
Test Steps Step	Description	Expected Result		

If there is already a test case in Spira that has <u>not already been linked to Rapise</u> then you can simply select that test case, which will display any existing manual test steps that exist:

Server		http://doctor/spin	alean			
Projec						-
Libra	ary Informat	ion System				1
Test C	ases:		New F	older	New Test Cas	se
		y to edit existing book y to create new author				
Test S	다 Abilit 전 Abilit Regressi	y to edit existing author y to reassign book to different author inn Tests				+
	다 Abilit 전 Abilit Regressi	y to edit existing author y to reassign book to different author	Expected Result	Actual Res	ult	1 1 A
	Abilit Bearessi Steps	y to edit existing author y to reassign book to different author ion Tests		Actual Res	ult	-
	Abilit Bearessi Steps	y to edit existing author y to reassign book to different author on Tests Description		Actual Res	ult	· · · ·
	Abilit Regressi Steps Step 1	y to edit existing author y to reassign book to different author on Tests Description User opens up Internet Explorer a	. The browser loads the login w	Actual Res	ult ,	
	Abilit Regressi Steps Step 1	y to edit existing author y to reassign book to different author on Tests Description User opens up Internet Explorer a User logs in to application	The browser loads the login w User taken to main menu scre	Actual Res	ut	· · · ·
Test S	Abilit Renressi Steps 1 2	y to edit existing author y to reassign book to different author on Tests Description User opens up Internet Explorer a User logs in to application	The browser loads the login w User taken to main menu scre	Actual Res	ult	

If this is the test case you want to associate the new Rapise test with, then simply click **Create from Spira**.

If you want to create a new test case in Spira to use, simply click New Test Case:

Create New Test Case	
Test Case name:	
	OK Cancel

Then enter the name of the new test case and click **OK**. Once it has been created you can then select it in the test case list and click **Create from Spira**.

Sometimes there is no existing folder inside Spira that makes sense to use. In which case you can first use the **New Folder** button to create an empty folder that new test cases can be created in:

Create New Spira Folder	
Folder name:	
	OK Cancel

Regardless of which option you choose, before you click **Create from Spira**, you have the choice of test methodology to use.

Currently there are four methodologies available in Rapise:

- **Basic: Standard Scripting Mode** this should be used for testing desktop applications or any testing not involving web browsers or mobile devices.
- Manual: Manual-only Testing this should be selected if you are only going to be creating or executing manual tests
- Mobile: Mobile Support this should be selected if you are going to be testing apps running on mobile devices
- Web: Cross Browser Testing Support this should be selected if you are going to be testing web applications running in a web browser.

If you do not plan on using Spira for managing your test scripts (or you are not able to connect when you want to create the test), you can click on the **Create Locally...** to just create the test case locally (see next section). You can always save to Spira later on.

Once you have created the test, Rapise will ask you to <u>choose the Scripting language</u> (<u>RVL</u> or <u>JavaScript</u>).

(b) Creating Locally

If you choose the option to **Create Locally** the following dialog box is displayed:

🗮 Create New Test	×
Please specify the name of the new test:	
Specify the folder where this test should be created: C:\Users\adam.sandman\Documents\My Rapise Tests	
C:\Users\adam.sandman\Documents\My Rapise Tests\My Rapise Tests.sste	est
Rapise supports multiple testing methodologies. Select one to be used for newly created	ated test.
Basic: Windows Desktop Application (Windows Native, .NET, Java, WPF etc)	
Manual: Manual-only testing	
Mobile: Mobile Support	
Web: Cross-Browser Testing Support	
	Cancel

You need to enter the following information and click Create:

- The name of the new test please enter the name of the new test that you wish to create.
- Folder please choose the folder on your local computer that you wish to store the Rapise test in.
- Specify methodology there are currently four methodologies available in Rapise:
 - **Basic: Standard Scripting Mode** this should be used for testing desktop applications or any testing not involving web browsers or mobile devices.
 - Manual: Manual-only Testing this should be selected if you are only going to be creating or executing manual tests
 - Mobile: Mobile Support this should be selected if you are going to be testing apps running on mobile devices
 - Web: Cross Browser Testing Support this should be selected if you are going to be testing web applications running in a web browser.

Once you click **Create**, Rapise will ask you to <u>choose the Scripting language</u> (<u>RVL</u> or <u>JavaScript</u>). Once you have chosen the scripting language, the new test will be created and saved locally.

2.4.4 Create Sub-Test Dialog

& Create Sub-Test	X
You are about to create another Test inside the folder of current Test. Please, provide Test folder name:	
C:\Users\Public\Documents\Rapise\Samples\SampleATM	
Use Methodology:	
Basic: Manual Scripting Mode	-
Rapise supports multiple testing methodologies. Select one to be used for newly created Test.	
New Test should have own set of Objects.	
If unchecked the test will re-use objects from containing Test. I.e. it will not have own file " <testname>.objects.js" and will refer one from its parent.</testname>	
New test should have own User-defined functions	
If unchecked the test will re-use functions from containing Test. I.e. it will not have own file " <testname>.user.js" and will refer one from its parent.</testname>	
Create Can	cel

Create a sub-test.

- New test should have own set of Objects: Uncheck it if you want to create a scenario re-using objects from parent test.
- New test should have own User-defined functions: Uncheck it if you want to create a scenario re-using utility functions from its parent test.

The Sub-Test is always created inside the folder of its parent test. If parent test is saved to a new location then sub-test is also saved as a sub-folder of a new location.

How to Open

Choose Create Sub-Test... in the context menu of a folder in <u>Test Files</u> dialog.

2.4.5 Content View

File Test Options	Edit				
Dpen 🗁 Open	📮 Start Page 🥃 Spira Dashboard	(a)	No Debugging 🔹	About Convert to NeoLoad	pi 🍋
Save to Spira	🕃 Test 🛛 🏨 Manual Steps	Record Play	Level 1 (Minimal) 🔹	💡 Activation 🛛 📑 Spira Properties	Find in Help
💣 Open Test from Spira	🕃 Functions 👍 Web Services	Spy		🔛 Object Mgr	Files
File	Shortcuts	Record and Execute	Debugging	Tools	Help
Files Application A Data Reports Scripts SampleATM js SampleATM user j SampleATM user j	Start Page SampleATMj function Test() 13 C (14 C > /** 15 > * @fileOverview 16 > * This example s 17 > * script as a se 18 > * parts, such as 19 > ** 20 > * The AUT being 21 > * terminal. It s	shows how you can at of functions. T s DoLogin, DoDepos used in this test reads data from AT iliar with SampleA ject tree.	test an MFC appl he test was reco it, etc. is SampleATM. T MDatabase mdb da	ication and organize your rded and then split into This is a version of ATM	пер

To view and edit files. This includes the following file types:

- JavaScript (.js) automated test script files
- Report (.trp) files that open in the <u>Report Viewer</u>.
- Excel (.xls) files that can be displayed in the Spreadsheet Editor
- REST (.rest) web service definition files that open in the <u>REST Editor</u>.
- SOAP (.soap) web service definition files that open in the SOAP Editor
- Analog Recording Files (.arf) that contain analog testing mouse clicks and coordinates.
- Manual test steps (.rmt) that open in the Manual Test Editor.

How to Open

Open a file using the <u>Test Files Dialog</u>. The file will open inside of the Content View.

2.4.6 Enter filter criteria for... Dialog

Filter based on	All	 of the following conditions: 		
Add	Status		- ((DBNull))	z
Delete				

Allow more than one <u>filter criteria</u> for the same column.

How to Open

In the <u>Report Viewer</u>, open the drop-down menu for one of the <u>filter cells</u>; select the **Custom** option (see below):

nent		Status		Ite	
			Ø	=	
	(Custor	(Custom)			
	(Blanks)			0	
	(NonBlanks			0	
ont	Fail			0	
	Pass			0	

Conditions

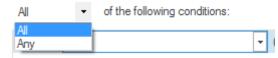
All	 of the following conditions: 		
Status		 ((DBNull)) 	-
	= Equals≠ Does not equal		
	< Less than < Less than or equal to > Greater than	<u><u>o</u>k</u>	<u>C</u> ancel
	≥ Greater than or equal to ∗ Like ⊷ Matches Regular Expression	-	

You may specify as many conditions as you like. Each condition has two properties, a Matching

Criteria on the left and a **filter value** on the right. The **filter value** is a string, and the **matching criteria** specifies what constitutes a match. For more details, look <u>HERE</u>.

Filter Aggregation

There are two ways you can aggregate / combine filter conditions:



- All: All conditions must be true to constitute a match.
- Any: At least one condition must be true to constitute a match.

Buttons			
Add			
Delete			

- Add: Add a extra condition row.
- **Delete**: Delete the selected condition. You can select a condition by clicking on the field name to the left of the matching criteria:

Status	•	((DBNull))	•	
Status	-	((DBNull))	Ŧ	1

- OK: Close the dialog and apply the filter.
- Cancel: Close the dialog. Do not apply the filter.

2.4.7 Errors View

Errors		ų
I	2000 B	
T:\Admin\Wrappers\Engine\P	ayer.js(407, 4) Microsoft JScript runtime error: Exception thrown and not caught	2

Purpose

The **Errors View** displays execution error details. Execution errors are those that cause <u>Recording</u> or <u>Playback</u> to stop.

How to Open

The Errors View is part of the Default Layout.

Error Message

T:\Admin\Wrappers\Engine\Player.js(407, 4) Microsoft JScript runtime error: Exception thrown and not caught

Double click on an error message to go to the corresponding source line.

Widgets

- The text box is a search box.
- The icons from left to right are Find Next Entry 2, Copy Selected 1, Clear All Text , and Select All Text .

2.4.8 Find and Replace Dialog

Screenshot	
Find and Replace	×
🥙 Find 🏠 Replace	
Eind what:	
Search	-
Look in: Entire Test And Subtests	
Find options	
Match case Include sub-folders	s
Match whole word Include hidden Look at these file types: * js	Ţ
	-
	-
Find Next Find All	
	_1

Purpose

To find and replace text in files displayed in the Rapise Content View.

How to Open

Select the Find in Files button on the Ribbon (Test tab > Tools menu).

Find in Files Tab

- Find what: Place the string you would like to search for in the Find what text box.
- Look In: this option specifies where the search will take place. You can limit the search to: current document, current selection, current test, the entire test and subtests, or a specific folder.
- Directory path: Use the Directory Path text-box to specify the directory in which to search. The Directory path text-box cannot be accessed (and is ignored) if the Test files checkbox is checked.

- Check the Include sub-folders option to search recursively from the directory specified in the Directory Path text-box. The Include sub-folders option cannot be accessed if the Test files checkbox is checked.
- Match case option: If unselected, case is ignored in the search.
- Match whole word option: If set to true, parts of words will not count as matches.
- Look at these file types: Search only files with the specified file type(s).

Find and Replace Tab

There is only one significant difference between the Find in Files Tab and Find and Replace Tab: the **Replace with** text-box.

• Replace with text-box: All occurrences of the string in the Find what text-box will be replaced with the string in the Replace with text-box when you press the Replace button.

2.4.9 Find Results View

Screenshot		
Find Results		
	a 🔍 🕄 💊	
C:\Users\Public\Documents\Rapise C:\Users\Public\Documents\Rapise C:\Users\Public\Documents\Rapise	Samples Adobe Flex Adobe Flex js(87,1): Samples Adobe Flex Adobe Flex js(91,1): Samples Adobe Flex Adobe Flex js(95,1): Samples Adobe Flex Adobe Flex js(101,1): Samples Adobe Flex Adobe Flex user js(38, Samples Adobe Flex Adobe Flex user js(38, 38)	

Purpose

Displays results for the Find and Replace Dialog.

How to Open

The Find Results view is part of the Default Layout.

Messages

C:\Users\Public\Documents\Rapise\Samples\AdobeFlex\AdobeFlex.js(101,1): Tester.AssertEqual("Verify that: Visible=true Double click on a message to go to the corresponding source line.

2) 0)

Widgets

- The text box is a search box.
- The icons from left to right are Find Next Entry 🧖, Copy Selected 🤍, Clear All Text 🚫, and



2.4.10 Find Text dialog

0	Screenshot
[Find and Replace
	🌁 Find 🏫 Replace
	Find what:
	Search 🗸
	Look in: Current Document
	Find options
	Match case Include sub-folders
	Match whole word Include hidden
	Look at these file types: •.js
l	Use: Standard Search 🔹
	Find <u>N</u> ext Find <u>A</u> II

Purpose

Find occurrences of the Search Term text in the currently visible Source Editor.

How to Open

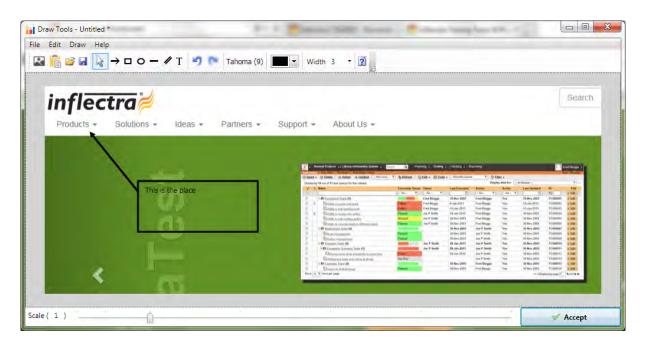
Ribbon > Edit Tab > Search menu > Find button

Or type **CTRL+F** on the keyboard when the source editor is option.

Find Tab

- Find what: Place the string you would like to search for in the Find what text box.
- Look In: this option specifies where the search will take place. You can limit the search to: current document, current selection, current test, the entire test and subtests, or a specific folder.
- Match case option: If unselected, case is ignored in the search.
- Match whole word option: If set to true, parts of words will not count as matches.

2.4.11 Image Capture



The **Drawing Tools** image editor lets you capture a section of the current screen or application under test, add annotations to help document the image and then attach the final result to the <u>current test</u> <u>case</u>, test step, or <u>manual test result</u>.

How to Open

You can open the Drawing Tools dialog box by clicking on the **Image icon** on the various rich text editors in Rapise. When you do that, Rapise will minimize itself and display the following screen:



You now need to draw a rectangle on your screen that tells Rapise which part of the screen you want to capture. Once that is done, the image editor will open with that part of the screen selected. If you click ESC on the keyboard, it will just open the editor with no initial image.

Image Editor Toolbar

The image editor provides the following tools:



- Image Capture this lets you discard the current image and capture a new screenshot instead
- Paste From Clipboard this lets you paste in an image from the Windows clipboard
- Open this lets you open an existing image saved on your local computer

- Save this lets you save the current image to your local computer
- Pointer this lets you select an annotation to edit (arrow, rectangle, ellipse, line, text, etc.)
- Arrow this lets you draw an arrow in the current color on top of the current image
- Rectangle this lets you draw square / rectangle in the current color on top of the current image
- Ellipse this lets you draw a circle / ellipse in the current color on top of the current image
- Line this lets you draw a straight line in the current color on top of the current image
- Pencil this lets you draw freehand in the current color on top of the current image
- Text this lets you add text in the current color and current font on top of the current image. You will need to draw a rectangle to mark the size of the text box before entering in the text.
- Undo this will undo the last operation
- Redo this will redo the last operation
- Font Name this will let you change the font family and size:

Tahoma Normal Normal Normal 9 Sylfaen Normal Normal Normal 8 9 Times New Roman Oblique Bold 10 11 Vrinda Shonar Bangla Microsoft Yi Baiti 16 18 Tahoma ABbCc XxYzZz 012345 ABbCc XxYzZz 012345 10 11	Font Family	Style		Weight	Stretch	Size	
Microsoft Tai Le Times New Roman Tunga Vrinda Shonar Bangla Microsoft Yi Baiti Tahoma Tahoma Microsoft Yi Baiti	Tahoma	Normal		Normal	Normal	9	
	Microsoft Tai Le Times New Roman Tunga Vrinda Shonar Bangla Microsoft Yi Baiti				Normal	10 11 12 14	
HADDUC AX1222 012345			-Phos W	V-7- 012245			
			abbee AA	1222 012345			

• Color - This lets you change the current color (used in the various annotations):



• Line Width - This lets you change the current line width (used in the various annotations)

Image Editor Footer

The footer of the Drawing Tools provides the following options:



- Scale this changes the zoom of the current window, allowing you to more easily view small/large images
- Accept this accepts the current image and inserts it into the test case, test step or test run that was being edited.

2.4.12 Incident Logging

New Incident *	- X	Louis Brancis and		
🛃 Save				
🔊 Details / Descri	iption			
Name: This is a n	new incident			
Description:				
BIUAA	rial 🔹 🛧 😻 🛽			
+				
Туре:		•		
Type: Detected By:		Owned By:	None	•
Detected By:		Owned By:	None	•
Detected By:	Fred Bloggs None	Owned By:	None	
Detected By: Priority:	Fred Bloggs None None	Owned By: Severity:	None	•

The **New Incident** logging dialog box lets you log a new incident (also known as a bug or defect) into a connected <u>SpiraTest</u> instance. If you logged the new incident during a <u>manual test execution</u>, it will be linked to the current test run.

How to Open

You can open the **New Incident** dialog box by either clicking 'New Incident' in the <u>Manual Ribbon</u>, or by clicking the 'Log Incident' button on the <u>Manual Playback</u> dialog box.

Details / Description

Name: This is a r	new incident					
Description:	iew meident.					
BIUAA	rial • A ^a 10 •	A* ===				
I was trying to create	a new book in the system and it displayed					
	(
Туре:	Bug		•			
Type: Detected By:			•	Owned By:	None	
Detected By:					None	
Detected By:	Fred Bloggs None		•			

The **Details/Description** section lets you enter the short name and long description of the new incident as well as the following fields:

- **Type** the type of the incident (e.g. bug)
- Detected By who found the bug (typically your user)
- Priority how important the bug is
- Severity how critical the bug is

Commonte

- Owned By who the bug should be assigned to (or left unassigned)
- Detected Release which version of the system was the bug found in
- Resolved Release which version of the system should the bug be fixed in
- Verified Release which version of the system was the bug retested in
- Custom Fields in addition any custom fields created in your Spira instance will be displayed

	s omment:			
ΙU	A Arial	• A ^a 10 •	1.	
1	A Arial	• Aa10 •	B	

The **Comments** section lets you enter a comment that will be logged with the new incident. The field is a rich text field that can contain formatted text.

Schedule)				
🔿 Schedule					
Start Date:	Select a date	15	End Date:	Select a date	15
Percent Complete:		0 %	Estimated Effort:	hours minutes	
Projected Effort: 0	hours, 0 minutes		Actual Effort:	hours	
			Remaining Effort:	hours minutes	

The Schedule section lets you enter in schedule/effort related information for the new incident:

- Start Date This is the planned start date of the new incident
- End Date This is the planned completion date of the new incident
- Estimated Effort This is the number of hours the incident is expected to take
- Actual Effort This is the number of hours that were actually expended
- Remaining Effort This is the number of hours remaining to fix the incident

In addition, the following calculated fields will be displayed:

- Percent Complete This is the measure of much of the incident has been completed. It is calculated from 100% (Remaining Effort / Estimated Effort)
- **Projected Effort** This the current measure of how long the incident is expected to take based on current information. It is calculated from (Actual Effort + Remaining Effort)

Attachments

Attachments

Filename Version Author Date Created Size

This section displays the list of attachments associated with the new incident. Since Rapise already has a <u>screenshot capture</u> utility built-in, this section is typically not used.

2.4.13 Manual Playback

Screenshot

Executing Ability to create new book		
Ability to create new book		>
Tests that the user can create a new book in the system [E] B / U A Arial Arial A arial A a 10 A A A A A A A A A A A A A A A A A A		
Expected Result: The browser loads the login web page	Sample Data: http://www.libraryinformationsystem.com/beta	
Actual Result- Please enter the actual test result if it differs, then click the app	ropriate button:	
Im B I Im Im <t< td=""><td>50-</td><td>l</td></t<>	50-	l

Purpose

The **Manual Playback** dialog box lets you execute a series of manual test cases (including those part of a test set) from within Rapise. The results from the manual test result will be reported back into your connected <u>Spira</u> instance. During the executing of the manual test, you can attach screeenshots, files and log incidents related to the test result

How to Open

You can open the **Manual Playback** dialog box by either clicking '**Execute Manual'** icon in the <u>Manual</u> <u>Ribbon</u>.

Test Case Details & Test Step Selector

Executing Ability to create new book	
Ability to create new book	آ
Tests that the user can create a new book in the system	
🖾 B Z U A Arial 🗸 🗛 10 🗸 💁 🖉 🖉 🚍 🚍 🖂 🖂 🖂	
Step 1 Step 2 Step 3 Step 4 Step 5 Step 6	
Pass V Pass All V Blocked O Caution I Fail X	
Step 1 of 6 - Please follow the directions outlined in the box below:	

The top part of the manual playback screen lets you view the name and description of the test case, **navigate between the test steps** and click one of the result buttons to indicate how the application being tested behaved:

- Pass The current test step was completed successfully and the expected result was observed
- Pass All All of the steps in the test case could be completed successfully and the expected results were observed in all steps
- **Blocked** The current test step could not be performed because something else prevented its completion
- Caution The current test step could be performed but the actual result only partially matched the expected result (there were minor differences)
- Fail Either the current test step could not be performed successfully or the observed actual result did not match the expected result

Test Step Expected & Actual Result

Executing Ability to create new book	14	
Ability to create new book		>
Pass V Pass All V Blocked O Caution Fail X		
Step 1 of 6 - Please follow the directions outlined in the box below:		
User opens up Internet Explorer and enters application URL: http://www.libraryin	formationsystem.com/beta	
Expected Result:	Sample Data:	
The browser loads the login web page	http://www.libraryinformationsystem.com/beta	
Actual Result- Please enter the actual test result if it differs, then click the appropriate but	tton:	
SSSSS		
🖼 🖪 I U A Arial 🔻 🗛 10 👻 🗛 😻 🖉 🖉 🚍 🚍 🚍 📾		La
Add Attachment		

This section displays the details of the current test step and lets you enter in the observed actual result:

- **Description** This displays the description of the action that the tester should carry out on the application being tested.
- Expected Result This contains a description of the expected result if the application performs as expected
- Sample Data This (optional) field contains any sample data that should be used during testing

• Actual Result - This is a formatted text box where the tester should enter in what actually happened during testing. It is required if you Fail, Block or Caution the test step, but is optional for steps that Pass.

In addition, you can click on the picture icon to <u>add a screenshot</u>, or use one of the two buttons underneath:

- Add Attachment this lets you choose a file from your local system and attach to the test result.
- Log Incident this lets you log a bug/incident that is connected to the test step (e.g. if it failed) and will display the <u>New Incident</u> dialog box.

Minimized Playback Dialog

Executing Ability to create new book		
 Ability to create new book 		>
Pass V Pass All Blocked O Caution	Fail X	
Step 1 of 6 - Please follow the directions out	lined in the box below:	
User opens up Internet Explorer and enter		
www.libraryinformationsystem.com/beta	s application one. http://	
www.nordrynnormationoyatern.com/beta		
Expected Result:	Sample Data:	
Expected Result: The browser loads the login web page	Sample Data: http://	*
	http://	*
The browser loads the login web page	http://	*
The browser loads the login web page Actual Result- Please enter the actual test resu SSSSS	http://	*
The browser loads the login web page Actual Result- Please enter the actual test resu	http://	* *

Sometimes you want to be able to reduce the amount of space taken up by the testing dialog box so that you can view the application and the test steps on the same screen at the same time. To make this easier, if you click on the Minimize (>|) icon in the top-right of the dialog box it will change the manual playback dialog to the mini version show above. You can click on the icon again to switch back to the standard player.

2.4.14 Manual Test Editor

	Manual Steps.mt**		.
 Ability to create 			
	user can create a new book in the system		
BIU	A Arial ▼ A ^a 10 ▼ A ▼ ♥ ▼ ■ ■ ■ !Ξ }Ξ ∞		
StepId	Description	Expected Result	Sample Data
Step 1 [TS:1]	Call		
1 Step 2 [TS:2]	User clicks link to create book	User taken to first screen in wizard	
1 Step 3 [TS:3]	User enters books name and author, then clicks Next	User taken to next screen in wizard	Macbeth, William Shakespeare
1 Step 4 [TS:4]	User chooses book's genre and sub-genre from list	User sees screen displaying all entered information	Play, Tragedy
1 Step 5 [TS:5]	User clicks submit button	Confirmation screen is displayed	
🐠 Step 4	@MyEunction():		

The **Manual Test Editor** lets you create and edit manual test cases that are stored in <u>Spira</u>. These manual test cases contain a high level description of the test case as well a detailed set of steps and associated expected results that make up the manual test script. These manual tests can be <u>executed</u> <u>manually</u> in Rapise (or in Spira) as well as used as the basis for creating a related automated test script. Such automated test scripts may be linked to individual test steps by means of the <u>test scenario</u> approach.

How to Open

You can open the **Manual** ribbon by either clicking on the **Manual Steps** icon on the main <u>Test ribbon</u> or clicking on the **ManualSteps.rmt** file in the <u>Test Files</u> tab. The <u>Manual Ribbon</u> will be displayed whenever you have the Manual Test Editor open.

Test Case Name/Description

A Abili	ty to create new book
Tests th	nat the user can create a new book in the system
Precon	nditions
	ser is already logged in ne book does not already exist
Je B	I U A Arial ▼ A210 ▼ ▲ ♥ ▼ ■ ■ ■ Ξ Ξ Ξ ∞

This section lets you edit the name and long formatted description of the test case. The rich text editor lets you choose the font name, font size, text color, highlight color, style (bold, underline, italic) as well as provides easy ability to add links, bullets and numbered lists.

In addition there is a button that lets you<u>add screenshots</u>.

Test Step Editor

StepId	Description	Expected Result	Sample Data
Step 1	Call		
[TS:1]			
🔮 Step 2	User clicks link to create book	User taken to first screen in wizard	
[TS:2]			
谢 Step 3	User enters books name and author, then clicks Next	User taken to next screen in wizard	Macbeth, William Shakespeare
[TS:3]			
N Step 4	User chooses book's genre and sub-genre from list	User sees screen displaving all entered	Plav. Tracedv

This section lets you add, edit and delete test steps from the manual test case. Each of the test steps contains four fields:

- Step ID this contains the position number of the test step (e.g. step 1) as well as the ID of the test step as it exists in Spira. If you click on the [TS:xxx] label it will automatically copy this into the Windows clipboard. This allows you to easily paste the ID of the test step into your automated test scripts which allows Rapise to report back test results to Spira against specific test steps.
- **Description** this is a description of the test procedure that the tester should perform
- Expected Result This is a description of the expected result that should be observed if the system being tested performs correctly
- Sample Data This is an optional field that contains any sample data that should be used in the test

Each of the fields provides a rich text editor lets you choose the font name, font size, text color, highlight color, style (bold, underline, italic) as well as provides easy ability to add links, bullets and numbered lists. In addition there is a button that lets you add screenshots to the test step.

For ease of editing, you can navigate between the rows and columns using the **ALT + Arrow keys** on the keyboard.

Automating Test Steps

Sometimes you have a primarily manual test case that you want to automate certain steps of. For example you may want to automate the setup of the test data or login to the application before carrying out manual testing. Such a test is called a <u>semi-manual test</u>.

To do this, you enter the syntax @FunctionName(); in the Description box of the test step. Then when you run the test, that step will be executed automatically. The @FunctionName(); refers to a JavaScript user function called function FunctionName() in the Test.user.js file.

For example:

W Step 4	@MyFunction():
*	🖾 B I U A Arial - Aª 10 - A - 🥙 - 🗏 = = = := := := :=

2.4.15 Mobile Settings Dialog

Purpose

This dialog box displays the list of mobile devices that have been configured for use by Rapise and lets you create a new profile, modify a profile or make a new profile based on an existing one.

elect Profile: android-nexus7	Save Test Url Duplicate Create Rename Delete
Connection	
Uri	http://127.0.0.1:4723/wd/hub
Details	
 Primary Capabilities 	(6)
арр	C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUT
browserName	
deviceName	Nexus 7
platformName	Android
platformVersion	4.4.2
udid	015d24bc5d4ffe05
 Common Capabilities 	(9)
autoLaunch	
automationName	
autoWebview	
Primary Capabilities	
esired capabilities are a set of keys and values (i	e., a map or hash) sent to the Appium server to tell the server what kind of automation session
ve are interested in starting up. There are also var	ious capabilities which can modify the behavior of the server during automation.

How to Open

You can open this dialog box from two places:

- From the main Rapise Options ribbon.
- From the Mobile Spy tool when you click on the 'Mobile Profiles' ribbon menu entry.

Menu Options

This dialog box has the following menu options:

- Select Profile This dropdown list lets you select a different mobile profile to be displayed in the dialog.
- Save This button will save the changes to the current mobile profile.
- **Test URL** This button will test the Connection (URL) from Rapise to <u>Appium</u> (which is used to manage the devices) and the connection from Appium to the physical (or simulated) device.
- Duplicate This button will create a new mobile profile based on the currently viewed one.
- Create This button will create a new empty mobile profile that you can edit.
- Rename This button will change the name of the current mobile profile being edited.
- Delete This button will delete the currently displayed mobile profile. There is no undo, so be careful!

Connection

This section lets you enter the URL used to connect to the <u>Appium</u> server which hosts the mobile devices being tested. It is typically of the form:

http://server:4723/wd/hub

Where the port number used by Appium is 4723 by default and the /ed/hub suffix is added.

Details

This section has various settings, some of which are used by all mobile devices, some only by simulated devices, some only by physical devices and some are specific to the type of device (iOS vs. Android):

- Primary Capabilities
 - app The absolute local path or remote http URL to an .ipa or .apk file, or a .zip containing one of these. Appium will attempt to install this app binary on the appropriate device first. Note that this capability is not required for Android if you specify appPackage and appActivity capabilities (see below). Incompatible with browserName. - Values: /abs/path/to/my.apk or http://myapp.com/app.ipa
 - browserName Name of mobile web browser to automate. Should be an empty string if automating an app instead. - Values: Safari for iOS and Chrome, Chromium, or Browser for Android
 - o platformName Which mobile OS platform to use Values: iOS, Android, or FirefoxOS
 - o platformVersion Mobile OS version Values: e.g., 7.1, 4.4
 - deviceName The kind of mobile device or emulator to use Values: iPhone Simulator, iPad Simulator, iPhone Retina 4-inch, Android Emulator, Galaxy S4, etc. On iOS, this should be one of the valid devices returned by instruments with instruments -s devices. On Android this capability is currently ignored.
 - o udid Unique device identifier of the connected physical device Values: e.g. 1ae203187fc012g

Common Capabilities

- o automationName Which automation engine to use Values: Appium (default) or Selendroid
- newCommandTimeout How long (in seconds) Appium will wait for a new command from the client before assuming the client guit and ending the session - Values: e.g. 60
- autoLaunch Whether to have Appium install and launch the app automatically. Default true -Values: true, false
- o language (Sim/Emu-only) Language to set for the simulator / emulator Values: e.g. fr
- o locale (Sim/Emu-only) Locale to set for the simulator / emulator Values: e.g. fr_CA
- o orientation (Sim/Emu-only) start in a certain orientation Values: LANDSCAPE or PORTRAIT
- o autoWebview Move directly into Webview context. Default false Values: true, false
- o noReset Don"t reset app state before this session. Default false Values: true, false
- fullReset (iOS) Delete the entire simulator folder. (Android) Reset app state by uninstalling app instead of clearing app data. On Android, this will also remove the app after the session is complete. Default false - Values: true, false

• For Android Only

- appActivity Activity name for the Android activity you want to launch from your package. This
 often needs to be preceded by a . (e.g., .MainActivity instead of MainActivity) Values:
 MainActivity, .Settings
- appPackage Java package of the Android app you want to run Values: com.example.android.myApp, com.android.settings
- appWaitActivity Activity name for the Android activity you want to wait for Values: SplashActivity
- appWaitPackage Java package of the Android app you want to wait for Values: com.example.android.myApp, com.android.settings
- o deviceReadyTimeout Timeout in seconds while waiting for device to become ready Values: 5
- androidCoverage Fully qualified instrumentation class. Passed to -w in adb shell am instrument e coverage true -w - Values: com.my.Pkg/com.my.Pkg.instrumentation.MyInstrumentation
- enablePerformanceLogging (Chrome and webview only) Enable Chromedriver's performance logging (default false) - Values: true, false
- androidDeviceReadyTimeout Timeout in seconds used to wait for a device to become ready after booting - Values: e.g., 30

- androidDeviceSocket Devtools socket name. Needed only when tested app is a Chromium embedding browser. The socket is open by the browser and Chromedriver connects to it as a devtools client. - Values: e.g., chrome_devtools_remote
- o avd Name of avd to launch Values: e.g., api19
- avdLaunchTimeout How long to wait in milliseconds for an avd to launch and connect to ADB (default 120000) - Values: 300000
- avdReadyTimeout How long to wait in milliseconds for an avd to finish its boot animations (default 120000) - Values: 300000
- o avdArgs Additional emulator arguments used when launching an avd Values: e.g., -netfast
- o useKeystore Use a custom keystore to sign apks, default false Values: true or false
- keystorePath Path to custom keystore, default ~/.android/debug.keystore Values: e.g., /path/ to.keystore
- o keystorePassword Password for custom keystore Values: e.g., foo
- o keyAlias Alias for key Values: e.g., androiddebugkey
- keyPassword Password for key Values: e.g., foo
- chromedriverExecutable The absolute local path to webdriver executable (if Chromium embedder provides its own webdriver, it should be used instead of original chromedriver bundled with Appium) - Values: /abs/path/to/webdriver
- autoWebviewTimeout Amount of time to wait for Webview context to become active, in ms. Defaults to 2000 - Values: e.g. 4
- intentAction Intent action which will be used to start activity (default android.intent.action.MAIN) -Values: e.g.android.intent.action.MAIN, android.intent.action.VIEW
- intentCategory Intent category which will be used to start activity (default android.intent.category.LAUNCHER) - Values: e.g. android.intent.category.LAUNCHER, android.intent.category.APP_CONTACTS
- intentFlags Flags that will be used to start activity (default 0x10200000) Values: e.g. 0x10200000
- optionalIntentArguments Additional intent arguments that will be used to start activity. See Intent arguments - Values: e.g. --esn <EXTRA_KEY>, --ez <EXTRA_KEY> <EXTRA_BOOLEAN_VALUE>, etc.
- o unicodeKeyboard Enable Unicode input, default false Values: true or false
- resetKeyboard Reset keyboard to its original state, after running Unicode tests with unicodeKeyboard capability. Ignored if used alone. Default false - Values: true or false
- noSign Skip checking and signing of app with debug keys, will work only with UiAutomator and not with selendroid, default false - Values: true or false
- ignoreUnimportantViews Calls the setCompressedLayoutHierarchy() uiautomator function. This
 capability can speed up test execution, since Accessibility commands will run faster ignoring some
 elements. The ignored elements will not be findable, which is why this capability has also been
 implemented as a toggle-able setting as well as a capability. Defaults to false Values: true or false
- For iOS Only
 - o calendarFormat (Sim-only) Calendar format to set for the iOS Simulator Values: e.g. gregorian
 - bundleld Bundle ID of the app under test. Useful for starting an app on a real device or for using other caps which require the bundle ID during test startup. To run a test on a real device using the bundle ID, you may omit the "app" capability, but you must provide "udid". - Values: e.g. io.appium.TestApp
 - o udid Unique device identifier of the connected physical device Values: e.g. 1ae203187fc012g
 - launchTimeout Amount of time in ms to wait for instruments before assuming it hung and failing the session - Values: e.g. 20000
 - locationServicesEnabled (Sim-only) Force location services to be either on or off. Default is to keep current sim setting. - Values: true or false
 - locationServicesAuthorized (Sim-only) Set location services to be authorized or not authorized for app via plist, so that location services alert doesn"t pop up. Default is to keep current sim setting. Note that if you use this setting you MUST also use the bundleld capability to send in your app"s

bundle ID. - Values: true or false

- autoAcceptAlerts Accept iOS privacy access permission alerts (e.g., location, contacts, photos) automatically if they pop up. Default is false. - Values: true or false
- nativeInstrumentsLib Use native intruments lib (ie disable instruments-without-delay). Values: true or false
- nativeWebTap (Sim-only) Enable "real", non-javascript-based web taps in Safari. Default: false.
 Warning: depending on viewport size/ratio this might not accurately tap an element Values: true or false
- safariAllowPopups (Sim-only) Allow javascript to open new windows in Safari. Default keeps current sim setting - Values: true or false
- safarilgnoreFraudWarning (Sim-only) Prevent Safari from showing a fraudulent website warning. Default keeps current sim setting. - Values: true or false
- safariOpenLinksInBackground (Sim-only) Whether Safari should allow links to open in new windows. Default keeps current sim setting. - Values: true or false
- keepKeyChains (Sim-only) Whether to keep keychains (Library/Keychains) when appium session is started/finished - Values: true or false
- o localizableStringsDir Where to look for localizable strings. Default en.lproj Values: en.lproj
- o processArguments Arguments to pass to the AUT using instruments Values: e.g., -myflag
- o interKeyDelay The delay, in ms, between keystrokes sent to an element when typing. Values:
 e.g., 100
- showIOSLog Whether to show any logs captured from a device in the appium logs. Default false
 Values: true or false
- sendKeyStrategy strategy to use to type test into a test field. Simulator default: oneByOne. Real device default: "grouped - Values: oneByOne, grouped or setValue
- screenshotWaitTimeout Max timeout in sec to wait for a screenshot to be generated. default: 10
 Values: e.g., 5
- waitForAppScript The ios automation script used to determined if the app has been launched, by default the system wait for the page source not to be empty. The result must be a boolean Values:
 e.g. true;, target.elements().length > 0;, "\$.delay(5000); true;

2.4.16 Mobile Test Locator Dialog

Purpose

This dialog box lets you create a test locator for mobile applications using one of the supported methods (XPath, ID, etc.) and display the results of using that locator interactively.

est Locator			
//UIAApplication[1]/UIAWindow[1]/UIATableView[1]/UIATableCell[1]/UIAStati	XPath	24	
	Id	Calculated Enabled	True
	CSS	Id IndexInParent	Master - Detail Table View, -1
	Tag Name	TagName Text	UIAStatic Text Master - Detail Table View,
	Class Name	Title XPath	(-1) UIAStatic Text[0,64][320 //UIAApplication[1]/UIAWir
	Name	A Layout Height	44
	Android	Width X	320 0
	iOS	Properties	64
	Accessibility	label	Master - Detail Table View.
	Link.	name value	Master - Detail Table View,
	Partial Link	Ι	
	Script	[
		Enabled	

How to Open

You open this dialog from the Mobile Spy by clicking the Test Locator button on that dialog.

How to Use

To use this dialog, you simply choose which type of locator you wish to test (in the example above we are using XPath on an iOS device) and click the button. The properties discovered from using this locator on the device in question will be displayed in the right panel.

The following locator types are available:

- XPath This allows you to enter an XPath selector that uniquely locates a specific element in the mobile object hierarchy
- Id This allows you to enter the ID of a specific object and test to see if it can be found.
- CSS For mobile website testing only, this lets you enter a CSS selector that can uniquely locate an object
- Tag Name This lets you find elements by their Tag Name field. For web testing this is the name of the DOM element.
- Class Name This lets you find elements by their UI Component Type
- Name This lets you find elements by their Name field
- Android This lets you enter a string corresponding to a recursive element search using the UiAutomator Api (Android-only)
- **iOS** This allows you to enter a string corresponding to a recursive element search using the UIAutomation library (iOS-only)
- Accessibility This lets you enter a string corresponding to a recursive element search using the Id/

Name that the native Accessibility options utilize.

- Link Based on the WebDriver standard, it lets you find hyperlinks using an *exact match* of the link anchor text
- **Partial Link** Based on the WebDriver standard, it lets you find hyperlinks using a *partial match* of the link anchor text
- Script For iOS testing, this lets you enter raw script that will be sent to the iOS device to find the element

2.4.17 NameValue Collection Editor Dialog

Screenshot

embers:	new_test_param pro	perties;
ServerProjectId ServerTestCaseFolderId ServerTestCaseId new_test_param	Ask DefaultValue Description IsOutput Name Options Variable	ters False False new_test_param (Collection)
		OK Cancel

Purpose

To specify Custom Strings and their values.

How to Open

Open from the <u>Settings Dialog</u>, TestParams option:

Set	tings 부
	2 ↓
⊿	Advanced
	CommandLi
	EntryPoint Test
⊿	TestParams 3 items
	Serverf 1
	Server 1
	Server 2
\triangleright	Execution
\triangleright	Methodology
\triangleright	Recording
\triangleright	Screen Capture
\triangleright	Settings
Те	stParams
	st Parameters.
Ma	ay be used for integration with oth

Widgets

• Add a custom string. If you press Add, you'll see	this:
---	-------

Members:	 new_test_param pro	operties;
0 ServerProjectId 1 ServerTestCaseFolderId 2 ServerTestCaseId 3 new_test_param Add <u>R</u> emove	Ask Default Value Description IsOutput Name Options Variable	eters False False new_test_param (Collection)
	-	OK Cancel

- Remove: removes selected custom string.
- OK: Save changes and close dialog.
 Cancel: Close dialog without saving changes.

2.4.18 NeoLoad Convertor Dialog

Screenshot			
Convert Current Test to Neo	Load —		×
Virtual User			
VirtualUser			
Base Container			
Actions			•
Container/Business Transact	tion		
Detect dynamic parameter	S		
	Cancel	Cor	vert

Purpose

The purpose of this dialog is to allow you to convert a functional test script from Rapise into a protocolbased performance script that can be executed using the <u>NeoLoad</u> performance testing tool from Neotys.

How to Open

Go to the Test ribbon and click the Convert to NeoLoad button:

Virtual User

In this field, you need to enter the name of the virtual user to create in NeoLoad:

- The default value is "VirtualUser"
- If the name is already used, then it is automatically renamed using "_X" suffix, with X an integer incremented.
- If the name has invalid characters then they will be escaped as an underscore (_).

Base Container

This specifies the base container where we want to start the recording (Init / Actions / End)

• The default value is Actions.

Container/Business Transaction

This is used to specify the current recording container in NeoLoad. It is just based on a single level. There is no way to specify a tree of containers.

- The default is no container.
- If the name is already used then it will be made unique by adding _1, _2, etc.
- If the name is empty then no container will be used.

Detect Dynamic Parameters

When you check this box, Rapise tells NeoLoad to scan the protocol traffic to look for known dynamic parameters (e.g. Session IDs, ASP.NET ViewState) that change on each HTTP request and need to be parameterized by NeoLoad to ensure the performance scripts are robust and well-defined (v.s. having a hardcoded Session ID).

Actions

- Convert will start the Rapise > NeoLoad test conversion process
- Cancel will abort the conversion and return you to Rapise

2.4.19 NeoLoad Settings Dialog

Screenshot				
NeoLoad Integration Settings		_		×
<u>≷</u> ∎ 2↓ □				
~				
NeoLoad Data Exchange URL	http://loca	alhost:7400/Da	taExchang	e/v1/Serv
NeoLoad Proxy Address	127.0.0.1	:8090		
NeoLoad REST Service URL	http://loca	alhost:7400/Re	cording/v1	/Service.
NeoLoad Data Exchange URL Rapise uses this URL to send perform	mance me	etrics to NeoLo	oad contro	ller.
		OK	Са	ncel

Purpose

The purpose of this dialog is to allow you to specify the global options for integrating Rapise with <u>NeoLoad</u>, the performance testing tool from Neotys.

How to Open

Go to the Options ribbon and click the NeoLoad Integration Settings button:

Settings

Normally there is no any reason to change these settings, but for completeness they are described below:

- NeoLoad Data Exchange URL this is the URL to the NeoLoad data exchange API
- NeoLoad Proxy Address this is the IP address and prt of the NeoLoad HTTP proxy
- NeoLoad REST Service URL this is the URL to the NeoLoad recording service REST API

2.4.20 Object Tree Dialog

Objects
√ id n id[]
Object Tree C:\Users\adam.sandman\Docum
Inflectra Library Information System - Inte
Author_ [Author:]
Author_Management [Author Managem
A Book_Management [Book Management]
ab ctl00\$MainContent\$btnSubmit [ctl00\$M
A Home [Home]
▶ 🗆 librarian [librarian]
A Log_In [Log In]
Interpretation Name [Name:]
a 🔲 Library Information System Log In - Intern
▹ ๗ ctl00\$MainContent\$LoginUser\$Logi [ctl
In the second
Isername_[Username:]
Library Information System Book Manage
▶ 🛄 Global
(i) User Functions
var User Variables
Objects Files SOAP Settings

Purpose

Display learned objects.

How to Open

The **Objects** dialog is part of the **Default Layout**.

Context Menu (root node)

Right click the Object Tree node to see:

Refresh
Collapse all
Expand all
Filter

- Refresh checks for new objects to display.
- Collapse all collapses the entire object tree.
- Expand all expands the entire object tree.
- Filter... filters the object tree.

Context Menu (object)

Right click on an object in the Object Tree dialog to see:

Flash	
Re-Learn	
Remove	
Clone	
Add Parameter	

- Flash opens the application/url where the object is located. A frame will blink around the object to show you where it is on the page.
- Re-Learn will open up the <u>Recorder</u>, allowing you to re-learn the object. This is useful if the AUT has changed and the object definition will no longer correctly locate the object.
- Remove simply removes the selected object from the tree.
- Clone makes a copy of the object definition and adds the cloned version into the tree. You can then make changes to the cloned copy.
- Add Parameter opens up a dialog box that lets you add a custom parameter to the learned object definition (stored in the Test.objects.js file).

2.4.21 Options Dialog

General Tools	
Directories	
Help Files	C:\Users\Public\Documents\Rapise\Help
Samples	C:\ProgramData\Documents\Rapise\Sample:
User Libraries	C:\ProgramData\Documents\Rapise\Scripts\Lib
User Scripts	C:\ProgramData\Documents\Rapise\Scripts
A Misc	
SplashScreen	True
Settings	
Auto Reload Modified Files	False
Default Folder	C:\Users\adam.sandman\Documents\My []
Enable Execution Monitor	True
Frame Style	Modern
Default Folder	
Default folder for creating new tests.	

E Misc

Use the Options dialog to change the global Rapise settings. Your changes will apply to all tests.

How to O	per	ו
Go to the O	to the Options ribbon and click the Settings	
	Q	Build OCR Font Database
	3	Reset Layout
bettings	S	Spira Connection Settings
		General
Misc		

SplashScreen True SplashScreen: A splash screen is the image that appears while a program initializes. The Rapise splash screen looks like this:

© 2017 Inflectra Corporation

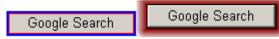
D		-
R	nr	ICO
110	IP	IJE R
	1	0
ading module Acc	essibility	

1.1.24

Set SplashScreen to False to prevent the splash screen from appearing.

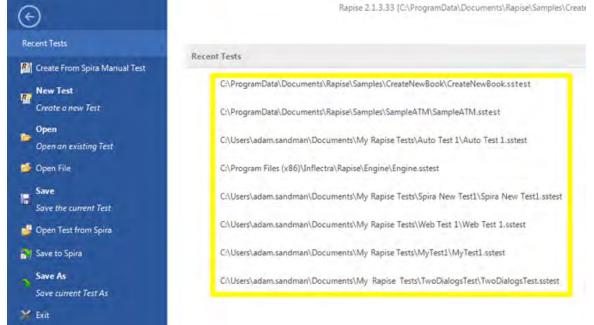
Se	ettings		
	Settings AutoReloadModifiedFiles	True	
	DefaultFolder	Temp	
13	FrameStyle	Modern	
1.5	LoadLastTestOnStartup	True	
	NormalizeFileNames	True	
	RecentTests	10	
	ShowStartPageOnStartup	True	
15	StyleLibrary		

- AutoReloadModifiedFiles: If set to True, any files you modify outside of Rapise are automatically reloaded in Rapise.
- **DefaultFolder** specifies where new tests are kept before you explicitly save them. The location is relative to the Rapise executable.
- DefaultSpy specifies which of the various types of Object Spy will be displayed by default.
- Enable Execution Monitor specifies whether the execution monitor dialog box will be displayed during <u>playback</u>.
- FrameStyle: Specifies which frame to draw around objects when you <u>Record</u>, <u>Learn</u>, and <u>Spy</u>. The Basic frame is on the left and the Modern frame is on the right:



- LoadLastTestOnStartup: If set to True, Rapise will open the last test you worked on and saved. If set to False, Rapise will create a new test named MyTest<#> where <#> is an integer. A folder for MyTest<#> is created in the folder specified by the DefaultFolder option.
- NormalizeFileName: If set to True, files are referred to (in the *.sstest file) using a path relative to the *.sstest file. Otherwise, their absolute path is used.

• RecentTests: The maximum number of recent files displayed in the Recent Tests list. To see the Recent Tests list, open the Application Menu:



- Remember Debugger Layout: If True, Rapise will remember the window layout for debug mode separately. For example, this may be useful if you want to work full screen while authoring the Test and half-screen to debug. This way the AUT and the Rapise debugger fit on the screen.
- ShowDashboardOnStartUp: If True, the <u>Spira Dashboard</u> will open automatically when Rapise is opened.
- ShowStartPageOnStartUp: If True, the Start Page will open automatically when Rapise is opened.
- StyleLibrary: determines the color scheme of the Rapise window. If you click on StyleLibrary, you'll notice that a drop down arrow appears to the right. Press the arrow to see all of the Style options:

RecentTests	10
ShowStartPageOnStartup	True
StyleLibrary	Office2007Blue
StyleLibrary Path to a file having style library configuration.	Office2007Black Office2007Blue Office2007Silver
	OK Cancel

2.4.22 Output View

Screenshot			

Output	
	40 🗊 🗇 🗎
Microsoft (R) Windows Script Host Version Copyright (C) Microsoft Corporation. All right	
SeS Player Starting Verbose level:1 Creating a locator:HTML Creating a locator:REST	
<	
Output Warnings Errors Find Results	

The Output View displays Rapise output. The amount of output depends on the Verbosity Level.

How to Open

The **Output** view is part of the <u>Default Layout</u>.

Writing to the Output View

Use the global Log() function to write to the Output View.

Widgets

	Thageto
•	The text box is a search box.
•	• The icons from left to right are Find Next Entry 🧖, Copy Selected 🤍, Clear All Text 🚫, and
	Select All Text 🔜.

2.4.23 Properties Dialog

So	creenshot		
Pro	operties		Ļ
	2 ↓ 🖻		
	Value		*
Ξ	Misc		
Ŧ	Locators		
	version	0	*
Ot	ocators bject locators t entify the objec	hat can be used at within AUT.	d to

Purpose

To display the properties of the object, file, or folder you last clicked on. Objects are in the <u>Object</u> <u>Tree Dialog</u> and files/folders are in the <u>Test Files Dialog</u>.

How to Open

The Properties Dialog is part of the Default Layout.

2.4.24 Recording Activity Dialog

Screenshot

Recording Activity for "Internet Explorer HTML"							
#	Object	Action	Data	Comment			
Verify (Ctrl+1)		Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel
Ready						Advanced>>	Transparent

Purpose

The **Recording Activity Dialog** is used for <u>Recording</u>, Analog recording (<u>absolute</u> and <u>relative</u>), <u>Object</u> <u>Learning</u>, and creating <u>Simulated Objects</u>.

How to Open

This dialog is opened slightly differently depending on the type of test being recorded:

- Web Testing When you click on the Record / Learn button in the Test ribbon, this dialog will be displayed.
- Mobile Testing When you click on the Record / Learn button in the Test ribbon, this dialog will be displayed.
- **Other Testing** After selecting the application to be recorded in the <u>Select an Application to</u> <u>Record</u> dialog, this dialog is displayed.

Advanced Mode vs. Standard Mode

When you first open the dialog, it will be displayed in Standard mode:

#	Object	Action	Data	Comment			
	Verify (Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel

This mode shows the most widely used options and is normally sufficient for most recording needs.

However if you need to do analog recording or you want more control over the type of object being recorded, you can click on the **Advanced** link to switch to Advanced mode:

Rec	Recording Activity for "Internet Explorer HTML"											
#	Object	Action	Data	Comment								
	Verify (Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Pause	Finish (Ctrl+3)	Cancel					
	Analog (Ctrl+	. ,		Simu	lated	•						
Read	у					Advanced>>	> Transparent					

Clicking on the Advanced link will switch it back to Standard mode.

The Grid

As you interact with the AUT (Application Under Test), your actions are recorded in the grid of the **Recording Activity dialog**. The following screenshot shows the Recording Activity dialog after two interactions with <u>www.google.com</u>: (1) first, **Inflectra** was entered into the query text box and (2) the **Google Search** button was then pressed.

#	Object	Action	Data	Comment
巡1	q	Set	Inflectra	Set Text Inflectra in q
102	btnG	Click		Click on btnG

Context Menu

Reco	rding Activity	for "Inte	ernet Expl	lorer HTML"		
#	Object	Action	Data	Comment		
圆1	Products	Click		Click on Pre	oducts	
1⊘2 1⊘3	SpiraTest	Click		Click on Sp	piraTest	
1 3	Resources	Click		Click on P		
104	Trials and	Click		Click on 1	Delete Action(s)	
					Edit Action	

SPY

(Ctrl+5)

If you right click in the grid, you'll see a context menu with three options:

Unknown:

Verify

(Ctrl+1)

• Delete Action removes the selected row.

Learn

(Ctrl+2)

• Edit Action opens the Action Editor Dialog. This is also opened by double-clicking a grid entry.

Try Action

Pause

Finish

(Ctrl+3)

Advanced>>

Cancel

Transparent

Object ID:	
Action:	DoAction()
Comment:	Press button 'OK'

• Press Try Action and Rapise will execute the action.

Standard Mode Features SPY Verify Learn Finish Pause Cancel (Ctrl+1) (Ctrl+2) (Ctrl+5) (Ctrl+3) Analog (Ctrl+4) Simulated Transparent Ready Advanced>>

The following options are available in the dialog in both modes:

- Verify (Ctrl+1) Press to open the Verify Object Properties dialog.
- Learn (Ctrl+2) Use to <u>learn</u> an object. Place the mouse cursor over the object you wish to learn. It should become highlighted with a purple box. Press Ctrl+2 while the object is highlighted. You will see a line added to the Recording Activity dialog, signifying that the object was learned.
- SPY (Ctrl+5) The Spy Button opens the Object Spy dialog. The Object Spy dialog allows you to view the state of the objects in your program. Viewing object state is called <u>Object Spying</u>. The Object Spy dialog is described <u>here</u>. You can also use the SPY button to learn an object that is not visible or covered by another object.

- Pause The Pause Button temporarily stops Recording. Any interacting you do with the AUT is ignored. When you press the Pause Button, the title of the button changes to Resume. Press the Resume button to continue recording.
- **Finish** (Ctrl+3) The Finish button ends the Recording session. The dialog is closed, and the information collected during Recording is used to create a script. The script is displayed.
- Cancel The Cancel button stops Recording, closes the dialog, and discards any actions recorded or objects learned during the Recording session.

Advanced Mode Features

The following additional features are available in Advanced mode:

- Analog (Ctrl+4) The Analog button begins <u>Analog Recording</u>. Analog Recording tracks mouse movements, keyboard inputs, and clicks. To end Analog Recording, press CTRL+Break
- Simulated Drop-Down Menu An object can be learned if it matches a rule specified in the <u>Recording/Learning libraries</u> available. The drop-down menu lists the possible rules for learning objects in the current application:

Reco	rding Activity	for "Inte	ernet Ex	xplore	er HTML"					
#	Object	Action	Data		Comment					^
圆1	Products	Click			Click on Products					
1⊘2	SpiraTest	Click		Rule	4	*	Library			
103	Resources	Click		_Sim	ulated					
104	Trials and	Click		HTML	_Object		Internet Explo			~
	Verify Ctrl+1)	Learn (Ctrl+2)							Can	icel
	Analog (Ctrl+4)				_Simulated		-			
Pause	d				Start S	Sim	ulated Lea Ad	vanced>>	Tran	nsparent

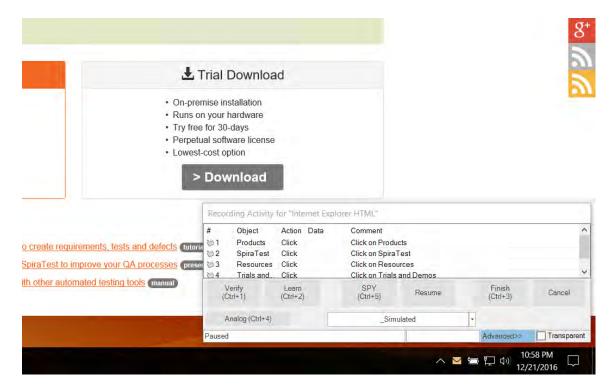
If you cannot learn an object with one library rule, try another in the list. Create a Simulated Object only if the other, more flexible alternatives have been exhausted.

To learning and object using a specific library:

- 1. Double click on a rule in the drop down list. The button text should change to the text that you selected
- 2. Press the button
- 3. Select an object on the screen and make sure it is highlighted with a rectangle
- 4. Press Ctrl+2 to learn the object

Transparent Option

While the Recording Activity (RA) dialog is open, it is always on top. The Transparent checkbox makes the RA Dialog transparent so that you can interact with objects behind it. The image below illustrates the difference:



When you click the Transparent option, it will look like the following:

						8+
±1	rial Download	d				8+ ⊘
 Runs or Try free Perpetu Lowest- 	nise installation n your hardware for 30-days ial software license cost option Download	I				
o create requirements, tests and defects (tutoria SpiraTest to improve your QA processes (preser	NA Communit	Action Data Click Click Click Click	Comment Click on Produ Click on Spirat Click on Reso Click on Trials	Test uces		Â
ith other automated testing tools (manual)	Verify (Ctrl+1)	Leam (Ctrl+2)	SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel
	Analog (Ctrl+4)		_Sim	ulated	-	
	Paused				Advanced>>	🖌 Transparent
				~		8 PM

The main advantage of this mode is that you can click in the area where the RA dialog is displayed and the clicks will be sent through to the AUT.

2.4.25 Replace Text Dialog

Screenshot								
Find and Repla	ace 📧							
🆄 Find	ti Replace							
Find what:	Search 🝷							
Replace with:	Spira_RestService_40_Project 💌							
Look in: Current Document								
Find options	se Include <u>s</u> ub-folders							
Match who	ole word Include hidden							
Look at these	file types: *.js							
Use: Standar	rd Search 🔹							
Find and Replace Find Replace Find what: Search Replace with: Spira_RestService_40_Project Look in: Current Document Find options Include sub-folders Match gase Include hidden Look at these file types: *.js								
Find !	Next Replace Replace All							

Purpose

Replace occurrences of the Search Term text with the Replacement Text in the currently visible <u>Source</u> Editor.

How to Open

Ribbon > Edit Tab > Search menu > Replace button.

Replace Tab

- Find what: Place the string you would like to search for in the Find what text box.
- Look In: this option specifies where the search will take place. You can limit the search to: current document, current selection, current test, the entire test and subtests, or a specific folder.
- Match case option: If unselected, case is ignored in the search.
- Match whole word option: If set to true, parts of words will not count as matches.
- Replace with text-box: All occurrences of the string in the Find what text-box will be replaced with the string in the Replace with text-box when you press the Replace button.

2.4.26 Report Viewer

Screenshot

C:\ProgramData\Documents\Rapise\	Samp	les\	. 3 1	A BEER	- 🗁 🗙	Merge Cells	E	Collapse								
😵 Plain 🐄 Hierarchical File			Expe to Ex	ort Export	ave Layout Leset Layout Layout	Choose Columns Data	Images Disp	Expand			^					
Test Files II	-	Start F	- (19) c	2 4 C - C		teNewBook_2014-09-04_11-47.tr	F	nay			- X					
4 🛄 Test	1.					(enewbook_2014-03-04_11-47,8	p				- 10					
Reports	Dr	ag a	column he	eader here to g	roup by that column.											
 Scripts CreateNewBook.js 		#	Туре	Start -	N	lame	Status	Browser	Comment	Iteration						
CreateNewBook.objec	1.4	1	(M)	-	NI.		140	W)	ED.	-						
CreateNewBook.user.j			Assert	11:47:13.659	Username:.DoSetText(["	librarian"])	Pass	Internet Explorer HTML	Returned Value: true	0						
			Assert	11:47:13.862	Password:.DoSetText(["I	ibrarian"])	Pass	Internet Explorer HTML	Returned Value: true	0						
			Assert	11:47:14.096	ctl00\$MainContent\$Logi	nUser\$LoginButton.DoClick(Pass	Internet Explorer HTML	Returned Value: true	0						
			Assert	11:47:14.517	Book Management.DoCli	ck([])	Pass	Internet Explorer HTML	Returned Value: true	0						
			Assert	11:47:15.063	(Create new book) .DoCl	ick([])	Pass	Internet Explorer HTML	Returned Value: true	0						
								Assert	11:47:15.609	Name:.DoSetText(["The	Restaurant at the end of th	Pass	Internet Explorer HTML	Returned Value: true	0	L
							Assert	11:47:15.812	2 Author:.DoSelect(["Agatha Christie"])	Pass	Internet Explorer HTML	Returned Value: true	0			
							Assert	11:47:16.015	Genre:.DoSelect(["Science Fiction"])		Pass	ss Internet Explorer HTML	Returned Value: true	0		
			Assert	11:47:16.186	ctl00\$MainContent\$btn5	ubmit.DoClick([])	Pass	Internet Explorer HTML	Returned Value: true	0						
Test Files Settings Object Tree	×		Assert	11:47:16.654	Failure in Test		Fail	Internet Explorer HTML		0.						
Properties 0						Test Fail										
21 0	-										٣					
	Outp	ut									Πļ.					
	8				🔊 📄 🚫 🗎						_					
				pt Host Version 5.8 poration. All rights reser	ved.						ů.					
	Ver Cre	SeS Rayer Stating Vetose Ivel: 1 Creating a locator Adobe Rex Adobe Rex: Revis Noaded														
	1000	out W	/arrtings Errors	Find Results												
	5.04		and a second	INUM						Inflectra Corpor	ation					

Purpose

The Report Viewer displays test result (trp) files.

How to Open

Use the <u>Test Files Dialog</u> to open a report (trp) file. The report file will be opened in a **Report Viewer** in the <u>Content View</u>. The <u>Report Tab</u> of the Ribbon will also open.

Or, you can <u>Playback</u> the test script. The report file will display in a **Report Viewer** after the test completes.

See Also

- For more info on Reports, see Automated Reporting.
- For information on manipulating reports, see Ribbon: Report.

2.4.27 REST Definition Editor

Screenshot

REST		2 LibraryInform	etionSystem.re	Start Page 2 WebServicesREST is								
*** * .		REST Request										
LibraryInformat		Name	Get_Books									
- Get_Sessio Get_Books Get_BookB	lyid .	Method	GET ~	http://www.ibraryinformationsystem.org/services/sets	service svc/book?session_id+(session_id)	Send Record						
- Update_Book - Update_Book - Delete_Book	ok sk	Credentials Headers	librarian 💌									
- Updatelmag Updatelmag Get Sessio	tionSystem rest		Name Accept	Value Value	Operations 🔀							
- Get_Books			Content-Type	application/json	×							
- Insert_Book	→C QK_BookByld →C Issart_Book → Update_Book →C Delete_Book →C UpdateImage_Book		•									
- Delete_Boo			Name		Operations							
- Updateimag			(session_id)	12345	Insert in URL 🛛 🛞							
Objects Files PES	T Settings	Body	\odot	9								
21 1												
Misc		Response Headers										
Body	the second second		Nario		Value							
Crédential Headers	System Net NetworkCrede (Collection)	Status Code		200 OK								
Method	GET	Content-Length	Y	3586								
Name Parameters	Get_Books (Collection)	Cache-Control		private								
Url	http://www.libraryinformatic	Content-Type		application/json, charset=utl-8								
		Date		Thu. 22 Dec 2016 19:40:50 GMT								
		Set-Cook in		ASP NET_Sessionid=5wbbshommj	252045sytbs: path+/, HttpOnty							
		Server		Microsoft-IIS(8.0								
Nome		X-AspNet-Vars	ion	4 0 30319								
		Y.0		AGE NET								
		Hesponse Body	Fungarine Headers	Output Warnings Errors Find Results								

Purpose

The **REST Definition Editor** allows you to edit <u>REST web service</u> definition files (.rest).

How to Open

Use the <u>Add Web Service Dialog</u> to create a new REST definition (.rest) file. The definition file will be opened in a **REST Editor** in the <u>Content View</u>. The <u>REST Tab</u> of the Ribbon will also open.

File	Test Opti	ons RI	EST			
Save Requests	Update Object Tree	Add Header	Add Parameter	Add Credentials	Get_Session GET *Get_Authors GET	Create Script Remove Step Clean
	File		Edit		Script Step	S

Or, you can double-click on an existing .rest file in the <u>Test Files View</u> explorer window. The definition file will be opened in a **REST Editor** in the <u>Content View</u>. The <u>REST Tab</u> of the Ribbon will also open.

Request

REST Request							
Name:	Get_Books						
Method:	GET v	http://www.ilbraryinformatio	nsystem org/services/rest	ervice svc/book?session_id=[session_id]	~	Send	Record
Credentials	hbrarian 🔀						
leaders	\odot						
	Nam		Value	Operations			
	Accept	application/jso		×			
	Content-Type	application/jso	1				
Parameters	\odot						
	Nam	0	Value	Operations			
	{session_id}	12345		Insert in URL			
Body:	٢						
	Auto Raw	XML JSON					
	1						^
	¢.						2

The request form has several sections that you need to populate:

- Method the type of HTTP request being made (GET, POST, PUT, DELETE, etc.)
- URL the URL of the web service request with any parameter tokens included (e.g. {session_id} in our example above)
- Credentials Any HTTP Basic Authentication Headers
- Headers Any other HTTP headers (both standard and custom)
- **Parameters** Any parameters that have been defined in the URL that will be called from the Rapise test script.
- **Body** The body of the request (for POST and PUT requests). This can be in any text-serialized format such as XML or JSON.

Response

The HTTP Response Headers are displayed:

Name	
Status Code	200 OK
Content-Length	3586
Cache-Control	private
Content-Type	application/json; charset=utf-8
Date	Thu, 22 Dec 2016 19:40:50 GMT
Set-Cookie	ASP.NET_SessionId=5wkbu5InommjIz52045sytbo; path=/; HttpOnly
Server	Microsoft-IIS/8.0
X-AspNet-Version	4.0.30319
X-Powered-By	ASP.NET

The HTTP Response in XML format is formatted and displayed:

Response Body	
Auto Raw XML JSON	
<pre>1 <?xml version="1.0" encoding="utf-8"?></pre>	
2 <pre>2 </pre>	

The HTTP Response in JSON format is formatted and displayed:



This displays the output from the last web service request. It has several tabs:

- **Response Header** Displays a list of the HTTP response headers (name and value). If the request received a 200 OK code back, it's displayed in green, if it receives an error code back, it's displayed in red.
- Response Body
 - o Raw Displays the raw text of the HTTP response body received from the server.
 - **XML** If the received body content is identified as XML, this tab displays nicely formatted XML that is easier to read than the raw response body.
 - JSON If the received body content is identified as JSON, this tab displays nicely formatted, indented JSON that is easier to read than the raw response body.

Operation Explorer

REST
🖶 🚰 🐚 🗙 🖕
▲ 📓 LibraryInformationSystem.rest
- Get_Session
- Get_Books
- Get_BookById
- Insert_Book
- Update_Book
- Delete_Book
- Updatelmage_Book
LibraryInformationSystem.rest
- Get_Session
- Get_Books
-© Get_BookById
- Insert_Book
- Update_Book
- Delete_Book
- Updatelmage_Book
Objects Files REST Settings

This section lets you add, open, delete and clone REST requests in the definition file.

- Add request Adds a new REST operation to the current .REST definition file
- **Open request** Opens the currently selected REST operation in the current .REST definition file. This is the same as double-clicking on the item name.
- Clone request Makes a copy of the currently selected REST operation and allows you to give the copy a new name.
- **Delete request** Deletes the currently selected REST operation from the current REST definition file.

See Also

- For more info on REST Web Services, see REST Web Services.
- For a tutorial on creating a REST web service test, see the Web Services REST Tutorial.

2.4.28 Ribbon: Test

Screenshot					
	Report				** •
Dpen 🗁 Open	🖾 Start Page 🛬 RVL	0	No Debugging +	About Object Mgr	
Save to Spira	🕼 Test 🛛 🖓 Manual Steps	Record Play	Level 1 (Minimal) +	📍 Activation 💪 Convert to NeoLoad	Find in Help
💣 Open Test from Spira	📓 Functions 👍 Web Services	Spy		🚺 Libraries 📑 Spira Properties	Files
File	Shortcuts	Record and Execute	Debugging	Tools	Help

Purpose

The **Test** tab is the primary tab in the Rapise ribbon and provides tools to help with creating and executing tests. It also provides the options to add web services and/or manual test steps to the current test.

How to Open

The **Test** tab is always available in the ribbon.

File		
	📴 Open	
Save	🔉 Save to Spira	
Save	🍯 Open Test from Spira	
	File	

The File section provides the following options:

- Save saves the current test locally. To save to a different location, use the Save As option in the File menu.
- Save to Spira allows you to save the Rapise test so that it updates the version in your <u>Spira</u> test management repository
- Open Test from Spira allows you to open a Rapise test that is stored in a <u>SpiraTest</u> test management repository

Sho	ortcuts		
3	Start Page		RVL
3	Test		Manual Steps
8	Functions	-	Web Services
	Sh	orto	uts

The Shortcuts section provides links to the most commonly used functionality in Rapise:

- Start Page this button opens the Rapise Start Page.
- RVL- this button opens the Rapise Visual Language (RVL) editor and displays the RVL Ribbon.
- Test This opens the primary test script file (normally MyTest.js where MyTest is the name of your test)
- Functions This opens the user functions <u>script file</u> containing any user-defined testing functions (called MyTest.user.js where MyTest is the name of your test)
- Manual Steps displays the <u>Manual Test Steps Ribbon</u> that lets you view and edit the <u>manual tests</u> associated with this test.
- Web Services allows you to add a new web service definition to your Rapise test. Clicking on this
 displays the Add Web Service dialog box.

Record and Execute



The Record and Execute section provides tools relating to recording tests, learning objects, spying objects in the running application and executing the test.

- Record this button is used for recording and learning, clicking it will open the <u>Recording Activity</u> <u>Dialog</u>.
- Spy this button opens the <u>Spy Dialog</u>. You use the Spy to look at the running application and find specific objects that you want to perform an <u>operation</u> or <u>verification</u> on.
- **Play** this button executes the current test. You can change which test script to open in the <u>Settings Dialog</u>. The test script is specified by **Settings** > **ScriptPath**.

Note there are some additional options available when using the **Record** button:

- Record clicking Record on its own will start Rapise recording using the current selected libraries (this is the default action)
- Shift+Record This brings up the library selector (shown above), and is equivalent to clicking the [] libraries button in the ribbon before starting recording.
- CTRL+Record This brings up the application selector (see below) and lets you change the application and library being recorded:

Debugging	
No Debugging	-
Level 1 (Minimal)	-
Debugging	

The Debugging section provides tools for debugging the code in the test during playback.

- **Debugging Level** The top drop-down list specifies if you would like to use an <u>External Debugger</u>. If so, you can either connect on execution (the **Run with External Debugger** option) or only connect if an error occurs (the **Run External Debugger on Error** option).
- Logging Level The lower drop-down list controls the Verbosity Level.

The Tools section provides the following options:

- About this displays information about the running instance of Rapise, including the version number.
- Convert to NeoLoad this button opens the <u>NeoLoad conversion dialog</u>. This lets you convert a Rapise test script into a <u>NeoLoad performance test</u>.
- Activation this button opens the Rapise license activation screen. This can be used to deactivate the current license so that it can be used on a different machine.
- Spira Properties this button opens the <u>Spira Properties dialog</u> that allows you to see the name of the SpiraTest project and test case that the current Rapise test is linked to.
- **Object Manager** this button opens the <u>Object Manager</u> add-in; this add-in is used to copy recorded objects between test scripts.

• Libraries - this button opens the <u>Select an Application Dialog</u> so that you can add/change the <u>libraries</u> being used to record the current test.



The Help section provides access to the interactive help system as well as the **search and replace** tools.

- Find in Files clicking on this button opens the Find and Replace Dialog.
- Help this button provides access to the interactive help system. You can also bring up the help system by pressing F1 on the keyboard

2.4.29 Ribbon: Report

Scree	ensho	t									
File	Test	Options	Report								
C:\Users\	adam.sa	ndman\Docu	uments 🝷 🕄		PDF		- 🖻 🗙		🛗 Merge Cells	na	🗲 Collapse
镧 Plain	📲 Hiera	archical		Export to Excel	Export to PDF	Save Layout		Choose Columns		Images	⊞ ∉ Expand
		File		Exp	port	Layout			Data	I	Display

Purpose

The Report tab is for use with report (trp) files.

How to Open

The Report tab is available anytime you have a report (trp) file visible in the Content View.

C:\5	Subvers	ion\Projects\Rapise\	Trunk\Hel *	2
[2]	Plain	Hierarchical		3

- The drop-down menu contains a history of previously opened reports.
- Press Plain to view test steps, assertions, and messages aligned in a table.
- Press Hierarchical to more clearly see what assertions, messages, and data are associated with which test steps.





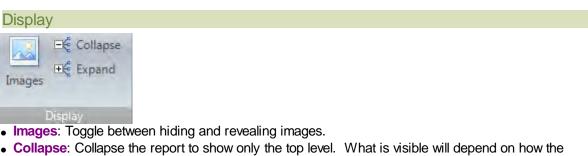
- Press Export to Excel to save the report as an excel file.
- Press Export to PDF to save the report as an Acrobat PDF file.

Layout	
	- 🗁 🗙
🚽 Save Layout	
S Reset Layout	
Layout	

- The drop-down menu lets you choose between previously saved layouts.
- You must press Save Layout to keep your layout changes after closing Rapise.
- Press Reset Layout to undo any changes you've made.

Data	
Choose Columns	🛄 Merge Cells
	Data

- Press Choose Columns to hide or reveal report columns.
- Merge Cells: Merge identical consecutive cells.



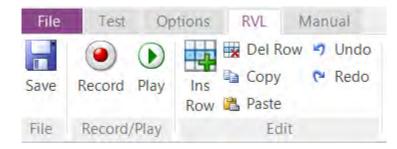
- Collapse: Collapse the report to show only the top level. What is visible will report is sorted.
- Expand: Expand all report rows.

See Also

<u>Automated Reporting</u>

2.4.30 Ribbon: RVL

Screenshot

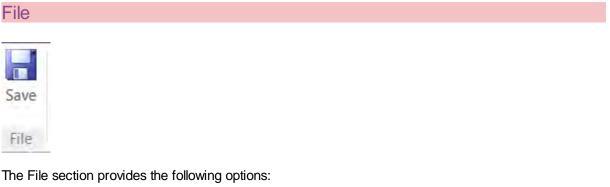


Purpose

The Rapise Visual Language (RVL) ribbon lets you edit the steps in a test script written using the <u>Rapise</u> <u>Visual Language</u>. Some of these operations are also repeated in the main <u>Test ribbon</u> when editing a test script that uses RVL.

How to Open

You can open the **RVL** ribbon by either clicking on the **RVL** icon on the main <u>Test ribbon</u> or clicking on the **Test.rvl.xlsx** file in the <u>Test Files</u> tab.



• Save - saves the current test locally. To save to a different location, use the Save As option in the File menu.

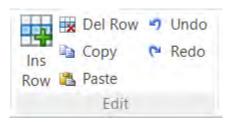
Record/Play Record Play Record/Play

The Record and Plan section provides tools relating to recording tests, learning objects, spying objects in the running application and executing the test.

- **Record** this button is used for **recording and learning**, clicking it will open the <u>Recording Activity</u> <u>Dialog</u>.
- Play this button executes the current test. You can change which test script to open in the Settings

Dialog.

Edit



- Ins Row This inserts a new, blank RVL row
- Del Row this deletes the currently selected RVL rows
- Copy this copies the selected RVL rows to the clipboard
- Paste this pastes the RVL rows copied to the clipboard
- Undo this undoes the last operation performed
- Redo this performs the last operation that was undone, using Undo

2.4.31 Ribbon: Spreadsheet

Screensl	hot		
File	Test	Options	Spreadsheet
Open Externally			
File	/		

Purpose

The Spreadsheet tab is for use with excel (xls) files.

How to Open

The Spreadsheet tab is available anytime you have an excel (xls) file visible in the Content View.



• **Open Externally** - this option lets you open the Excel spreadsheet inside the full Excel application instead of the embedded Rapise spreadsheet editor.

2.4.32 Ribbon: Edit

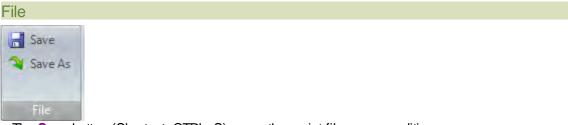
Screenshot				
File T	est Options	Edit		
🛃 Save 🌂 Save As	Paste & Cut	UndoRedo	Courier New + 10 +	Toggle Breakpoint
File	Clipboard	History	Font	Debug

Purpose

The Edit tab of the Ribbon provides tools for editing script files.

How to Open

The Edit tab is available anytime you have a javascript file visible in the Content View.



- The Save button (Shortcut: CTRL+S) saves the script file you are editing.
- The **Save As** button allows you to create a new, differently named copy of the script file you are editing.

Clipboard



- The Paste button (Shortcut: CTRL+V) pastes from the clipboard.
- The Cut button (Shortcut: CTRL+X) erases whatever text you have highlighted, and copies it to the clipboard.
- The Copy button (Shortcut: CTRL+C) copies whatever text you have highlighted to the clipboard.

History

5	Undo
6	Redo
H	istory

- The Undo button (CTRL+Z) reverses the last deletion or insertion made in the Source Editor.
- The Redo button (CTRL+Y) reverses the last undo action.

Font	
Courier New	- 10 -
Font	

• Use the above font and size drop-down menus to change the text appearance. The entire file will be affected.

Debug	
Toggle Breakpoint	
Debug Press the Toggle Br	eakpoint button (Shortcut: F9) to insert or remove a breakpoint at the current

• Press the **Toggle Breakpoint** button (Shortcut: F9) to insert or remove a breakpoint at the current cursor position.

2.4.33 Ribbon: Debugger

Screensl	hot		
File	Test	Edit	Debug
Run	(,≣	Step Over	.
🗺 Step Ir	n 🗾	Stop Deb	ugger
👌 Step O	Dut 🤭	Break	
	Debu	gger	

Purpose

The Debugger Tab provides tools for use with the Internal Debugger.

How to Open

The **Debugger Tab** is available while the **Internal Debugger** is being used. To use the Internal Debugger, first enable it, then <u>Playback</u> your script. Instructions for enabling the Internal Debugger are <u>HERE</u>.

Debugger

- Run (F5): Continue executing the script.
- Step In (F11): Step into a function/procedure.
- Step Out (Shift+F11): Continue until the current procedure is exited.
- Step Over (F10): Go to the next line in the current procedure/function.
- Stop Debugger (Shift+F5): Stop executing the script and exit the debugger.
- Break (F9): Create a breakpoint in the script at the cursor.

2.4.34 Ribbon: Manual

Screenshot				
File Test M	lanual			
Save Reload	Add Step Remove Step	 View in Browser New Incident 	Release Execute Manual	Record Screenshots Whole Window Record Manual With Cursor
Spira	Editor	External Tools	Playback	Recording

Purpose

The **Manual** ribbon lets you record, edit and play <u>manual tests</u> that have either been created in Rapise or have been downloaded from <u>Spira</u>. Rapise provides powerful **exploratory testing functionality** that lets you rapidly create manual tests by simply clicking through the application rather than having to laboriously create test steps one at a time by hand.

These <u>manual tests</u> can then be either <u>executed from within Rapise</u> or saved to Spira so that they can be executed by any tester that has access to the Spira web interface. In addition, these manual steps can be used as the basis for test automation by linking specific <u>test scenarios</u> to manual test steps.

How to Open

You can open the **Manual** ribbon by either clicking on the **Manual Steps** icon on the main <u>Test ribbon</u> or clicking on the **ManualSteps.rmt** file in the <u>Test Files</u> tab.

Spira Reload Save Spira

- The main Save icon will save the current test to Spira, both the manual test steps and any automated testing files.
- The Reload icon will refresh the current test from the copy helds in Spira.
- The **Save Local** will save the manual test steps and any open automation files locally. You can use this to save files before doing a batch upload to Spira.

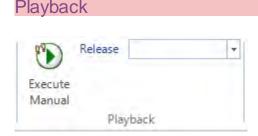
Editor



- The Add Step icon will add a new test step to the current manual test case displayed in the manual test editor.
- The Remove Step icon will remove the highlighted test step from the current manual test
- The \hat{T} icon will move the highlighted test step one position higher in the current manual test
- The $\[mathcal{P}\]$ icon will move the highlighted test step **one position lower** in the current manual test

External Tools			
View in BrowserNew Incident			
External Tools			

- The View in Browser icon will display the current manual test inside the Spira web interface
- The **New Incident** icon will open the <u>Incident Logging</u> dialog box so that you can log a new incident in Spira.



- The **Execute Manual** icon will execute the current manual test. When you click the Execute Manual icon, you will be asked to save the test case to Spira, then the latest version from Spira will be downloaded into the Rapise <u>manual test execution wizard</u> so that you can start manual testing.
- The **Release** dropdown list displays the list of releases in the current Spira project:

er 🕅 Release	-
1.0.0.0 - Library System Release 1	
1.0.1.0 - Library System Release 1 SP1	
1.0.1.0.0001 - Iteration 001	
1.0.1.0.0002 - Iteration 002	
1.0.1.0.0003 - Iteration 003	
1.0.2.0 - Library System Release 1 SP2	
1.0.2.0.0001 - Iteration 001	
1.0.2.0.0002 - Iteration 002	Ŧ

You can then choose the appropriate release that the current test is being executed against.

Recordii	ng	
Record Manual	 Record Screenshots Whole Window With Cursor 	
	Recording	

- The Record Manual icon will start the <u>Select Application to Record</u> dialog box. This dialog box is the same one that you'll use for automated testing, however when you click through the application under test it will record <u>manual test steps</u> instead of automated script code.
- The **Record Screenshots** option will tell Rapise to capture the current screenshot when performing manual recording and include the screenshot with the recorded test step. These are two sub-options:
 - Record Whole Window When checked, this will record the entire window. Warning, this may take up large amounts of disk space. Otherwise it will record just the object underneath the current cursor.
 - \circ Record Cursor This will record the location of the mouse pointer/cursor inside the image.

2.4.35 Ribbon: REST



Purpose

The **REST** tab is for use with editing <u>REST web service</u> definition files.

How to Open

The **REST** tab is available anytime you have a REST definition file (.rest) file visible in the <u>Content</u> <u>View</u>.

ile			
H	~		
Save	Update		
Requests	Object Tree		
	File		

This turns each of your REST requests into Rapise learned objects that can be scripted against.

Edit		
-	{-}	
Add	Add	Add
Header	Parameter	Credentials
	Edit	

The Edit section of the REST ribbon lets you edit the HTTP headers, parameters and credentials associated with the current request.

• Add Header - Allows you to add a standard or custom HTTP header to the current REST request:

Name:	Accept	
Nume.	Люсері	
Value:	application/json	(•

Add Parameter - Allows you to add a parameter name/value to the current REST request. This is
useful when you want your test script to be able to pass through different values (e.g. get book #1
vs. book #2):

Parame	ters are used in the Request URL to genericize the REST request:
Vame:	session_id
Value:	82499bcc-37e4-4c64-820e-a2d798cd1e84

 Add Credentials - Allows you to add an HTTP basic authentication credential (username and password) to the request. Saves you having to add the header manually (which would require base64 encoding the username and password):

Usemame:	librarian	
Password:		
		Add Cancel

Script Steps

Get_Session GET		Create Script			
*Get_Authors GE	*Get_Authors GET				
		Clean			
	Script Steps	_			

The Script Steps section of the REST ribbon lets you see the list of recorded REST operations and use them to create your <u>test script</u> in the main test file.

- Create Script This takes all of the recorded steps and automatically creates the matching JavaScript code in your test script.
- Remote Step This removes the recorded step from the script box
- Clean This removes all of the recorded steps from the script box.

Each of the steps displayed in the script box will contain the name of the REST operation along with its HTTP method (GET, POST, PUT, DELETE, etc.). Steps displayed in **bold*** with an asterisk also have a <u>verification point recorded</u>. That means when the script is generated, it will include a **Tester.Assert** function to verify the results.

2.4.36 Ribbon: SOAP

Scr	eenshot		
File	Test Options SOAP		
	http://www.libraryinformationsystem.org/Services Get WSDL	🛅 Create Script	- 19
Save	Custom Endpoint http://www.libraryinformationsystem.org/Services	🐰 Remove Step	₩ Add
Sare		× Clean	Credentials
File	Endpoint	Script Steps	HTTP

Purpose

The **SOAP** tab is for use with editing <u>SOAP web service</u> definition files (also known as Web Service Definition Language (WSDL) files).

How to Open

The **SOAP** tab is available anytime you have a SOAP definition file (.soap) file visible in the <u>Content</u> <u>View</u>.

File	
Save	
File	
0	Occurs the comparison of COAD an another to the constant file basis and its d

• Save - Saves the current set of SOAP operations to the .soap file being edited.

Endpoint		
http://www.libraryi	informationsystem.org/Services	Get WSDL
Custom Endpoint	http://www.libraryinformationsy	stem.org/Services
	Endpoint	

The Endpoint section lets you enter in the URL to the SOAP Web Service Definition Language (WSDL) file that contains a definition of all the SOAP operations exposed by the web service.

You enter in the URL in the top box and then click the **Get WSDL** button.

You can click on the **Custom Endpoint** button to toggle the display of the custom endpoint text box. This lets you override the default URL returned by the WSDL file and can be useful if you want to use the WSDL from one instance and invoke the operations against a different instance.

Script Steps	
Connection_Authenticate {"userName":"librarian","password":"librarian"}	Create ScriptRemove StepClean
Script Steps	

The Script Steps section of the SOAP ribbon lets you see the list of recorded SOAP operations and use them to create your <u>test script</u> in the main test file.

- Create Script This takes all of the recorded steps and automatically creates the matching JavaScript code in your test script.
- Remote Step This removes the recorded step from the script box
- Clean This removes all of the recorded steps from the script box.

Each of the steps displayed in the script box will contain the name of the SOAP operation along with the specified parameters in JSON format. Steps displayed in **bold** with an asterisk also have a <u>verification</u> <u>point recorded</u>. That means when the script is generated, it will include **Tester.Assert** functions to verify the results.

HTTP	
*	
Add	
Credentials	
HTTP	
Add Creder	ntials - Allows you to add an HTTP basic authentication credential (username and

 Add Credentials - Allows you to add an HTTP basic authentication credential (username and password) to the SOAP operation. This is useful for SOAP operations that use HTTP basic authentication.

d Network	Credential	×
Usemame:	librarian	•
Password:		
		Add Cancel

2.4.37 Ribbon: Options

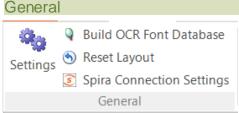
Screenshot							
File	Test	Options					
🙀 🍳 Build OCR Font Datab			Database	sé	Selenium Settings	4	Java Settings
	🕤 Reset Layout		Ö	Web Spy Settings	Mobile Settings		
	🖲 Spira	a Connection	n Settings	٩	Browser Settings	G	NeoLoad Integration Settings
General				Web Testing		Other Testing	

Purpose

The **Options** tab provides access to all the global settings and options in Rapise. Settings relevant to just the current test are available in the <u>Settings</u> pane.



The **Options** tab is always available in the ribbon.



The General section provides the following options:

- Settings This brings up the <u>Global Settings dialog</u> that lets you change any of the system-wide settings for Rapise.
- Build OCR Font Database Pressing the Build OCR Font Database button updates the list of screen fonts that Rapise recognizes when using an OCR object. Whenever you install new Fonts onto the computer you should click this button to have then added to the Rapise font database.
- Reset Layout Pressing the Reset Layout button restores the default layout. Rapise will restart.
- Spira Connection Settings Pressing the Spira Connection Settings button takes you to a dialog box that lets you change how Rapise is integrated with the <u>SpiraTest</u> test management system. It will let you change the URL, username and password used to connect.

Web Testing

- 📓 Selenium Settings
- Web Spy Settings
- Browser Settings

Web Testing

The Web Testing section provides links to the different settings that can be changed for web testing:

- Selenium Settings This button displays the <u>Selenium settings</u> dialog box. This is used to edit the different Selenium web browser profiles that can be used by Rapise.
- Web Spy Settings This button displays the <u>Web Spy Settings</u> dialog box. This lets you change the settings related to using the Web Spy to inspect the DOM objects in web pages.
- Browser Settings This button displays the <u>Browser Settings</u> dialog box. This lets you edit and select the web browser profile being used for web testing (for non-Selenium browser profiles)

Other Testing

- 🛓 Java Settings
- Mobile Settings
- NeoLoad Integration Settings

Other Testing

The Other Testing section provides tools relating to testing non-web applications, including <u>Java</u> <u>applications</u>, <u>mobile device apps</u> and <u>load-testing with NeoLoad</u>.

- Java Settings this button displays the <u>Install Java Access Bridge</u> dialog box. Installing the Java Access Bridge lets Rapise connect to Java AWT/Swing applications so that they can be tested.
- **Mobile Settings** this button displays the <u>Mobile Settings</u> dialog box. This lets you configure the different mobile devices that are available for testing by Rapise.

• **NeoLoad Integration Settings** - this button displays the <u>NeoLoad Integration Settings</u> dialog. These settings may need to be changed when using Rapise with NeoLoad.

2.4.38 Scripting Choice Dialog

Screenshot							
R Scripting Language	_		×				
Choose preferred scripting langua	ge:						
RVL (Rapise Visual Language)							
JavaScript (Advanced Users)							
Don't ask (may be changed later in Options\Settings\Default Scripting Language)							

Purpose

You use the **Choose Scripting Language** dialog to choose between writing your tests in the <u>Rapise</u> <u>Visual Language (RVL)</u> scriptless table language, or in the <u>JavaScript script editor</u>.

How to Open

This dialog is automatically opened after you click Create in the Create New Test dialog.

Misc

If you click the "Don't Ask' checkbox, Rapise will default to your last selection automatically when you create a test. You can change this options in the main <u>Settings dialog</u>.

2.4.39 Select an Application to Record... Dialog

Screenshot

Select From Running Applications Run A	Application	
Inflectra Library Information System	stem - Window	s Internet Explorer (C:\Program Files\Internet
Applications		
vailable Applications	PID	Path
Adobe Photoshop CS3 Extended - [Untitle C:\Program Files\DellTPad\Apntex.exe HelpAndSamples - Microsoft Visual Studio	5336 (47172	C:\Program Files (x86)\Microsoft Office \Offic C:\Windows\SysWOW64\rundll32.exe C:\Windows\SysWOW64\rundll32.exe C:\Program Files (x86)\Adobe \Adobe Photo C:\Program Files \DellTPad\ApntEx.exe C:\Program Files (x86)\Microsoft Visual Stud
Html Sinilectra (Library Information System - Wir	5048 rd 62680	C:\Windows\explorer.exe C:\Program Files \Internet Explorer\iexplore a
Menu Msg TdmNotify Tutorial: Record and Playback - Mozilla F Windows Media Player X-PRO Vonage	644 63624 5204 ire 48244 50832 644	C:\Program Files (x86)\X-PRO Vonage\X-PF C:\Windows\SysWOW64\rundll32.exe C:\Program Files\Wave Systems Corp\Trust C:\Program Files (x86)\Mozilla Firefox\firefox C:\Program Files (x86)\Windows Media Play C:\Program Files (x86)\X-PRO Vonage\X-PF
Libraries	III	,
Cancel	(m. e.	resh Select

Purpose

The **Select an Application to Record**... (SAR) Dialog appears before <u>Recording</u> takes place when testing **desktop applications**.

It queries the user for which program to record, as well as what <u>Recording Library</u> to use.

If you are recording the same application for the second time then SAR is not shown. The recording proceeds to last used application if it is still available on the screen.

How To Open

To open the SAR Dialog, press the **Record/Learn** button on the Ribbon (**Test** tab > **Recording & Learning** menu):



Libraries

Library	Description
Auto	Detect library automatically
NET	.NET 1.1, 2.0, 3.0, 3.5 with Accessility
Internet Explorer HTML	HTML DOM-based recorder for Internet Explorer
Firefox HTML	HTML DOM-based recorder for Mozilla Firefox
🔲 Generic	Generic library contains basic definitions for most commo

The **Library** table lists the available Recording Libraries. Select the one appropriate to the process/ program you will record. If you select **Auto**, Rapise will attempt to choose the correct recording library for you. See the <u>Recording Library</u> section for more information.

Available Applications

Available Applications	PID	Path
	7032	C:\Windows\explorer.exe
6	3744	C:\Program Files\Google\GoogleToolbarN
🔤 C:\Windows\system32\cmd.exe	4796	C:\Windows\system32\cmd.exe
🍪 Help & Manual	7024	C:\Program Files\EC Software\HelpAndM
🚉 Program Manager	7032	C:\Windows\explorer.exe
Sample ATM Login - Windows Internet Ex	7000	C:\Program Files\Internet Explorer\iexplor

The **Available Applications** table lists all of the processes running at the time you open the **SAR dialog**. If the process you would like to record is already open, you can select if from the table. Pick the appropriate recording library (above) first before you pick an application to record; your application choice will become unselected if you do not do it last.

Widgets			
Cancel	🗌 Show All 🛛 🛃 F	Refresh	Select

- The Cancel button closes the dialog.
- Show All: While unchecked only top level application windows reflected in the Windows Task Bar are shown in the 'Available Applications' list. Check this and press Refresh to see all top level windows available on the screen.
- **Refresh List**: Press to refresh the Available Applications table. After refreshing, you will see processes that began after the SAR dialog was opened.
- Select button: To record a process from the Available Applications table, select the process and then press the Select button.

Run Application Tab

Select an Application to Record	6	
Select From Running Applications Run Application		
Full path to application:		
C:\Users\Public\Documents\Rapise\Samples\SampleATM\SampleATM.exe	-	Browse
(e.g. c:\windows\system32\calc.exe)		
Use working directory:		
Run		
Cancel		

- Path drop down list: If the program you would like to record is not already open, you can specify its path here. If the program is already running, you can select it from the Available Applications table.
- Browse button: Browse for an application to open and record.
- Use working directory: To set a specific working directory when launching the application, check the box and enter in a value for the working directory.
- Run button: To record a program that is not currently open, fill in the Path text-box and press the Run button.
- The Cancel button closes the dialog.

2.4.40 Selenium Settings Dialog

Purpose

This dialog box displays the list of mobile devices that have been configured for use by Rapise and lets you create a new profile, modify a profile or make a new profile based on an existing one.

Screenshot

ele	enium Settings	-	
Se	elect Profile: Safari v S	ave Test Duplicate Create Rename	Delete
_	Connection		^
	Uri	http://192.168.0.1:4444/wd/hub	
v			
v		(3)	
	browserName	Safari	
	platform		
	version		
>	Chrome Specific	(9)	
v	Common Capabilities	(11)	
	acceptSslCerts		
	applicationCacheEnabled		
	browserConnectionEnabled		
	databaseEnabled		
	elementScrollBehavior		
	javascriptEnabled		
	locationContextEnabled		~
	nativeEvents		*
U	nativeEvents ri RL of Selenium server. Leave blank to automate a local browser.		
			Close

How to Open

You can open this dialog box from the Rapise <u>Options ribbon</u>, it is the **Selenium Settings** button in the **Web Testing** tab.

Menu Options

This dialog box has the following menu options:

- Select Profile This dropdown list lets you select a different Selenium profile to be displayed in the dialog.
- Save This button will save the changes to the current Selenium profile.
- **Test** This button will test the Connection (URL) from Rapise to <u>Selenium WebDriver</u> (which is used to connect to the web browsers) and the connection from Selenium to the web browser.
- Duplicate This button will create a new Selenium profile based on the currently viewed one.
- Create This button will create a new empty Selenium profile that you can edit.
- Rename This button will change the name of the current Selenium profile being edited.
- Delete This button will delete the currently displayed Selenium profile. There is no undo, so be careful!

Connection

This section lets you enter the URI used to connect to the <u>Selenium WebDriver</u> server which hosts the web browsers being tested. It is typically of the form:

• http://server:4444/wd/hub

Where the port number used by Selenium is 4444 by default and the /wd/hub suffix is added.

If you are using web browsers on the local machine (that is running Rapise) you don't need to enter in a URI and can leave the entire field blank.

Details

This section has various settings, some of which are used by all web browsers, some are browser specific:

- Primary Capabilities
 - o browserName The name of the browser being automated. The current allowed values are:
 - Internet Explorer
 - Firefox
 - Chrome
 - Safari
 - Opera
 - MicrosoftEdge
 - platform A key specifying which platform the browser should be running on. This is used by Selenium Grid and/or cloud platforms such as SauceLabs to dynamically select the appropriate Selenium server. For other cases you can leave blank. The allowed values for Selenium Grid include:
 - WINDOWS
 - XP
 - VISTA
 - MAC
 - LINUX
 - UNIX
 - ANDROID
 - version The browser version to use. This is used by Selenium Grid and/or cloud platforms such as SauceLabs to dynamically select the appropriate Selenium server. For other cases you can leave blank.

Common Capabilities

- javascriptEnabled Whether the Selenium session supports executing user-supplied JavaScript in the context of the current page
- o databaseEnabled Whether the session can interact with database storage.
- \circ locationContextEnabled Whether the session can set and query the browser's location context.
- o **applicationCacheEnabled** Whether the session can interact with the application cache.
- browserConnectionEnabled Whether the session can query for the browser's connectivity and disable it if desired.
- o webStorageEnabled Whether the session supports interactions with storage objects.
- \circ <code>acceptSslCerts</code> Whether the session should accept all SSL certs by default.
- rotatable Whether the session can rotate the current page's current layout between portrait and landscape orientations (only applies to mobile platforms).
- nativeEvents Whether the session is capable of generating native events when simulating user input.
- unexpectedAlertBehaviour What the browser should do with an unhandled alert before throwing out the UnhandledAlertException. Possible values are 'accept', 'dismiss' and 'ignore'.
- elementScrollBehavior Allows the user to specify whether elements are scrolled into the viewport for interaction to align with the top (0) or bottom (1) of the viewport. The default value is to align with the top of the viewport. Supported in IE and Firefox (since 2.36).
- Chrome Specific this contains various settings specific to the Chrome web browser
- · Firefox Specific -this contains various settings specific to the Firefox web browser
- IE Specific this contains various settings specific to the IE web browser
- Safari Specific this contains various settings specific to the Safari web browser

- **Proxy** this contains various settings for using HTTP proxy servers when connecting from Rapise to Selenium WebDriver
- Remote WebDriver Specific

 webdriver.remote.quietExceptions Disable automatic screnshot capture on exceptions. This is False by default.

2.4.41 Settings Dialog

Screenshot	
Settings	д
₽ ↓ □	
Advanced	
CommandLine	
EntryPoint	Test
E TestParams	3 items
Execution	
E Methodology	
E Recording	
E Screen Capture	
Settings	
Metadata	C:\Temp\RapiseTests\CreateNe
ObjectsPath	CreateNewBook.objects.js
ReportPath	CreateNewBook.trp
ScriptPath	CreateNewBook.js
TestPath	C:\Temp\RapiseTests\CreateNe
UserFunctionsPath	CreateNewBook.user.js
Recording	

Purpose

Use the Settings Dialog to change test specific settings.

4

How to Open

The Settings dialog is part of the **Default Layout**.

Advanced

Ξ	Advanced	
	CommandLine	

• CommandLine is a freeform text box. Use it to specify values for global variables (beginning in g_) to pass the <u>recorder</u> and <u>player</u>. You can view which global variables are available in the source files (such as **Player.js**, **SeSCommon.js**, etc).

Execution

Ξ	Execution	
	CacheObjects	False
	CommandInterval	100
	IterationsCount	1
	ObjectLookupAttemptInterval	150
	ObjectLookupAttempts	1

- CacheObjects: Remember object locations and try to reuse them for speed. This is helpful with dialog based applications.
- CommandInterval: Time interval (in milliseconds) between script commands during script execution.
- IterationsCount: Your test script will be executed this many times consecutively during <u>Playback</u>.
- ObjectLookupAttemptInterval: This is the time Rapise will wait between attempts to locate an object.
- ObjectLookupAttempts: This is the number of times Rapise will attempt to locate an object.

Recording

E Recording

BeautifySavedObjects False

 BeautifySavedObjects affects how the <u>Script Recorder</u> writes object information to your test script. If False, the object definition will be written as a single line:

```
var saved_script_objects={
   Balance:{"version":0,"object_type":"SeSSimulated","object_name":"Transaction
Completed Successfully\n\nAccount 0000005
Balance:1046.00", "object_class": "Static", "object_role": "ROLE_SYSTEM_STATICTEXT", "object_
text":"Transaction Completed Successfully\n\nAccount 0000005
Balance:1046.00","locations":[{"locator_name":"Location","location":
{"location":"4.4.4","window_name":"SmarteATM","window_class":"#32770"}},
{"locator_name":"LocationPath","location":
{"window_name":"SmarteATM","window_class":"#32770","path":[{"object_name":"Transaction
Completed Successfully\n\nAccount 0000005
Balance:1046.00","object_class":"Static","object_role":"ROLE_SYSTEM_STATICTEXT"},
{ "object_name": "Transaction Completed Successfully\n\nAccount 0000005
Balance:1046.00", "object_class":"Static", "object_role":"ROLE_SYSTEM_WINDOW"},
{"object_name":"SmarteATM","object_class":"#32770","object_role":"ROLE_SYSTEM_DIALOG"}]}
}]}
};
```

If **True**, the object definition will be written in a manner that takes more space, but is easier to read and change:

```
var saved_script_objects={
  Balance:{
   "version": 0,
   "object_type": "SeSSimulated",
   "object_name": "Transaction Completed Successfully\n\nAccount
   00000005 Balance:1046.00",
   "object_class": "Static",
   "object_role": "ROLE_SYSTEM_STATICTEXT",
   "object_text": "Transaction Completed Successfully\n\nAccount
     00000005 Balance:1046.00",
   "locations": [
      {
         "locator_name": "Location",
         "location": {
            "location": "4.4.4",
            "window_name": "SmarteATM",
```

© 2017 Inflectra Corporation

	"window_class": "#32770"
},	
{	
	//section omitted for brevity
}	
]	
}	
};	

Objects that were learned in previous recordings are affected by the value of BeautifySavedObjects.

Screen Capture

Screen Capture	
Capture Execution	False
Capture Recording	False
Include in Report	False
Widget Only	False

- Capture Execution: Set this to True if you want to save screen images for each recognized object during playback.
- Capture Recording: Set this to True if you want to save screen images for each action during recording.
- Include in Report: Set this to True to include the saved images in the execution report during playback.
- Widget Only: Set this to True to only save the widget area in the screenshot, as opposed to the whole window.

TestParams

The TestParams section includes various custom test parameters:

Click to open the TestParams Collection Editor Dialog.

There is a build-in set of test parameters for <u>cross-browser testing</u>. When you open up a test that uses one of the HTML libraries it will display the following built-in test parameter that you can use to change the **playback browser**:

Sett	ings		
]2↓ □		
۵	Advanced		
	CommandLine		
	EntryPoint	Test	
⊿	TestParams	5 items	=
	Browser	Internet Explorer HTML	
	Record Title	Internet Explorer HTML	
	ServerProjectId	Firefox HTML	
	ServerTestCaseFolde	Chrome HTML	
	ServerTestCaseId	6	
⊿	Execution		
	CacheObjects	False	
	CommandInterval	1000	
	Iteration Count	1	
	ObjectLookup Attemptiote	200	÷.

Settings

⊡	Settings		_
	Metadata	C:\Users\Public\Documents\Shared	
	ObjectsPath	TwoDialogsTest.objects.js	
	ReportPath	TwoDialogsTest.trp	Ξ
	ScriptPath	TwoDialogsTest.js	
	TestPath	C:\Users\Public\Documents\Shared	
	UserFunctionsPath	TwoDialogsTest.user.js	_
			Ψ.

- UserFunctionsPath: Path (relative to the test directory) to the file with user-defined functions utilized in this test. Normally this file has name in form *.user.js.
- ObjectsPath: Path (relative to the test directory) to file containing object tree information. This file contains saved_script_objects structure with all object locators gathered during recording and learning. Normally this file has name in form *.objects.js.
- **ReportPath**: Path (relative to the test directory) to the test's report file. Normally this file has extension form .trp which stands for Test Report.
- ScriptPath: Path (relative to the test directory) to the test script.
- TestPath: Path to the test definition file (*.sstest).

2.4.42 SOAP Definition Editor

SOAP		Start Page - WebServicesSOAP.son							
✓ ⊴ WebServicesSOAP		Invoke RequestResponse							
→ BasicHttpBinding_ISoapService		Participation of the participation of the second seco							
Author_Retrieve		Input	Volue						
Author_RetrieveById	Book_Delete Book_Insert	a a took_RetreeveById	Value	True					
		() Headers	Туре	System.Boolean					
		 O Body 							
Book Retrieve Book Retrieve Book RetrieveByDateRange Book RetrieveById Gook Update		III Int32 id = 15 ☑ Doolean idSpecified = True:							
					W Book_UploadImage				
					Connection_Authenticate				
Connection Disconnect									
 BasicHttpBinding_ISospService1 									
Author_Retrieve									
Author_RetrieveById									
Sonk_Delete		Output	Value Invoke						
· Book_Insert									
Book_Retrieve									
Book_RetrieveByDateRange Book_RetrieveById	*								
Objects Files SOAP Settings	-								
Properties									
12 24 I III									
		Response Body							
		Auto Raw XML JSON							
		<pre>1 <?xml version="1.0" encoding="utf-0"?> 2 = <s:envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"> 3 = <</s:envelope></pre>							
			an Clients (fault code)						
		<pre>s <faultcode>s:Client</faultcode> <faultstring xml:lang="en-US">Unauthorized, call Connection Authenticate first</faultstring>Unauthorized, call Connection Authenticate firstUnauthorized, call Connection Authenticate first</pre>							
			g="en=us">unauthorized, call connection_Authenticate first <td>aurestring></td>	aurestring>					
		<							
		Response Body Response Headers Output War	nings Errors Find Resolts						

Purpose

The **SOAP Definition Editor** allows you to edit <u>SOAP web service</u> definition files (.soap) that contain data downloaded from SOAP Web Service Definition Language (WSDL) URLs.

How to Open

Use the <u>Add Web Service Dialog</u> to create a new SOAP definition (.soap) file. The definition file will be opened in a **SOAP Editor** in the <u>Content View</u>. The <u>SOAP Tab</u> of the Ribbon will also open.

File	Test	Options	SOAP					
	http://www.libraryinformationsystem.org/Services Get WSDL			stem.org/Services	Get WSDL		🛅 Create Script	- 49
Save	Custom Endpoint http://www.libraryinformationsystem.org/Services			libraryinformationsy	stem.org/Services		👗 Remove Step	⊯ Add
							🗙 Clean	Credentials
File	Endpoint			ndpoint		Script Steps		HTTP

Or, you can double-click on an existing .soap file in the <u>Test Files View</u> explorer window. The definition file will be opened in a **SOAP Editor** in the <u>Content View</u>. The <u>SOAP Tab</u> of the Ribbon will also open.

Start Page WebServicesSOAP.soo			
Input	Value		
 ♦ • ♦ ⊕ Consection_Authenticate (1) Headers (2) Body (4) Body (4)	Value Istual Type	librarian False System String	
Сари	Value Invoke Record		
	Value Type	True Systam Boolean	

The invoke tab lets you visually choose a specific SOAP operation and execute it. It has the following sections:

- Input You can expand the various SOAP operations and see the input headers and body parameters that need to be sent to the function. You can click on each parameter and supply a value on the right-hand pane.
- Output You can expand the various SOAP operations and see the output headers and result that should be returned from the operation (if successful). You can click on each header or the body and see the type of data returned.
- Invoke Once you have supplied the appropriate values, click the Invoke button send the SOAP request and get the data back from the request.
- Record Clicking this button after a successful invoke of the operation will add it to the list of recorded test scripts shown in the <u>SOAP ribbon</u>.
- **Verify** Clicking this button after the Record will add a Tester.Assert(...) <u>verification</u> <u>checkpoint</u> to the recorded test script. This will make Rapise automatically verify all of the returned values.

Request / Response

This tab displays the raw SOAP XML request and response. When a SOAP operation fails, this is useful when debugging since it lets you see the raw data being sent to the web service:

	ebServicesSOAP.son			
RequireProof		Request		
AllowAutoRedire AllowWriteStrear BasicAuthPastw BasicAuthUserN ContentType HttpProxy	False	Auto [Raw] XML ISON xml version="1.0" encoding="utf-8"? <soap:envelope http:="" www.inflectra.c<br="" xmlns:soap="http://schemas.xmlsoap.org/soap/en</td><td>nvelope/" xmlns:xsi="http://</td></tr><tr><td>Method
Pipelined
PreAuthenticate
SendChunked</td><td>False</td><td>3</td><td></td></tr><tr><td>SOAPAction
Timeout</td><td>">100000</soap:envelope>	Response	Send
Und http://www.librar UseCookeCants True UseCookeCants True		ResponseCode: 200 (OK) Content-Length 296 Cache-Control private Content-Type:text/xml; charset=utf-8 Date: Thu, 22 Dec 2016 20:59:14 GMT Server/Microsoft-IS/8.0 X-AspNet-Version 4.0.30319 X-Powered-By ASP.NET <sienvelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:body></s:body></sienvelope>		

Typically you will want to view this information in either **Raw** or **XML** modes since SOAP doesn't support JSON as a serialization format.

Start Page ww W	/ebServicesSOAP.soa			
roke Request/R	Response			
RequestPrope AllowAutoRedire AllowAutoRedire BasicAuthPassw BasicAuthUserN ContentType HttpProxy KeepAlive Method	False False	Request Auto Raw XML ISON 1 «7xml version="1.0" encoding="utf=0"?> 2 «soap(XML]tope xmlns:soap="http://schemas.xmlsoa 3 «soap:Body> 4 «connection Authenticate xmlns="http://sww.i 5 «userNamo>libratianc/userNamo> 6 «password>libratianc/password>	p.org/soap/envelope/" xmlns:xsi="http://www.w3.o nflectra.com/LibraryInformationSystem/Services/"	
PreAuthenticate		<pre>7 </pre>		
SendChunked	False	<		5
SOAPAction Timeout	"http://www.inflectra.c 100000	Response		Send
Ud UseCookieConta UseDefoultCrede	False	ResponseCode: 200 (OK) Content-Length:296 Cache-Control'private Content-Type text1xml; charset=utf=8 Date:Thu, 22 Dec 2016 20:59:14 GMT Server:Microsoft=IIS/8.0 X-ApNet-Version: 4.0.30319 X-Powered-By:ASP.NET <s envelope="" xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">-</s>	s Body>	

Response Body

Response Headers				
Name				
ResponseCode	OK (OK)			
Content-Length	296			
Cache-Control	private			
Content-Type	text/xml; charset=utf-8			
Date	Thu, 22 Dec 2016 20:59:14 GMT			
Server	Microsoft-IIS/8.0			
X-AspNet-Version	4.0.30319			
Response Body Response Headers Output	Warnings Errors Find Results			

The HTTP Response in SOAP XML format is formatted and displayed:



This displays the output from the last web service request. It has several tabs:

- Response Header Displays a list of the HTTP response headers (name and value). If the request
 received a 200 OK code back, it's displayed in green, if it receives an error code back, it's displayed
 in red.
- Response Body
 - o Raw Displays the raw text of the HTTP response body received from the server.
 - XML If the received body content is identified as XML, this tab displays nicely formatted XML that is easier to read than the raw response body.

Operation Explorer

SOAP		
▲ 🔄 WebServicesSOAP		
BasicHttpBinding_ISoapService		
=le Author_Retrieve		
author_RetrieveById		
=♦ Book_Delete		
=♦ Book_Insert		
= Book_Retrieve		
Book_RetrieveByDateRange		
all a straight and a straight		
= Book_Update		
=♦ Book_UploadImage		
Connection_Authenticate		
Connection_Disconnect		
A - BasicHttpBinding_ISoapService1		
author_Retrieve		
author_RetrieveById		
=♦ Book_Delete		
=le Book_Insert		
all a straight and a straight		
Book_RetrieveByDateRange		
= Book_RetrieveById		
Objects Files SOAP Settings		

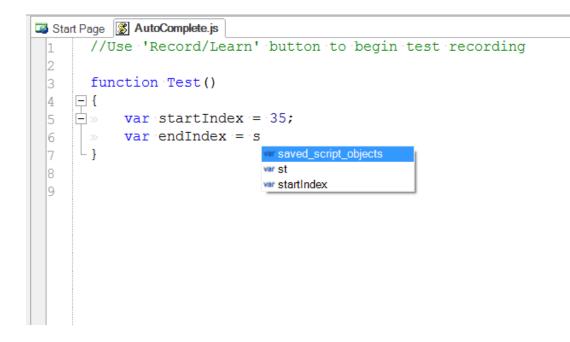
This section lets you see all of the SOAP web service endpoints in the current WSDL file and view the individual operations.



- For more info on SOAP Web Services, see <u>SOAP Web Services</u>.
- o For a tutorial on creating a SOAP web service test, see the Web Services SOAP Tutorial.

2.4.43 Source Editor

Screenshot



Purpose

To display and edit javascript files. The editor supports <u>Syntax Highlighting</u>, <u>Syntax Checking</u>, <u>Code</u> <u>Folding</u> and <u>Code Completion</u>.

How to Open

Use the <u>Test Files Dialog</u> to open a javascript file. The javascript file will be opened in a **Source Editor**, in the <u>Content View</u>. The <u>Edit Tab</u> of the Ribbon will also open.

2.4.44 Spreadsheet Viewer

Screenshot

Start Page Calc.xls Calc Data +Add New				
	А	В	С	D
1)	Item1	Operation	Item2	Result
2	15	+	13	28
3	5	*	6	30
4	19	-	3	16
5	8	1	4	2
*				

Purpose

To display excel (xls) files.

How to Open

Use the <u>Test Files Dialog</u> to open an excel file. The excel file will be opened in a **Spreadsheet Viewer**, in the <u>Content View</u>. The <u>Spreadsheet</u> tab of the Ribbon will also open.

2.4.45 Start Page

Screenshot		
🗃 Start Page 🏽 Web Test.js		× ×
SpiraTeam Open Spira Dashboard Use the dashboard to link tests to test cases, create and run test sets, view automation hosts and more.	Current Test Name: Web Test Scenario: Test Description User Functions: Web Test user is Script: Web Test user is Parameters Spira Properties Browser: Chrome HTML Record Title: regerc.*	
CreateNewBook EditExistingBook EditExistingBook EditExistingBook EarPoint HTML5 Java JavaSWT Java JavaSWT Joury-UI Managed JourgeATM SampleATM SampleATM SampleATM SinulatedObject UsingCustomStrings	Start Here Your Rapise Tutorial show: □ Recent Tests Start Here Your Rapise Tutorial C:Subversion/Projects/SpiraTeam Add Ons/Trunk/Version Control Tools/TaraVaultProviderRapise_TestProjectLocker/SpiraTeam Add Ons/Trunk/Version Control Tools/TaraVaultProviderRapise_TestProjectLocker/SpiraSelenium/UsingSelenium/UsingSelenium.stest C:\Subversion/Brojects/SpiraTeam Add Ons/Trunk/Version Control Tools/TaraVaultProviderRapise_TestProjectLocker/SpiraSelenium/UsingSelenium.stest C:\Subversion/Brojects/SpiraTeam Add Ons/Trunk/Version Control Tools/TaraVaultProviderRapise_TestS/Selenium/UsingSelenium.stest C:\Subversion/Bapise/TestProjectLocker/SpiraSelenium/UsingSeleniumAPI.seleniumAPI.stest C:\Usersiadam.sandman/Documents/My Rapise Tests\SoleniumAPI.SeleniumAPI.stest C:\Usersiadam.sandman/Documents/My Rapise Tests\SoleniumAPI.selest C:\Usersiadam.sandman/Documents/My Rapise Tests\SoleniumAPI.selest C:\SpiraRepository/Demo Test 2 TC163.stest C:\ProgramDatalDocuments\Rapise\Samples\UsingSpreadSheet\UsingSpreadSheet.stest	Online Resources • Rapise Home Page • Documentation • Forum • Help Desk

Purpose

To display the latest news and information regarding Rapise and the currently open test.

The Start Page is intended to be a convenient entry point for most common tasks related to test design and execution. The Start Page provides:

- 1. A link to the Spira Dashboard: This will open the <u>Spira Dashboard</u> that provides a convenient way to interact with Inflectra's SpiraTest test management system or Inflectra's SpiraTeam application lifecycle management system.
- 2. Current Test information block, including:
 - o 3. Test Name and available scenarios
 - 4. Test Parameters including the Spira Properties for the test. These include the IDs of the project and test case in SpiraTest. In addition, for web-based tests there will be the special Browser selection property. All tests will include any custom properties set by user.
 - o 5. Test Description. This information is taken from a Readme.htm file (if it exists in the test folder of the current test). If this file does not exist then the first /** ... */ comment inside the Test function is displayed instead.
- 6. Quick Start Guide This is an interactive tutorial for beginners who are using the system for the first time. It may be hidden by unchecking the Show checkbox.
- 7. Recent Tests. This displays a clickable list of recently used tests
- 8. Browser Samples. This displays a list of available Rapise samples. Some samples are

shipped with Rapise while others are provided from the online public repository.

• 9. The Fetch Samples button is used to download/update additional samples from online public repository.

How to Open

The **Start Page** opens automatically with Rapise. This behavior can be modified in the <u>Options</u> dialog using the **ShowStartPageOnStartup** setting.

2.4.46 Spira Dashboard

Purpose

This page displays information from the SpiraTest or SpiraTeam server that this instance of Rapise is connected to. More details on using Rapise with either SpiraTest or SpiraTeam can be found in the separate **Using Rapise with SpiraTest Guide**. A copy of this guide should be in the Start > Programs menu created by the Rapise installer.

The dashboard displays information about the current Spira project, including the associated test cases, test sets and automation hosts:

Screenshot

A typical S	Spira dashboard w	ill look like the follo	wing:	
📬 Start Page 🏽 🛞 We	b Test.js 🥃 Spira Dashboard		Options	• X
SpiraTea	am Dashboard for	Test Management		~
Project Rapise Demo Proje	Description			Login / Register Welcome, administrator!
Test Case	S.		My Assigned	Logout
✓ Id	Name	Description	Action	Auto Login: 🗹
TC000035 TC000164 TC000165 TC000036 TC000037	Functional Tests Flex 4 Test Desktop Demo Test Regression Tests Library Web Test		Save As Save As	Spira Connection Spira URL: http://doctor/SpiraTeam User Name: administrator Local Folder: C:SpiraRepository Edit Connection Settings
Test Sets	Name	Description	+ Create Test Set Add to Test Set	Automation Hosts
TX000019	la Rapise Demo Set		Execute	Token: WIN7
<u>TX000022</u>	DH Automated Testing		Execute	Description: null + Create Host for this Machine
TX000037	🖓 Rapise Demo 2		Execute	

Each of the sections is explained separately below:

Spira Login/Sign-Up

This section will display the name of the currently configured Spira user (if there is one) together with the option to either login to an existing Spira instance or to sign-up for a new one:

Login / Register		
Login or <u>Register</u>		
Auto Login:		

- Login: this will log you into the instance of Spira listed in the Connection Info section
- Sign Up: this link will take you to the Inflectra website where you can sign up for a Spira account.
- Auto Login: if you select this option, Rapise will automatically login to Spira when it first starts up.

Once you login to the instance of Spira, the widget will change to the following:

Login / Register		
Welcome, administrator!		
Logout		
Auto Login: 🗹		

• Logout: this will log you out of the instance of Spira listed in the Connection Info section

Connection Info

This section will display the URL, login and corresponding local repository folder for the current Spira instance (if one has been set).

Spira Connection				
User Name:	Spira URL: http://doctor/SpiraTeam User Name: administrator Local Folder: C:\SpiraRepository			
Edit Connection Settings				

To change the current connection (or to set one up if this is a new installation of Rapise), click on the **[Edit Connection Settings]** button. That will display the <u>Connection Settings</u> dialog box:

URL:	http://doctor/spirateam	
User Name:	administrator	
Password:		
Default Repository Path:	C:\Temp\RapiseTests	
Token (read only):		

You can then change the current Spira connection using this dialog box. See the topic on <u>Spira</u> <u>Integration</u> for more details.

Automation Hosts

This section will display a list of the automation hosts available in the currently selected Spira project:

Automation Hosts	
Select host	~
+ Create Host for this Machine	

An **automation host** is a notional computer that Spira uses to assign specific **test sets** to specific computers running Rapise. This allows you to schedule tests to run on different computers remotely. When you first connect to Spira, it will not know which automation host the current machine matches.

Using the dropdown list you can select one of the displayed automation hosts:

Automation Hosts	
Select host	~
Select host VM-Win7 VM-Rapise	

That will tell Rapise that this local computer is in fact this Spira automation host. Any test sets scheduled in Spira for this automation host will now be executed on this computer running Rapise.

If you don't see the current automation host listed, you can click on the **[Create Host for this Machine]** button to create a new automation host entry for the current computer:

Automation Host Name:	TARDIS	
Token:	TARDIS	
Description:	1	
		OK Cancel

This screen lets you enter a display name (Name), system name (Token) and long description for a new automation host that Rapise will create in the current Spira project. Click **[OK]** to confirm the new automation host.

Test Cases

This section displays a list of **test cases** that are either created by the current Spira user or are assigned to the current Spira user:

Te	est Case	S		My Assigned
~	ld	Name	Description	Action
	TC000035	🗁 Functional Tests		
	TC000164	Flex 4 Test		Save As
	TC000165	E Desktop Demo Test		Save As
	TC000036	🗁 Regression Tests		
	TC000037	Library Web Test		Open
				+ Create Test Set Add to Test Set

Each test cases will be displayed with the ID, name and long description of the test case together with an icon that indicates the type of test case:

- 1. 🔄 Manual test case that has no automation script attached.
- 2. 1 Test case that has an existing Rapise test attached.
- 3. 🖆 Test case that has a non-Rapise automation script attached.

Clicking on the hyperlink ID will open up the test case inside Spira in your web browser. For test cases that have a linked Rapise test, there will be an **[Open]** button available. Clicking on this button will open the test in Rapise. For test cases that do not have a linked Rapise test, there is the **[Save As...]** option that lets you save the current Rapise script against the test case.

In addition there are two other options available:

• Create Test Set: Clicking on this button will allow you to create a new test set inside Spira. It

Name:	New Test Set
Description:	2014-09-08 14:23:11/Auto created/Test Cases: TC00002,TC00003

will display the following dialog box when you select at least one test case and click the button:

Enter in the name of the test set you want to create and click [OK].

• Add to Test Set: When you select at least one test case and one test set, then click this button it will add the selected test cases to a specific test set.

Test Sets

This section displays a list of **test set** that are either created by the current Spira user, are assigned to the current Spira user, or are assigned to the automation host that this instance of Rapise is installed on:

Test Sets			My Created
✓ Id	Name	Description	Action
TX000019	Rapise Demo Set		Execute
TX000022	E DH Automated Testing		Execute
TX000037	Rapise Demo 2		Execute
TX000038	Rapise Test Set Multiple		Execute
TX000050	Arz Demo Test Set		Execute
TX000051	Mission Manager 1.0 Test		Execute

Each test set will be displayed with the ID, name and long description of the test set.

Clicking on the hyperlink ID will open up the test set inside Spira in your web browser. For test sets that are marked as **automated**, there will be an **[Execute]** button available. Clicking on this button will execute the test in **RapiseLauncher**. This is described in more detail in the <u>SpiraTest Integration</u> topic.

2.4.47 Spira Properties Dialog

Screensnot		
📀 SpiraTeam v4.0	REST API Test Properties	\times
Spira Properties		
Spira Project:	SpiraTeam]
Test Folder:	REST API Tests]
Test Case:	REST v4.0 API Test]
	> Unlink from Test Case	
	Close	

Purpose

The purpose of this dialog is to allow you to show you the <u>SpiraTest / SpiraTeam test case</u> that the current Rapise test script is associated with.

How to Open

Go to the Test ribbon and click the Spira Properties button in the Tools tab:

Settings

The dialog box will list the following:

- Spira Project this displays the name of the project in SpiraTest that the test script is saved in
- Test Folder this displays the name of the folder that the linked SpiraTest test case is contained in
- Test Case this displays the name of the test case in SpiraTest that the test script is linked to

If you want to unlink the current test script from SpiraTest (for example to save to a different SpiraTest project), click on the **'Unlink from Test Case'** button.

2.4.48 Spy Dialog

Screenshot

🔎 Rapise Spy	- 0	×
Start Tracking (Ctrl-G) Learn Ob	ect	H •
UIAutomation Object		
- Tree	- Properties	
Node1[Treeltem]	RuntimeldProperty System.Int32[]	^
[Button]	 ExpandCollapsePattemIdentifier 	
Node1[Text]	ExpandCollapseStatePrc Expanded	
Node1Node1[TreeItem]	✓ Primary	
Node1Node2[TreeItem]	AutomationID	
	IndexInParent 2	
	Name Node1	
	NativeHandle 0	
	 SelectionItemPattemIdentifier 	
	IsSelectedProperty False	
	SelectionContainerPrope	\sim
- HWND - Tree	- Properties	
AUTWPF[HwndWrapper[AUTWPF.exe;;d5a8f4f6-7fdf-4	ScreenWidth 1920	~
	ScreenX 0	
	ScreenY 0	
	> Size 1000, 750	
	✓ Primary	
	AccessibleObject AUTWPF[ROLE_SYS	TEM
	ClassName HwndWrapper[AUTW	PF.ex
	> Rect 192, 192, 1000, 750	
	Text AUTWPF	
	Visible True	

Purpose

The Spy dialog is used to Object Spy.

How to Open

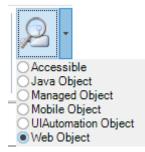
There are three ways to open the Spy dialog: 1. Press the **Spy** Button on the Ribbon (**Test** tab > **Tools** menu)



2. Press the **Spy** Button on the <u>Recording Activity Dialog</u> Note: If you use this method, the dialog has an extra **Learn Selected** button.

Choosing the type of Spy

You can change the type of Spy that will be launched by clicking on the down arrow to the right of the Spy icon in the main application Ribbon:



There are **six** types of Spy available:

- 1. Accessible This is used to inspect applications that expose their properties using the Microsoft Active Accessibility (MSAA) technology. This is typically used by applications written in MFC, ATL, Qt, C++ and Visual Basic.
- Java Object This is used to inspect applications written using the Java AWT and Swing UI frameworks.
- 3. **Managed Object** This is used to inspect applications written in .NET 1.1, .NET 2.0, .NET 4.0 using Microsoft Windows Forms.
- 4. **Mobile Object** This is used to inspect mobile applications running on iOS or Android devices as well as the iOS or Android simulator
- UlAutomation Object This is used to inspect applications that expose their properties using the Microsoft's newer UlAutomation technology. This is typically used by applications written in WPF, Silverlight and Java SWT.
- Web Object This is used to inspect web applications using any of the supported web browsers. It also allows you to dynamically query web pages using <u>CSS</u> or <u>XPath</u> and learn the results as objects.

Start Tracking

The **Start Tracking** button (or CTRL+G) causes Rapise to enter **Tracking Mode**. In **Tracking Mode**, Rapise investigates the object under your mouse. It identifies the object's type and learns the object's properties. As you move your mouse, the objects you point to are highlighted (a box is drawn around them).

Stop Tracking

The **Stop Tracking** button is only visible in **Tracking Mode**. Press Stop Tracking (or CTRL+G) to exit Tracking Mode. The Spy dialog will display information for the last object highlighted.

Learn Object

The **Learn Object** button will be visible if you clicked the Spy during recording (vs. just clicking the Spy icon in the Test ribbon). This lets you add a specific object in the Spy treeview to the currently recorded test.

Maximize/Minimize buttons

The maximize and minimize buttons control the appearance of the dialog. They either hide or make visible the sections to their right or below. See the example below. The button highlighted in yellow in the left image is pressed to show/hide the appropriate pane.

Rapise Spy			
Start Tracking (Ctrl-G)			⊞•
JIAutomation Object Tree	- I operties		
 Rapise Documentation[Window] [ToolBar] [Pane] [Pane] [Pane] [Tab] [Rapise Documentation[TitleBar] System Menu Bar[MenuBar] Minimize[Button] Restore[Button] 	dem Type Property Labeled ByProperty Localized Control Type Property Name Property Orientation Property Processid Property Runtimeld Property Automation ID Index.In Parent Name	Not supported Not supported pane Not supported 20376 System.int32[] No id D.	
- HWND	- roperties		
 napise Documentation[HH Parent] [ToolbarWindow32] [HH Child] [Shell Embedding] > [Shell DocObject View] [HH SizeBar] 	PID PosHeight PoeWidth PosX PosY PDD	20376 666 1122 244 62 2992	
✓ [HH Child] ▶ [SysTabControl32]	Process ProcessName Size-	C:\Windows\hh.exe 1122, 666	
	AccessibleObject	(ROLE_SYSTEM_WINDOW)	

Accessible Object

This is the Spy dialog that is used for Accessible (MSAA) objects. It is described in more detail in the <u>Accessible Spy</u> topic.

- Tree		- Properties		
 [ROLE_SYSTEM_CELL] Screenshot(ROLE_SYSTEM_TEXT.) select an application to record dialog(ROLE_SYSTEM_GRAPHIC.mik:@MSITStore:C:\User [ROLE_SYSTEM_TEXT.] Purpose(ROLE_SYSTEM_TEXT.] The [ROLE_SYSTEM_TEXT.] 		State Text Primary ChildCount ChildId DefaultAction Description	55 CHILDID_SELF	1
Select an Application to Record[ROLE_SYSTEM_TEXT.] (SAR) Dialog appears before [ROLE_SYSTEM_TEXT.] Recording[ROLE_SYSTEM_LINK.mk:@MSITStore:C:/Users/Public/Documents/Rapise/Hel takes place. It queries the user for which program to record, as well as what [ROLE_SYST Recording Library[ROLE_SYSTEM_LINK.mk:@MSITStore:C:/Users/Public/Documents/Ra to use. [ROLE_SYSTEM_TEXT.] [ROLE_SYSTEM_TEXT.]	*	HWND IndexinParent LocationRECT Name Role State Value	[Internet Explorer_Server] 0 (X=246,Y=108,Width=1101,Height=1966) ROLE_SYSTEM_CELL 0	

The **Accessible Object** section of the Spy dialog shows properties of the object that are visible through the Microsoft Active Accessibility interface.

Tree

The spied upon object and its children are displayed here.

Properties

Object fields and field values are displayed here.

Tools

- Mouse Button Click: Emulate Left mouse click for the item selected in Spy tree.
- Default Action: Execute DoDefaultAction for given accessible object.
- Set Selection: Perform accSelect using the option flags set in the corresponding checklist (above).

Java Object

This is the Spy dialog that is used for Java (Swing / AWT) objects. It is described in more detail in the Java Spy topic.

💩 Start Tracking (Ctrl-G)		H -
🗕 Java Object		
- Tree	Properties	
🖃 panel0[java.awt.Panel]	Primary	
label0[java.awt.Label]	Handle	790046
textArea1[java.awt.TextArea]	IndexInParent	0
list1[java.awt.List]	Name	panel0 E
button1[java.awt.Button]	VmID	463120
	Secondary	
	accessibleContext	java.awt.Panel\$AccessibleAWTPanel@df8ff1
advAwtControls[java.awt.Button]	alignmentX	0.5
checkbox1[java.awt.Checkbox]	alignmentY	0.5
	background	java.awt.Color[r=200,g=220,b=255]
choice2[java.awt.Choice]	component	
······································	componentCount	9
	components	[Ljava.awt.Component;@cf40f5

The Java Object section of the Spy dialog shows properties of the object that are visible through the Java Access Bridge interface.

Tree

The spied upon object and its children are displayed here.

Properties

Object fields and field values are displayed here.

Managed Object

This is the Spy dialog that is used for Managed (.NET) objects. It is described in more detail in the <u>Managed Spy</u> topic.

Start Tracking (Ctrl-G)				- 11
Managed Object				
- Tree		Properties		
□· MainForm[AUTDotNet.MainForm]		ManagedSpy Properties	ies	*
propertyGrid1[System.Windows.Forms.PropertyGrid]		Component Type		_
		Handle	1180844	
dateTimePicker1[System.Windows.Forms.DateTimePicker]	-	IsManaged	True	
dataGridView1[System.Windows.Forms.DataGridView]	-	⊟ Misc		
toolStrip1[System.Windows.Forms.ToolStrip]		AcceptButton		
		AccessibilityObject		
btAlias[System.Windows.Forms.Button]		AccessibleDefaultAction[Descripti	
listView2[System.Windows.Forms.ListView]		AccessibleDescription		
···· listView1[System.Windows.Forms.ListView]		AccessibleName		
contextMenuHolder[System.Windows.Forms.Label]		AccessibleRole	Default	
the autTabControl[System.Windows.Forms.TabControl]		ActiveControl		
treeViewNoCB[System.Windows.Forms.TreeView]		ActiveMdiChild		
autLabel[System.Windows.Forms.Label]		AllowDrop	False	
		AllowTransparency	False	
treeViewCB[System.Windows.Forms.TreeView]		Anchor	Top, Left	
autEdit[System.Windows.Forms.TextBox]		AutoScale	False	
···· radio2[System.Windows.Forms.RadioButton]	•		5, 13	*

The **Managed Object** section of the Spy dialog shows properties of the object that are visible through .NET Framework reflection interface.

Tree

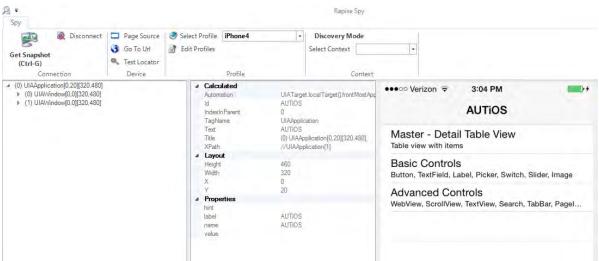
The spied upon object and its children are displayed here.

Properties

Object fields and field values are displayed here.

Mobile Object

This is the Spy dialog that is used for Mobile objects. It is described in more detail in the <u>Mobile Spy</u> topic.



The **Mobile Object** section of the Spy dialog shows a snapshot of the screen displayed on the connected Mobile device as well as the properties of the currently selected object. You can selected the object either by clicking on the screen snapshot or the control hierarchy displayed to the left. The properties displayed will depend on the type of mobile device being tested (iOS vs. Android).

Tree

The spied upon object and its children are displayed here. When you click on an object it will also be highlighted in the **snapshot** view to the right.

Properties

Object fields and field values are displayed here.

Snapshot

This displays a snapshot of what is displayed on the mobile device being tested. The objects in the snapshot are clickable, which allows you to visually select objects from the hierarchy.

Tools

- Get Snapshot (CTRL + G) This will connect to the mobile device and get the latest snapshot from the mobile device and display in the right-hand window.
- Disconnect This option disconnects the Spy from the mobile device and ends the connection.
- Learn Object This option is only displayed in Recording mode and lets you take the currently selected object and add it to the <u>Object Tree</u> for the current test. It can then be used as a scriptable object in the test script.

- **Page Source** This lets you view the source of the mobile device in a text editor such as Notepad. It will show the objects in the treeview represented as an XML document.
- Go to URL This will instruct the mobile device to navigate its built in web browser to a specific URL.
- **Test Locator** This will display the <u>Mobile Test Locator</u> dialog box that lets you try different locators to resolve specific objects in the object hierarchy. It will include options such as using XPath and IDs.
- Select Profile This lets you change the profile of the mobile device you are testing while the Spy dialog is open.
- Edit Profiles This will open up the <u>Mobile Settings</u> dialog box. You cannot be connected to do this.
- Context This will display either 'Discovery Mode' or 'Recording Mode'.

Events

The <u>Mobile Spy</u> also includes an **Events** ribbon tab that lets you send events to mobile device from Rapise, as if you were actually performing them on the device:

Spy	Events			
Тар	Precise Tap 💌	Text		Accept Alert
Swipe 🝷	Scroll To	Send Keys		Dismiss Alert
Shake		Execute Script		Change Orientation
	Touch		Text	Misc

This dialog lets you perform the following events on the device:

- Tap this will simulate tapping the currently selected object on the device
- Swipe you specify the start and end points of the swipe operation. This is useful for simulating a real swipe on the device in a specific direction at a specific location (e.g. on a progress selector)
- Shake for devices that support it (e.g. iOS) this simulates shaking the device physically
- **Precise Tap** you specific the specific location on the screen within the bounds of the current object that you will be simulating a tap.
- Scroll To simulates scrolling to the selected object in the device object tree (which may not be visible).
- Text / Send Keys to use this, enter in text in the text box and click 'Send Keys', this sends text to the currently selected object as if you were using the virtual keyboard on the device.
- Accept Alert if you have a popup alert on the device, this simulates accepting it
- Dismiss Alert if you have a popup alert on the device, this simulates dismissing it
- Change Orientation for devices that support it, this simulates changing the orientation of the device from landscape to portrait (or vice-versa)
- Execute Script this is not currently supported and is for future functionality

UIAutomation Object

This is the Spy dialog that is used for UI Automation (WPF, Silverlight) objects. It is described in more detail in the <u>UIAutomation Spy</u> topic.

Start Tracking (Ctrl-G)			- 1
 UIAutomation Object 			
- Tree	Properties		
□ [List] IistBoxItem0[ListItem] IistBoxItem1[ListItem] IistBoxItem2[ListItem] IistBoxItem3[ListItem] IistBoxItem4[ListItem]	Item Type Property Labeled ByProperty Localized Control Type Property Name Property Native Window Handle Property Orientation Property ProcessId Property	Not supported Not supported list 1378098 Not supported 8728	
	RuntimeIdProperty Primary AutomationID IndexInParent Name NativeHandle SelectionPatternIdentifier CanSelectMultipleProperty	System.Int32[] listBox1 18 1378098	E

The **UIAutomation Object** section of the Spy dialog shows properties of the object that are visible through the UIAutomation interface.

Tree

The spied upon object and its children are displayed here.

Properties

Object fields and field values are displayed here.

Web Object

This is the Spy dialog that is used for Web objects. It is described in more detail in the <u>Web Spy</u> topic.

P = R Spy	Rapise Spy
Get Snapshot (Ctrl-G) Browser IE	Test XPath Learn Test CSS Learn Locators
DOM Tree <html xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"> <html> <html< html=""> <ht< td=""><td>DOM Element ID 7bf3872c-bced-c921-bbc6-e9decc135fce tagName DIV Attributes class class page</td></ht<></html<></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html></html>	DOM Element ID 7bf3872c-bced-c921-bbc6-e9decc135fce tagName DIV Attributes class class page
<script clear="" hideskiplink="" src="/WebResource.axd?d=9_D5u0u-Rawwl" type="text/javascr</td><td>height 690 width 962 x 196 y 77 Selectors css css html > body > form#ct01 > div.mth-of-type(3)</td></tr><tr><td> <div class="></div> <div class="main"> <h2></h2> <div style="margin-right: 150px;"> </td><td>xpath /html/body/form[@id='ctl01"]/div[3]</td></tr><tr><td></td><td>Recording Activity for "Inflectra Library Information System - Internet Explorer"</td></tr><tr><td> div class="clear"></div><div class="clear"></div><div class="footer"></div><scripttype="text/javascript"></script>	# Object Action Data Comment

The **Web Spy** is used to inspect web applications running on any of the supported web browsers (currently Internet Explorer, Firefox and Chrome). It allows you to view the hierarchy of objects in the web browser **Document Object Model (DOM)**. In addition it makes the testing of dynamic data-driven web applications easier because it lets you test out dynamic <u>XPath</u> or <u>CSS</u> queries against the web page and verify that the objects return match your expectations.

For more information on the Web Spy, please refer to the Web Spy topic.

- HwND	- Properties		
 ✓ Rapise Documentation[HH Parent] [Toobar\/indow32] ✓ [HH Child] ✓ [Shell Embedding] ▶ [Shell DocObject View] [HH SizeBar] ▲ [HH Child] ▲ [SysTabControl32] [Insctls_updown32] [SysTreeView32] 	PosX PosY PPID > Process ProcessName > Size > Primary AccessibleObject ClassName > Rect Text Visible WindowStvle	244 62 2992 C.\Windowe\hhitexe 1122,666 [ROLE_SYSTEM_WINDOW] Shell Enbedding 244, 62, 1122,666 True WS_TABSTOP_WS_CLIPCHILDREN, WS_C	I IPSI

The **HWND Object** section of the Spy dialog shows properties of the object that are visible with its HWND handle.

Tree

The spied upon object and its children are displayed here.

Properties

Object fields and field values are displayed here.

Tools

- Mouse Button Click: Emulate Left mouse click for the item selected in Spy tree.
- Highlight: Draw rectangle surrounding selected object (HWND or Accessible).

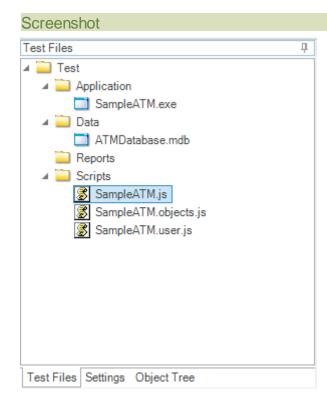
These tools can be accessed from the right-click Spy context menu:

Parent
Highlight
Refresh
Default Action
Default Action Mouse Click

See Also

- Microsoft Active Accessibility is described here http://msdn.microsoft.com/en-us/magazine/cc301312.aspx
- HWND is described <u>HERE</u>.
- Microsoft UIAutomation is described here http://support.microsoft.com/kb/971513/

2.4.49 Test Files Dialog



Purpose

- The Test Files dialog allows you to navigate and alter the Test hierarchy, including the following:
 - The main JavaScript script files (*.js)
 - o The report files (*.trp)
 - o Images captured during execution using Checkpoints
 - REST web services (*.rest)
 - SOAP web services (*.soap)
 - Analog recording files (*.arf)
 - Excel spreadsheets (*.xls and *.xlsx)
 - Applications to launch (*.exe or *.bat)
 - Other data files (*.txt)

How to Open

The Test Files dialog is part of the Default Layout.

Context Menu (Folder)

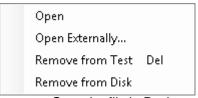
Right click on a folder to see:

-	Eller Oreante and add a re-
	Remove All from Disk
	Remove from Test Del
	Create Sub-Test
	Reload
	New Group
	Add File(s)
	Create File

- Create File: Create and add a new file to the test.
- Add File: Add an existing file to the test.
- New Group: Create a logical grouping of files in the test. This will not add a folder to the file system.
- Reload: Refresh group contents. Use it for <u>filter groups</u> ('IsFilterGroup' is set to 'True' in group properties), e.g. for Report group.
- Create Sub-Test...: Launch Create Sub-Test dialog.
- Remove from Test: Remove the selected grouping from the test. This does not delete included files from your hard disk.
- Remove All from Disk: Remove all files included into the selected grouping from your hard disk.

Context Menu (File)

Right click on a file to see:



- Open: Open the file in Rapise.
- **Open Externally...**: Open the file using associated program. E.g. if a Notepad is registered in Windows to open TXT files, then TXT file will be opened by Notepad.
- Remove from Test: Remove the file from your test. This does not delete the file from your hard disk.
- Remove from Disk: Remove the file from your test and hard drive.

Filter Groups

Filter groups read its contents from disk according to specified path and wildcard. You may setup a filter group by editing group properties:

Pro	operties	中
•		
Ξ	Misc	
	FilterPath	Reports
	FilterWildcard	*.trp
	IsFilterGroup	True 💌
	Name	Reports
tri		show the list of all files rd' belonging to 'FilterPath'

- FilterPath: Root path to find files via wildcard (valid only if 'IsFilterGorup' is 'True').
- FilterWildcard: Filter wildcard (valid only if 'IsFilterGorup' is 'True').
- IsFilterGroupt: 'True' if directory should show the list of all files matching 'FilterWildcard' belonging to 'FilterPath'.
- Name: Group name.

2.4.50 Variable/Call Stack View

ariables			1
Function/Vat	Value	Type	, c
표 Test	C:\Program		
🖃 JScript global code	T\Admin\Wrappers\Engine\Player.js:411		
WScript	<i>()</i>	Object	
SeSWait	{}	Object	
_SeSGetObjectInfo	{···}	Object	
C-CCI	13	Object	

Purpose

Lists the functions in the current call stack. Beneath each function, variables/objects local to that function are listed with their value and type.

How to Open

Begin <u>debugging</u> a script. The Variable/Call Stack View will open automatically.

Go to a function definition

Double click on a function to go to its definition.

2.4.51 Verify Object Properties Dialog

Scree	nshot		
Verify	Object Propertie	2S	
#	Name 🔺	Value	*
	Bitmap	Login	
	BWBitmap	Login	
	Class	button	
	Enabled	true	
	Height	26	
	Href	javascript:doPostBack('ctl00	
	ld	btnLogin	
	InnerText	Login	
	Name		
	NodeText	Login	
	ObjectType	HTMLObject	
	PageTitle	SmarterMail	
	PageURL	https://webmail.inflectra.com/Login	
	State	1048580	
	Style		-
	OK	Cancel	

Purpose

Use the Verify Object Properties dialog during recording to add checkpoints.

How to Open

- 1. First, open the <u>Recording Activity Dialog</u>.
- 2. Position the mouse over an object and press Ctrl+1, or

3. Press the Verify button and then click the target object with the mouse cursor.

Create a Checkpoint

Your checkpoint will be associated with a particular object. That object's properties will be listed in the **Verify Object Properties** dialog. Check those properties that you wish to verify during <u>playback</u>. Enter expected values for the selected properties in the **Value** column. **Note**: The **Bitmap** and **BWBitmap** properties are images of the object.

Bit mapIBWBitmap·BWBitmap·ClassButtonClassButtonHeight29Mame1Name1ObjectTypeWin32ButtonState0Value·Width36WindowTest1X719Y276	*	Name /	Value	
Class Button Height 29 Name 1 ObjectType Win32Button State 0 Value		Bit map.		
Height29Name1ObjectTypeWin32ButtonState0Value-Width36WindowText1X719		BWBitmap	Ŧ	
Name1ObjectTypeWin32ButtonState0Value		Class	Button	
ObjectType Win32Button State 0 Value 0 Windth 36 WindowText 1 X 719	Z	Height	29	
State 0 Value 0 Width 36 WindowText 1 X 719		Name	1	
Value Width 36 WindowText 1 X 719		ObjectType	Win32Button	
Width 36 WindowText 1 X 719		State	0	
WindowText 1 X 719		Value		
🗖 X 719		Width	36	
		WindowText	1	
₩ 276		х	719	
		٧	276	

Press the OK button. The Verify Object Properties dialog will close, and the <u>Recording Activity</u> dialog will contain a new Verify action:

#	Object	Action	Data	Comment		
21	01111	<u> </u>	0.11.11.22			
2	1	Verify	29	Verify that: Heigh	nt=29	
						_
Veri	fy (Ctrl+1)	Learn (Ct	rl+2) 🥨	Spy(Ctrl+5)	Pick Object	Pause
	og (Ctrl+4)		_Simulated		Cancel	Finish (Ctrl+3)

The generated script will have a corresponding assert statement:

```
//Verify that: Height=29
Tester.AssertEqual("Verify that: Height=29", SeS('Obj1').GetHeight(), "29");
```

See Also

- <u>Recording</u>
- <u>Assert Statements</u>

2.4.52 Warning View

Screenshot	
Warnings	д
C:\Program Files\Inflectra\Rapise\UsingMSExcel\UsingMSExcel.js (1,1): mismatched input ')' expecting RPAREN	
Output Warnings Errors Find Results	

Purpose

To display syntax error messages as you edit javascript files.

How to Open

The Warning view is part of the Default Layout.

Error Message

C:\Program Files\Inflectra\Rapise\UsingMSExcel\UsingMSExcel.js (1,1): mismatched input ')' expecting RPAREN Double click on an error message to go to the corresponding source line.



Watch		
3 11 🔕 🖕		
Expression	Value	
Variables Watch Ou	tput Warnings Errors Find Results	

Purpose

To input expressions and view their values as the script executes.

How to Open

Begin <u>debugging</u> a script. The Watch View will open automatically.

Natch		7
) II 🕹 🖕		
Expression	Value	

2. Double click on the highlighted line, under the Expression column. A text box will appear.

	Ţ.
Value	-
	Value

3. Input the expression you wish to investigate. Press Enter.

Watch		ų.
) II 🔕 🖕		
Expression	Value	
g_commandInterval	100	

Widgets				
Watch				
; 🗇 🗈 🥝 🖕				
From left to right	: Copy (an entire row) 🔍	Copy Watch Value 💷,	Delete 🔕.	

2.4.54 File Menu

Purpose

The File menu provides quick access to all the File management functions in Rapise. Many of these are also available on the main <u>Test ribbon</u>.

Screenshot

E		Rapise 3.0.4.25 [C:\Users\ \Documents\My Rapise Tests\Web Color Test\Web Color Test.sster
Recent Tests	Recent Tests	
Create a new Test	C:\Users\	\Documents\My Rapise Tests\Web Color Test\Web Color Test.sstest
Open Open an existing Test	C:\Users'	\Documents\My Rapise Tests\iOS Test 2\iOS Test 2.sstest
📂 Open File	C:\Temp' (login IB	Rugepresa.sstest
Save	C:\Users'	\Documents\My Rapise Tests\iOS Test 1\iOS Test 1.sstest
Save the current Test	C:\Users'	\Documents\My Rapise Tests\Website Ajax Tests\Website Ajax Tests.sstest
💦 Save to Spira	C:\Users'	\Documents\My Rapise Tests\Rest Test 1\Rest Test 1.sstest
Save As	C:\ProgramData\Docu	iments\Rapise\Samples\CreateNewAuthor\CreateNewAuthor.sstest
Save current Test As	C:\Users\	Documents\My Rapise Tests\Ocr Test\Ocr Test.sstest
	C:\Users\Public\Docu	ments\Rapise\Samples\UsingSpreadSheet\UsingSpreadSheet.sstest

Options

The File menu has the following options:

- Create a new Test creates a new Rapise test, it can be saved either to Spira or locally.
- Open an existing Test opens an existing test that is already available locally.
- Open File opens a single file and adds to the current test project
- Save the current Test saves the current test locally
- Open Test from Spira opens a test from Spira and downloads to the local repository
- Save to Spira saves the current test to the Spira test management system
- Save As saves the current test locally with a different file name
- Exit exits Rapise.

2.4.55 Web Spy Settings

Purpose

This dialog box displays the list of **Web Spy settings** and lets you change the behavior of the <u>Web Spy</u> tool.

Screenshot

Veb Spy Settings	
₩ 2↓ III	
DOM Attributes Exclude Filter DOM Attributes Include Filter Enable Universal XPath Generator Fast DOM Tree Build Show Internal ID XPath type to learn	True True False xpath_std
DOM Attributes Exclude Filter List DOM element attributes to hide in the DO	M Tree of the Web Spy (comma separated).
	OK Cancel

How to Open

You can open this dialog box from two places:

- From the main Rapise Options ribbon (in the Web Testing tab).
- From the <u>Web Spy</u> tool when you click on the 'Options' ribbon menu entry.

General Settings

This dialog box has the following settings:

- **DOM Attributes Exclude Filter** List the DOM element attributes to hide in the DOM Tree of the Web Spy (comma separated).
- **DOM Attributes Include Filter** List the DOM element attributes to show in the DOM Tree of the Web Spy (comma separated). If both Include and Exclude filters are set then Include filter prevails.
- Enable Universal XPath Generator If set to 'True' it creates the XPath and CSS selectors inside the Web Spy itself rather than relying on the web browser to do the generation. This is usually much faster (especially when using Internet Explorer)
- Fast DOM Tree Build If set to 'True' then Rapise uses embedded code to get the DOM Tree when using either the Internet Explorer HTML library or any of the <u>Selenium libraries</u>.
- Show Internal ID If 'True' then DOM Element pane shows internal ID of an element. This ID is

purely internal to Rapise and is not part of the HTML web page or web application

• XPath Type to Learn - This is described in the section below:

Different Types of XPath

The WebSpy has variety of options for XPath generation. Having a number of **different approaches for generating XPath** has its benefits. It gives Rapise incredible flexibility in getting the best results for different situations:

- 1. Sometimes we can recognize an element by its text. In such cases the simplest and most efficient XPath will use the node text.
- 2. Sometimes we have a regular structure (tables and grids) and it is better to have row and column indices inside the XPath.
- 3. Sometimes we deal with an application framework that uses custom attributes (e.g. aria, angular, bootstrap). So if we use them in our XPath then it will be cleaner and more robust.

Veb Spy Settings	
€ . 2↓ .	
DOM Attributes Exclude Filter DOM Attributes Include Filter Enable Universal XPath Generator Fast DOM Tree Build Show Internal ID XPath type to learn	True True False xpath_std
	Advanced,
	 ✓ Set ○ xpath_full ○ xpath_min ○ xpath_indexed ⊙ xpath_std ○ xpath_asia
DOM Attributes Exclude Filter List DOM element attributes to hide in the DOM	O xpath_aria
	OK Cancel

The Web Spy currently supports the following different types of XPath:

- **xpath_full** generates a fully featured XPath path using elements without attributes. This XPath starts with /html and goes through body and other elements towards the required node.
- xpath_min generated with the attributes defined in the "DOM Attributes Include Filter". If the include filter is empty, then it is produced with all attributes except those defined in the "DOM Attributes Exclude Filter". For example, if the include filter contains the "widgetid" custom attribute then the generated XPath would be: //div[@widgetid="dojox_grid__View_1"]

- **xpath_indexed** considers the node as nth of the same kind. For example, the page may contain 250 links across the page, and we want to learn the link somewhere in the middle. In this case indexed XPath will be of the form (//a)[123]
- xpath_std generated and minimized with use of pre-selected set of attributes:
 - o align
 - \circ class
 - o style
 - \circ size
 - \circ tabindex
 - o value
 - o width
 - o height
 - o colspan
 - o rowspan
 - \circ cellspacing
 - o cellpadding
 - \circ border
 - o on* (i.e. onclick, onblur and so on)
 - o Usually it contains most common attributes: id, name, for, role.
- **xpath_aria** generated with the use of just the core id/name attributes plus the special aria attributes: o id
 - o name
 - o for
 - o role
 - o aria-*
- **xpath_text** if possible, generated to match an HTML node simply by its text. For example, <button ...>Refresh</button> is found by: //button[normalize-space(text())="Refresh"]. In many cases this value is empty. This means that there are more than one node with such text.

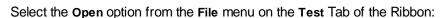
2.5 HowTos

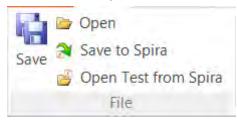
This section focuses on specific tasks that a Rapise user might want to accomplish.

2.5.1 Open a Test

Ribbon

You can open a test in two ways: (1) From the Ribbon, and (2) From the Application menu.





You can also open a test that is stored in **SpiraTest** (our web-based test management system) instead of the local filesystem. This is done by clicking on the **Open Test from Spira** option instead. More details on using Rapise with SpiraTest can be found in the <u>SpiraTest Integration</u> section.

Application Menu File Tab at the top left of the Rapise window. Open the Application Menu by clicking on the The Application menu has an Open Test option, and a list of Recent Test from which you may choose: Rapise 2.1.3.33 [C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1\! (\leftarrow) **Recent Tests Recent Tests** Create From Spira Manual Test C:\Users\adam.sandman\Documents\My Rapise Tests\Spira New Test1\Spira New Test1.sstest **New Test** Create a new Test C:\Users\adam.sandman\Documents\My Rapise Tests\Web Test 1\Web Test 1.sstest Open C:\Users\adam.sandman\Documents\My Rapise Tests\MyTest1\MyTest1.sstest **Open an existing Test** 💕 Open File C:\ProgramData\Documents\Rapise\Samples\CreateNewBook\CreateNewBook.sstest Save Ci\Users\adam.sandman\Documents\My Rapise Tests\TwoDialogsTest\TwoDialogsTest.sstest Save the current Test C:\Temp\SmokeTest\SmokeTest\SmokeTest.sstest 🍰 Open Test from Spira 🔧 Save to Spira C:\Users\adam.sandman\Documents\My Rapise Tests\Auto Test 1\Auto Test 1.sstest Save As C:\Users\adam.sandman\Documents\My Rapise Tests\REST Test Project\REST Test Project.sstest Save current Test As C:\Temp\RapiseTests\ECF_AutoProjectBackup-DO_NOT_DELETE\ECF_AutoProjectBackup-DO_NOT_DELETE.sstest 🎽 Exit

You can also open a test that is stored in **SpiraTest** (our web-based test management system) instead of the local filesystem. This is done by clicking on the **Open Test from Spira** option instead. More details on using Rapise with SpiraTest can be found in the <u>SpiraTest Integration</u> section.

File

2.5.2 Create a New Test

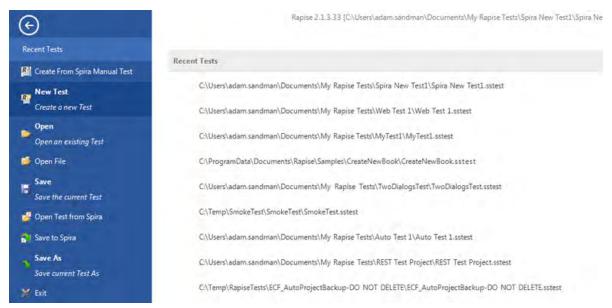
There are two ways to Create a New Test in Rapise:

- 1. From the main Application menu
- 2. From the Start Page

From the Application Menu

Open the Application Menu by clicking on the

Tab at the top left of the Rapise window.



Select the **New Test** option. The <u>Create New Test</u> dialog will appear. Follow the instructions on this dialog.

From the Start Page

Open up the Rapise Start Page:

SpiraTeam	Current Test			Documentation 🖻
Open Spira Dashboard Use the dashboard to link tests to test cases, create and run test sets, view automation hosts and more.	Scenario: User Functions:	DynamicsCRM Test Play DynamicsCRM.user.js DynamicsCRM.js	Test Description	Rapise Quick Start Guide Preparing Firefox for Testing Preparing Chrome for Testing Preparing for iOS Mobile Testin Preparing for Android Testing Using Selenium with Rapise
Fetch Samples	Parameters Spira Properties Browser: Record Title: Project: Test Folder:	Internet Explorer HTML regex* PR000001 TC000018	¥ ^	Shortcuts Create New Test Open Existing Test Start Recording
<u>CreateNewBook</u> <u>DynamicsAX</u>	Test Case:	TC000032	~	Online Resources
DynamicsCRM	Start Here Y	our Rapise Tutorial	show:	Rapise Home Page Documentation Forum
 <u>EditExistingBook</u> <u>FarPoint</u> 	Recent Tests			Help Desk

In the Shortcuts section, click on the 'Create New Test' option:

Sho	ortcuts
D	Create New Test Open Existing Test Start Recording

The Create New Test dialog will appear. Follow the instructions on this dialog.

2.5.3 Learn an Object

To illustrate learning an object, we return to the <u>TwoDialogs</u> sample.

First, let's learn the OK button using recording. We have done this before in the <u>TwoDialogs</u> sample. Steps:

- 1. Run TwoDialogs sample Windows application. By default this will be located in C:\Program Files \Inflectra\Rapise\Samples\TwoDialogs\TwoDialogs.exe
- 2. Start Rapise and create a new test, choose the **Basic Standard Scripting Mode**, and call it TwoDialogsRecording.
- 3. Press the Record/Learn button in the Test ribbon of Rapise.
- 4. When the "Select an Application to Record" dialog is displayed, choose the TwoDialogs.exe application and in the library list, select only the top library on the list "Auto." Press the Select button.
- 5. In the TwoDialogs application, use the mouse to press the OK button. Dismiss the alert message box complaining about the empty name.
- 6. Notice that two things will happen. Firstly, the OK button will be surrounded with a red marker, indicating that the OK button has been learned. Secondly, the action of clicking the OK button is recorded in the Recording Activity dialog. That recording has a single entry.:

# 1	Object OK	Action Action	Data	Comment Press button 'O	ĸ		
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel

7. Press the Finish button (or press Ctrl+3) to end the recording.

- 8. Rapise will return to be the foreground application, and it will have selected the TwoDialogsRecording.js (or whatever name you gave the test when you created it).
- 9. Notice that there is a single line or script that has been added to the script file:

SeS('OK').DoAction();

This line of script has two interesting parts.

The "SeS('OK') is the identity (not the locator or location) of the OK button. This is the object that was learned during recording.

The "DoAction()" is the instruction to the running script to take the action associated with a button. A normal button has only a single possible action - to be pressed.

The Record/Learn process has taken both steps for you, and joined them together.

Now, let's use (normal) object learning to learn the same OK button and to call a method for the object. Steps:

- 1. Run TwoDialogs sample application. By default this will be located in C:\Program Files\Inflectra \Rapise\Samples\TwoDialogs\TwoDialogs.exe
- 2. Start Rapise, create a new test, choose the **Basic Standard Scripting Mode** and call it TwoDialogsLearn.
- 3. Press the Record/Learn button in the toolbar.
- 4. When the "Select an Application to Record" dialog is displayed, choose the TwoDialogs.exe application. Leave the library selection in its default state we will not be using it this time. Press Select. Wait for the Recording Activity dialog to appear in the lower-right corner of the screen.
- 5. Hover the mouse over the OK button of the TwoDialogs AUT but do not press the button.
- 6. With the mouse positioned over the OK button, press Ctrl+2 (the "Learn" command). You will see the OK button surrounded with a red highlight. You will also see that the Recording Activity dialog has been updated with a Learn event.

Recor	rding Activit	ty for "Two	Dialogs"				
# ⊚1	Object OK	Action Learn		Comment Learned OK			
(p)	UK	Lealli	UK	Leallied OK			
Verify (Ctrl+1)		Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel
Learning object:						Advanced>>	Transparent

7. Press the Finish button or Ctrl+3 to end the recording session. You will now see that Rapise has "learned" about the OK button, and the Object Tree in the upper left-hand pane of the Rapise has a new entry called "OK" (shown here expanded). The list of items contained under the OK button entry in the Object Tree is the set of methods and properties available for the OK object.

Methods are listed with purple icons, read properties are listed with blue icons, and write properties are listed with blue and purple icons. The items are grouped by the library, so the DoAction() method is unique to the OK Win32 object whereas the DoClick() method is generic and available for all Windows objects.

Notice that the DoAction property is listed and recall in the previous section when we recorded pressing the button, the DoAction method was chosen for the button-press action.

Objects							
√ id n id‼							
▲ 🥥 Object Tree C:\Users\adam.sandman\Documents\M							
🔺 🛄 Inflectra Rapise Two Dialogs Sample							
⊿ 💩 OK [OK]							
= OoAction							
Generic							
=∳ DoAnalogPlay							
= OoClick							
=level DoDumpWidget							
= OoEnsureVisible							
= OoLButtonDown							
= OoLButtonUp							
= DoLClick							
= OoLDClick							
=							
=∳ DoMButtonUp							
= OoMClick							
= DoMDClick							
= OoMouseMove							
DoRButtonDown							
•							
Objects Files Settings							

8. While we are looking at this OK object, let's make a few observations about it. These observations will be useful for your later dealings with Rapise and will make the script more informational and relevant as you delve into Rapise. First, look down at the Properties box that appears under the Object Tree in the bottom left corner of the Rapise screen. The screenshot below has some of the tree nodes expanded.

roperties		
2↓ 🖻		
E Locators	(4)	
⊡ 0	(3)	
Enabled	true	
Locator Type	Location	
Path	4.1	
Window Class	param:window_class	
Window Name	param:window_text	
⊞ 1	(3)	
⊞ 2	(3)	
Ξ 3	(3)	
Enabled	true	
Locator Type	LocationRect	
🗆 Rectangle	(7)	
Class	param:object_flavor	
Height	26	
Left	101	
Name	param:object_name	
Role	param:object_role	
Тор	172	
Width	75	
Window Class	param:window_class	
Window Name	param:window_text	
0 Object		
ID	ΟΚ	
Version	0	
Recognition		
Class	param:object_flavor	
Flavor	Button	
Ignore Object Name	false	
Library	Generic	
Name	ОК	
Role	ROLE_SYSTEM_WINDOW	
Text	param:object_name	
Туре	Win32Button	
Window		ľ
WALL OF	#00770	

First, notice that the OK button has four (4) "locators" defined. When you have Rapise "learn" an object. it must collect data about that object so that it can relocate it even if the application has moved on the screen, and even if the application is in a different state of execution. In order to accomplish this, Rapise looks for all useful ways to uniquely identify the object. As bad, or perhaps worse, than not being able to find an object would be to find the wrong object on the AUT. Every time Rapise is required to locate this object, it will first try to use the first locator. If it fails to positively and uniquely match with that locator, it will try the second, and so on. Rapise will not give up and declare failure until it has failed to identify with all available locators.

Second, notice the ID entry in the Object section of the pane. This is the name of the object from Rapise's perspective. All Rapise names are available through the SeS() function call. Therefore, if we

want to refer to the "OK" object, we will use SeS("OK") to refer to it. Once we have correctly identified the object, all valid methods and properties can be accessed by using that object as the basis.

Thirdly, notice in the main editor window of the Rapise, that no code has been added. When you identified the OK button, all Rapise did was add the new object to the Object Tree. It did not write any code in the javascript file.

9. In the automated (recorded) section above, you saw that when you pressed the OK button on the dialog, Rapise recorded a function like this:

```
SeS("OK").DoAction();
```

This time, you will use the established name of the OK button object, but do something a little more interesting than its default action to demonstrate how to use Rapise.

10. Move the cursor into the editor part of the Rapise and make sure you are editing the file called TwoDialogsLearn.js. At the moment, this file still looks something like this:

Between the open and close brace, add the following command:

```
SeS("OK").DoClick();
```

Hit the Play button and watch what happens.

The click will register as a command to the object and it will perform the action on the object.

While we have the context of this situation, let's complicate it just a little more to illustrate the intricacy as well as the flexibility of Rapise and SeS.

There is a method whose names looks interesting: DoLButtonDown().

If we were to invoke DoLButtonDown() on the "OK" object, we would expect this would be the same as DoClick().

However, go back to the AUT for a moment. Using the mouse, press the left mouse button over the OK button but don't take your finger off the left mouse button.

What happens is that the button takes its pressed state in appearance, but the button is not clicked.

The reason for this is that the DoClick() (or DoAction()) events cause the mouse button top be clicked as well as released.

Therefore, we would need to have a pair of events:

```
SeS("OK").DoLButtonDown();
SeS("OK").DoLButtonUp();
```

in order to make the "click" happen.

Try this in the test script you have created by adding those two lines of code in place of the DoClick() line.

It doesn't work!

Let's play a little with this problem.

When you press the Play button, leave the mouse alone. Just press the left mouse button on the Rapise Play button and take your hand away from the mouse.

The script does not press the OK button in the TwoDialogs AUT.

Now, press the Play button on the Rapise and quickly move the mouse to hover over the OK button in

the TwoDialogs AUT. Now it works!

What's going on here is that the DoLButtonDown() and DoLButtonUp() methods are pressing the mouse irrespective of where the mouse cursor is positioned.

The other functions, DoClick and DoAction are methods that are applied to the button and so they are applied to the button.

Before we can expect DoLButtonDown() and DoLButtonUp() methods to work, we have to first the mouse cursor to the button.

```
function Test()
{
    SeS("OK").DoMouseMove(25, 15);
    SeS("OK").DoLButtonDown();
    SeS("OK").DoLButtonUp();
}
```

will accomplish that.

Notice that Rapise will actually move the mouse to the coordinates (25, 15) within the OK button. Also notice that if you move the mouse while the test is playing, you will make the test fail.

As a last experiment in this arena, try moving the mouse outside the boundaries of the OK button object before calling the DoLButtonDown() function.

```
function Test()
{
    SeS("OK").DoMouseMove(250, 150);
    SeS("OK").DoLButtonDown();
    SeS("OK").DoLButtonUp();
}
```

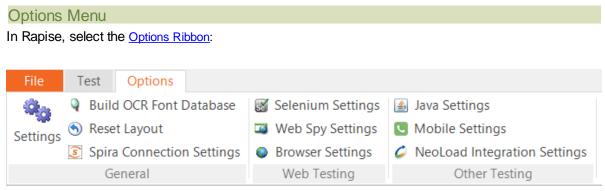
Once again, the script will fail.

2.5.4 Restoring the Default Layout

There are two ways to the restore the default layout: (1) On Startup, and (2) In the Options Menu.

On Startup

Press the Shift key while you open Rapise. Keep the Shift key down until Rapise is done initializing.



The Options Ribbon will appear. Click on the Reset Layout button under the General tab.

Rapise will restart and the layout will be returned to the default.

2.5.5 Change Test Entry Point

Rapise assumes that the entry point of a test - <u>Test()</u> function is defined in a file specified in <u>ScriptPath</u> property of the <u>Settings</u> dialog. If you want to place <u>Test()</u> function in another file then do not forget to update ScriptPath property of the test.

2.5.6 Deal with a Simulated Object

Example: The toolbox of Microsoft's Paint utility (c:\windows\system32\mspaint.exe) is a compound object that contains custom buttons and is surrounded by a containing box. To understand this completely, start mspaint.exe from the Rapise.

Steps:

(1) Open a new test under Rapise.

(2) Press the Record/Learn button on the application bar.

(3) When the "Select an Application to Record" dialog appears, select the Run Application tab. Enter mspaint in the "Full path to application" edit box. Press the Run button.

If you are unfamiliar with MS Paint, take a few minutes to play with it.

In particular, notice the toolbox that appears in the upper-left margin of the utility and the color selection box that appears on the bottom-left of the application window.

(4) Press Ctrl+5 to spy on the UI. Press Ctrl+G to spy on the Paint application. Notice several things about the behavior of the MS Paint application under the Object Spy.

(i) As you move the mouse inside the tools box, the entire surrounding box will show a red highlight but the individual tool buttons will not.

(ii) The same is true of the color palette and the bottom-left of the screen.

(iii) As you move the mouse over the apparent buttons and controls, the information in the spy dialog is more sparse than for other applications. The tool buttons do not have default actions, and they are not identified as buttons. Rather they are identified only as "child" objects.

This combination makes it impossible for Rapise to identify and learn the objects as integral objects.

Furthermore, notice that as you change the size of the Paint window, the relative positions of the color palette and the tool box change.

The only way in which Rapise can be 'taught' these controls (and others we will discover later) is by "simulating" them as though they were buttons that can accept commands such as the press event.

In fact, Rapise will recognize these non-objects without you having to take particular action. Let's discover this and what it means:

(1) Open a new test under Rapise; call it MSPaint.

(2) Press the Record/Learn button on the application bar.

(3) When the "Select an Application to Record" dialog appears, clear all selection boxes in the library list box. You will have to scroll that section of the dialog box to make sure all selections are clear. We are choosing no loaded libraries so that Rapise will not be able to "cheat" and know about any objects on the screen.

(4) select the Run Application tab. Enter mspaint in the "Full path to application" edit box. Press the

Run button.

(Applications that reside in C:\windows\system32 can be started by their names because C:\windows \system32 must be in the system path.)

- (5) When the Recording Activity dialog is displayed, press Learn (Ctrl+2)
- (6) Do a small amount of things in Paint. For example:
 - (i) Click on the light-grey color in the palette.
 - (ii) Click on the tipping paint-can (Fill with color).
 - (iii) Click on the empty canvas.
 - (iv) Click on the red color in the palette.
 - (v) Click on the "A" tool (Text).
 - (vi) Click in the canvas and type a few characters, such as "Hello."
 - (vii) Click in a blank place under the tool button.
- (7) Look at the Recording Activity dialog grid. It will be something like this:

#	Object	Action	Data	Comment
18	Colors	LClick	172,84	User clicks at: 172, 84 in 'Colors'
🖄 3	Fill with color	LClick	5,10	User clicks at: 5, 10 in 'Fill with color'
🖄 3	Simulated	LClick	422,111	User clicks at: 422, 111 in "
🖄 4	Colors	LClick	158,84	User clicks at: 158, 84 in 'Colors'
10 🖏	Tools	LClick	45,82	User clicks at: 45, 82 in 'Tools'
0 🖏	Simulated	LClick	336,89	User clicks at: 336, 89 in "
107	Tools	LClick	37,83	User clicks at: 37, 83 in 'Tools'
8 🍏	Text	LClick	373,169	User clicks at: 373, 169 in 'Text'
🖄 9	Simulated	LClick	267,165	User clicks at: 267, 165 in "
🖄 10	Untitled - Paint	SendK	Hello	Туре
11	Global	SendK	(ENTER)	Туре
	y (Ctrl+1)	Learn (Ctrl	+2) 😳 Simulated	Spy(Ctrl+5) Pick Object Pause

Notice that the two clicks in the canvas were recorded as "simulated" objects.

Notice also that the two pairs of clicks in the tools and colors sections were recorded as LClick (left click) in "Tools" and "Colors". However, there are no objects by these names. To find out where these pseudo objects came from, we need to look in the file MSPaint.objects.js (the name will be the name you gave the test project). The following excerpt from the MSPaint.object.js shows the start of the definition of the "Colors" object:

```
var saved script objects={
        "Colors":{
        "locations": [
            {
                "locator name" . "Location",
                "location": {
6
                   "window name": "param:window_text",
                   "window class": "param:window class"
               }
        11
12
            {
                "locator name": "LocationPath",
13
               "location": {
14
        » » » "window name": "param:window text",
15
                   "window class": "param:window_class",
16
               » "path": [
17
18
                    » {
                           "object_name": "param:object_name",
19
                           "object class": "param:object class",
                           "object role": "param:object role"
21
                       },
22
23
                       {
24
                           "object_name": "param:object_name",
2.5
                           "object class": "param:object class",
                           "object role": "ROLE SYSTEM WINDOW"
26
27
                       },
28
                       {
29
                           "object_name": "param:object_name",
                           "object class": "AfxControlBar42u",
                           "object role": "param:object role"
31
               » » },
```

(8) Press Ctrl+3 to end the recording.

2.5.7 Do Absolute Analog Recording

Let's once again use our trusty over-simplified TwoDialogs sample application to learn how to use absolute analog recording and use it to discover the value as well as the dangers associated with absolute analog recording.

Steps:

(1) Run the TwoDialogs sample AUT. By default this will be located in the C:\Users\Public\Documents \Rapise\Samples\TwoDialogs\TwoDialogs.exe location

(2) Start Rapise and create a new test and call it TwoDialogsAnalogAbsolute.

(3) Press the Record/Learn button in the toolbar of Rapise.

(4) When the "Select an Application to Record" dialog is displayed, choose the TwoDialogs.exe application and ignore the library list - we will not be using any library for analog recording. Press the Select button.

(5) The Recording Activity dialog will be displayed with an empty grid.

NOTE: this recording session is going o be a little different from previous sessions. Previously we could interrupt our object-related recording/learning with other activities and because Rapise was recording activity related only to the target application, our recording or object learning would be

unaffected. However, in analog recording, Rapise is monitoring the mouse and keyboard for the entire system - for all applications. This means that if you answer an email in the middle of analog recording, or log in to a secure system, all the steps including mouse movement, keystrokes, etc., will all be recorded. However, note also that screen contents are not recorded by Rapise.

(6) If the TwoDialogs UI has been occluded, bring it back to the front so you don't have to hunt for it when you start recording.

(7) When you're ready to record the session, hit Ctrl+4 on the Recording Activity dialog. You will need to click on the 'Advanced' button to display the Analog button.

Recording Activity for "TwoDialogs"

#	Object	Action	Data	Comment			
	Press Ctrl+Break to Stop Analog Recording						
Your mou	ise and keybo	ard activity is r	iow being reco	orded		Advanced>>	Transparent

NOTE: Pressing the Analog button on the Recording Activity dialog starts a relative analog recording session. Use the Ctrl+4 key sequence to start the absolute analog recording session.

Rapise will begin recording all mouse and keyboard activity until you stop the recording.

Note also that the prompt in the notification/status area of the Recording Activity dialog is different from that for relative analog. It tells you that "Your mouse and keyboard activity is now being recorded."

A minimized window will be created that indicates that analog recording is in progress and allowing you to stop the recording.

🖉 Analog Recording 🛛 🗖 🔳 🔤
To finish recording/playback use 'Ctrl-Break' shortcut or press 'Close' button.
Close

(7) Go to the TwoDialogs AUT and click anywhere in the application's window to start the analog recording.

Click the mouse on the empty "Please enter your name" text box.

Type a name in the text box.

Hit the <tab> key or click the left mouse button to advance the input position to the second text box.

Type another name.

Move the mouse to the OK button and press the mouse left button.

(8) When you have recorded enough, switch to the Analog Recording dialog box and press the close button or hit the keys Ctrl+Break.

NOTE: If you use the "close" button on the Analog Recording dialog, the movement of the mouse to the Analog Recording dialog, and the mouse-click on the Close button will be recorded as part of the

analog recording output. This might not be a desirable outcome at playback time because the Analog Recording dialog will not be present and the mouse click will be played in a potentially random place on the screen. For this reason, Ctrl+Break is probably a better option to terminate analog recording.

NOTE: The grid will have no entry added until you end the analog recording with the Close button in the Analog Recording dialog. When you do, it will add an entry to the grid.

# 1	Object Global	Action Anal	Data Analog∖A	Comment Plays recorded	d events from "An	alog\Ana	log0001.arf' file	9
	/erify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume		Finish (Ctrl+3)	Cancel
ŀ	Analog (Ctrl+4))		_Simu	ulated	•		

(9) You can now record additional analog sessions, if you wish.

(10) You can record normal object activity before and/or after the analog recording. When you have finished all recording press the Finish button or hit Ctrl+3. Notice that the Analog entry is added to the grid.

(11) The Rapise screen will now be restored and will have placed focus in the editor pane of the Rapise with TwoDialogsAnalogAbsolute.js script displayed. You should see code something like the following:

//Plays recorded events from "Analog\Analog0003.arf" file
SeS('Simulated').DoAnalogPlay("Analog\\Analog0003.arf");

(12) Press the Play button on the Rapise toolbar to playback the recording you made. Be sure not to interfere with the mouse or keyboard whilst the recording is playing back.

NOTE: You will see all mouse and keyboard activity reproduced as the analog recording plays. The recording will start from the point where you left-clicked the mouse to begin the recording (step 7 above) and will end with clicking the close button in the Analog Recording dialog or at the last action before you pressed Ctrl+Break.

(13) When the analog playback is complete, use the mouse to move the Two Dialogs AUT to a different location on the screen. Play the recording again, and watch the operation unfold. The most important thing to realize is that the absolute analog recording will playback the recording wherever the application is positioned on the screen wherever the AUT was positioned when you made the recording. Absolute analog recording records relative to the top-left corner of the system screen. Try this for yourself, but be sure to minimize all applications before starting.

2.5.8 Do Relative Analog Recording

Let's once again use our trusty over-simplified TwoDialogs sample application to learn how to use relative analog recording.

Steps:

(1) Run the TwoDialogs sample AUT. By default this will be located in C:\Users\Public\Documents \Rapise\Samples\TwoDialogs\TwoDialogs.exe

(2) Start Rapise and create a new test and call it TwoDialogsAnalogRelative.

(3) Press the Record/Learn button in the toolbar of Rapise.

(4) When the "Select an Application to Record" dialog is displayed, choose the TwoDialogs.exe application. Since we will not be using a library for this recording, the library selection is irrelevant. Press the Select button.

(5) The Recording Activity dialog will again be displayed with an empty grid.

NOTE: this recording session is going o be a little different from previous sessions. Previously we could interrupt our object-related recording/learning with other activities and because Rapise was recording activity only related the target application, our recording or object learning would be unaffected. However, in analog recording, Rapise is monitoring the mouse and keyboard for the entire system - for all applications. This means that if you answer an email in the middle of analog recording, or log in to a secure system, all the steps including mouse movement, keystrokes, etc., will all be recorded. However, note also that screen contents are not recorded by Rapise.

(6) If the TwoDialogs UI has been occluded, bring it back to the front so you don't have to hunt for it during recording.

(7) When you're ready to record the session, hit the Analog button on the Recording Activity dialog.

NOTE: The key sequence Ctrl+4 starts an absolute analog recording session. Press the Analog button to start the relative analog recording session.

When you press the Analog button, two things will happen. Firstly, the status bar of the Recording Activity dialog will change to read, "Click on object to start analog recording."

Reco	ording Activit	ty for "Two	Dialogs"				
#	Object	Action	Data	Comment			
	Verify Ctrl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume	Finish (Ctrl+3)	Cancel
	Analog (Ctrl+4			_Simu	lated	•	
Click c	on object to sta	irt analog re	cording			Advanced>>	Transparent

After the next mouse click, Rapise is recording all mouse and keyboard activity until you stop the recording.

Secondly, a minimized window will be created that indicates that analog recording is in progress and allowing you to stop the recording.

🖉 Analog Recording	
To finish recording/playl shortcut or press 'Close	
Clos	e

(7) Go to the TwoDialogs AUT and click anywhere in the application's window to start the analog recording.

Click the mouse on the empty "Please enter your name" text box.

Type a name in the text box.

Hit the <tab> key or click the left mouse button to advance the input position to the second text box.

Type another name.

Move the mouse to the OK button and press the mouse left button.

(8) When you have recorded enough, switch to the Analog Recording dialog box and press the close button or press the key sequence Ctrl+Break. If you use the "close" button on the Analog Recording dialog, the movement of the mouse to the Analog Recording dialog, and the mouse-click on the Close button will be recorded as part of the analog recording output. This might not be a desirable outcome at playback time because the Analog Recording dialog will not be present and the mouse click will be played in a potentially random place on the screen. For this rason, Ctrl+Break is probably a better option to terminate analog recording.

NOTE: The grid will have no entry added until you end the analog recording with the Close button in the Analog Recording dialog. When you do, it will add an entry to the grid.

Recor	Recording Activity for "TwoDialogs"							
# ⊚1	Object Global	Action Anal	Data Analog\A	Comment Plays recorded	l events from "An	alog\A	nalog0001.arf' file	9
	′erify trl+1)	Learn (Ctrl+2)		SPY (Ctrl+5)	Resume		Finish (Ctrl+3)	Cancel
A	nalog (Ctrl+4))		_Simu	ılated	-		
Unknow	/n:						Advanced>>	Transparent

(9) You can now record additional analog sessions if you wish.

(10) You can record normal object activity before and/or after the analog recording. When you have finished all recording press the Finish button or hit Ctrl+3.

(11) The Rapise screen will now be restored and will have placed focus in the editor pane of the Rapise with TwoDialogsAnalogAbsolute.js scrip displayed. You should see code something like the following:

//Plays recorded events from "Analog\Analog0003.arf" file

SeS('Simulated').DoAnalogPlay("Analog\\Analog0003.arf");

(12) Press the Play button on the Rapise toolbar to playback the recording you made. Be sure not to interfere with the mouse or keyboard whilst the recording is playing back.

NOTE: You will see all mouse and keyboard activity reproduced as the analog recording plays. The recording will start from the point where you left-clicked the mouse to begin the recording (step 7 above) and will end with clicking the close button in the Analog Recording dialog. If you used Ctrl +Break to end the recording then the last recorded activity will be the one that keystroke.

(13) When the analog playback is complete, use the mouse to move the Two Dialogs AUT to a different location on the screen. Play the recording again, and watch the operation unfold. The most important thing to realize is that the relative analog recording will playback the recording wherever the application is positioned on the screen. This is because you used relative analog recording. However, once the recording within the AUT is complete, all mouse motion and keyboard strokes are relative to the current position of the AUT. Suppose that during analog recording, you click the OK button in TwoDialogs.exe, then move the mouse to terminate the recording using the analog recording Close button. Now, prior to playback, you move the AUT to a different location on the screen and hit playback. All the activity within the AUT will be faithfully reproduced. However, the mouse motion outside the AUT will be relative to the position, so the following activities will not be accurately reproduced. Try this for yourself, but be sure to minimize all applications before starting so you don't cause mouse events where they will do harm to other applications on the screen.

2.5.9 Changing the URL of Website being Tested

Overview

We are in the process of adding a new testing server that we would like to be able to run automation scripts against.

I just wanted to see if there is a process documented for converting scripts and objects to point to a different web address?

Recommended Solution

There are several ways of doing this, but the way that makes your scripts easier to modify in the future is simply to add a global variable to your test script:

g_base_url = 'http://www.libraryinformationsystem.org';

and then open up the MyTest.objects.js file and change all references of:

"url": "http://www.libraryinformationsystem.org/Books.aspx"

to

```
"url": g_base_url + "/Books.aspx"
```

(and similarly for other URLs).

Then you can just change the base URL in either the script or pass through a value through the Rapise command-line or from SpiraTest (as a parameter).

Alternatively you can keep the script the same and just add the following code at the start of the test:

Navigator.Open('http://www.website.com');

or using a variable:

Navigator.Open(g_webSiteBaseUrl);

2.5.10 Extracting test data from an Excel spread sheet

Overview

Often you want to be able to parameterize your Rapise tests to have a common set of test functions that can use different combinations of test data. You can use a MS-Excel spreadsheet to store the test data and use Rapise to read out the matching values. This articles provides a sample for doing this.

Recommended Solution

In this example we have a spreadsheet that contains some lookup data:

Test Test Data

```
Test1valuetest1Test2valuetest2Test3valuetest3Test4valuetest4Test5valuetest5
```

We want to dynamically query this Excel sheet and find the test data associated with a specific case. For example if we query for "Test2" we want to return back the test data "valuetest2".

The function that will do this uses the built-in Spreadsheet object:

```
function FindValueFromFile(filename, valueToFind)
 //Open the spreadsheet
var success = Spreadsheet.DoAttach(filename, 'Sheet1');
 Tester.Assert('Open Spreadsheet', success);
 //Now loop through and see if we can find that value
 var rowCount = Spreadsheet.GetRowCount();
 Spreadsheet.SetRange(2, rowCount + 1, 1, 2);
 //Loop through all the rows and find the match
 var data = '';
 while(Spreadsheet.DoSequential())
 ł
  if (Spreadsheet.GetCell(0) == valueToFind)
  data = Spreadsheet.GetCell(1);
  }
 }
 return data;
```

2.5.11 Accessing Files and I/O Functions

Overview

Sometimes using Rapise you need to access files on the Windows filesystem, either for testing purposes, or to object test data stored in a flat CSV, TSV or text file. This sample illustrates how you can access the File System using Rapise and the Windows FileSystemObject (<u>http://msdn.microsoft.com/en-us/library/aa242706(v=vs.60).aspx</u>).

Code Sample

The following sample code illustrates how to access files using Rapise:

```
function Test()
{
    //IOMode constants
    var IOMode_ForReading = 1;
    var IOMode_ForWriting = 2;
```

```
var IOMode ForAppending = 8;
    //var Format constants
    var IOFormat_ASCII = 0;
    var IOFormat Unicode = -1;
    //This sample demonstrates how to manipulate the Windows
file system using Rapise
    //We shall open a text file for writing in this simple
example
    var fso = new ActiveXObject('Scripting.FileSystemObject');
    var ts = fso.CreateTextFile('C:\\Temp\\MyTestFile.txt');
    ts.Close();
    var file = fso.GetFile('C:\\Temp\\MyTestFile.txt');
    ts = file.OpenAsTextStream(IOMode ForWriting,
IOFormat Unicode);
    ts.WriteLine('Hello World!');
    ts.Close();
    Tester.Message('Wrote File');
    //Now read this file back
    file = fso.GetFile('C:\\Temp\\MyTestFile.txt');
    ts = file.OpenAsTextStream(IOMode_ForReading,
IOFormat_Unicode);
    var text = ts.ReadLine();
    Tester.Message(text);
    ts.Close();
}
```

2.5.12 Sending Special Keys to the Current Application

Overview

You can use the Global.DoSendKeys('...'); command in Rapise to send keypresses to the current application. Sometimes you will want to send special control keys to the application (e.g. Page Down, or CTRL + Key). This article explains the way to do this.

Information and Examples

To send special characters, you just use the list available in the Windows API SendKeys.Send function:

http://msdn.microsoft.com/en-us/library/system.windows.forms.sendkeys.send(v=vs.110).aspx

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the codes in the following table:

Key

BACKSPACE BREAK Code {BACKSPACE}, {BS}, or {BKSP} {BREAK}

CAPS LOCK	{CAPSLOCK}
DEL or DELETE	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS or INSERT	{INSERT} or {INS}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
PRINT SCREEN	{PRTSC} (reserved for future use)
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}
Keypad add	{ADD}
Keypad subtract	{SUBTRACT}
Keypad multiply	{MULTIPLY}
Keypad divide	{DIVIDE}

For example, to send the ENTER keypress, just use:

Global.DoSendKeys('{ENTER}');

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key

code with one or more of the following codes:

	Key	Code
SHIFT		+
CTRL		٨
ALT		%

2.5.13 Detecting the presence of an object

Often you need to be able to check for the presence of an object and then depending on whether the object is visible perform one of two possible sets of operations. This article explains the recommended way of doing this

Recommended Solution

You can use the special SeSFindObj('object name') command to perform the check:

```
if(SeSFindObj('Object Name'))
{
     // Object is available
}
else
{
     // Object is not available
}
```

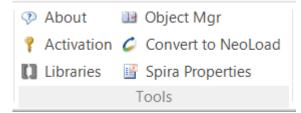
2.5.14 Change the libraries being used

Sometimes you are recording a test using a <u>library</u> (e.g. Firefox HTML) and you realize that you need to add additional libraries to the test to make the recording more efficient.

For example, you are recording using the Firefox web browser and you need to include the <u>Microsoft</u> <u>Dynamics 365</u> libraries so that you can more easily record actions in Dynamics 365 while using Firefox. The additional library will allow Rapise to recognize the grids and treeviews as Dynamics objects rather than just standard HTML tables or DIVs.

How to Change the Libraries

To change the recording library being used, click on the [] Libraries icon in the Tools section of the Test ribbon:



That will bring up the Recording selection dialog box:

ActiveXC1	rests incli, wer and onvenight Applications		
	Support for Component One 8.0 ActiveX Librari		
	Stepport for component one of Active Active Active Station		
☐ Web Service	Used for testing RESTful web services		
	UI Automation support for Java SWT		
Internet Explorer HTML	HTML DOM-based recorder for Internet Explorer		
Selenium	Selenium based playback engine		
Selenium Firefox HTMI	HTML DOM-based recorder for Mozilla Firefox		
User	Default user-defined library		
Chrome HTML	HTML DOM-based recorder for Google Chrome		
DOM jQuery UI	DOM-based recorder for jQuery UI widgets		
	Console Application		
	DOM-based recorder for GWT		
	Support for HTML5 new elements		
DOM GWT-Ext	DOM-based recorder for GWT-Ext		
	DOM-based recorder for YUI		
DOM SmartGwt	DOM-based recorder for SmartGWT		
DOM Spira	DOM-based recorder for Spira UI widgets		
DynamicsNAV	Dynamics NAV controls		
DomDynamicsAX	Dynamics AX7+ support library		
DomDynamicsCrm	Dynamics CRM support library		
DomDynamicsCim DomSalesforce	SAlesforce support library		
DynamicsAX	Dynamics AX controls		
	Tests common Windows Win32 controls		
☐ FarPoint	Low level FarPoint Grid Support		
_ FarPoint	Android and iOS test automation library		
Advanced Accessibility	For experts only. This library should be used to		
SeSIntegration	SmarteScript Integration		
	Smartescript megration		
Cancel		_	

Now you can check the boxes of the additional libraries you want to use (or remove any existing ones that you no longer need).

Keyboard Shortcuts

In addition, there are a couple of shortcuts that you may want to use:

- **Record** clicking Record on its own will start Rapise recording using the current selected libraries (this is the default action)
- Shift+Record This brings up the library selector (shown above), and is equivalent to clicking the [] libraries button in the ribbon before starting recording.
- CTRL+Record This brings up the application selector (see below) and lets you change the application and library being recorded:

Available Applications	PID	Path
	7032	C:\Windows\explorer.exe
Image: A state of the state	3744	C:\Program Files\Google\GoogleToolbarN
🔤 C:\Windows\system32\cmd.exe	4796	C:\Windows\system32\cmd.exe
🥝 Help & Manual	7024	C:\Program Files\EC Software\HelpAndM
🚉 Program Manager	7032	C:\Windows\explorer.exe
Sample ATM Login - Windows Internet Ex	7000	C:\Program Files\Internet Explorer\iexplor

2.6 Technologies

This section focuses on specific technologies supported by Rapise.

2.6.1 Web Testing

Purpose

Rapise lets you record and play automated tests against web applications on a variety of web browsers including **Firefox**, **Internet Explorer** and **Google Chrome**. Rapise lets you record or create your tests against one browser and then play the same test back against all of the other browsers. Some of the web browsers will require the <u>installation of a special Rapise plugin</u> to allow automated testing.

Rapise provides comprehensive support for testing Web applications. Rapise supports cross-browser testing. It uses the web browser Document Object Model (DOM) to interact with the current web page. The various web browsers on the market have various differences in DOM implementation. In many cases these differences are not significant. But sometimes they require special handling. Rapise tries to overcome the differences and make the recorded scripts as universal as possible:

Cross Browser Testing

When developing and testing a web application you naturally need to test it with different web browsers and of course (based on bitter experience) multiple version of each web browser. With Rapise you can record a test script using one browser and then play it back using Mozilla Firefox, Google Chrome or Microsoft Internet Explorer:



With Rapise, you can run your recording in a different browser than the one in which it was recorded simply by changing the specified browser in the playback settings:

Sett	tings		џ
⊿	Advanced		*
	CommandLine		
	EntryPoint	Test	
⊿	TestParams	5 items	Ξ
	Browser	Internet Explorer HTML	
	Record Title	Internet Explorer HTML	
	ServerProjectId	Firefox HTML	
	ServerTestCaseFolde	Chrome HTML	
	ServerTestCaseId	ъ	
⊿	Execution		
	CacheObjects	False	
	CommandInterval	1000	
	Iteration Count	1	
	Object aclous Attemption	200	4

In addition, it is possible to have more control over the cross browser execution using the available APIs and configuration variables. You can also run the recording in multiple browsers in succession using either a Rapise <u>sub-test</u> or simply executing the test from our <u>SpiraTest test management</u> system and passing through different parameter values.

DOM API

In addition to the usual <u>recording</u>, <u>learning</u> and <u>playback</u> that is similar to testing other technologies (desktop, mobile, etc.) there are some unique functions that are available on all recorded web objects:

Upon learning a web element in Rapise, you get an object of type **HTMLObject**. Each **HTMLObject** provides set of functions to facilitate the cross-browser access to web element parents and children.

DOM Function	Description
DoDOMChildAt	Returns n-th child (zero-based).
DoDOMChildrenCount	Returns number of children elements for this one.
DoDOMFindParentWithAtt ribute	Returns parent element (if any) with given attribute matching given string or 'regex'
DoDOMGetAttribute	Returns specified attribute.
DoDOMNextSibling	Returns next sibling element for this one.
DoDOMParent	Returns parent element having this element.
DoDOMPrevSibling	Return previous sibling element for this one.
DoDOMRoot	Returns Root element having this element.

- Web Spy How to use the Web Spy to inspect web pages and dynamically query for HTML elements
- <u>XPath</u> An explanation of the XPath language, how it can be used to dynamically query objects in web application and some examples
- <u>CSS</u> An explanation of how to use CSS selectors (common in frameworks such as jQuery) to dynamically query objects in web applications
- <u>Web Testing Tutorial</u> a basic example of how to record, learn and playback tests using a our sample web application
- <u>Setting up Web Browsers</u> describes the steps needed to configure each of the web browsers to work with Rapise.

2.6.1.1 Cross Browser Testing

Choosing the Browser When Creating a Test

When you first create a Rapise test with the **Methodology** set to **Web** you will be asked to choose the initial <u>web browser profile</u>:

eb Test Setup	A REAL PROPERTY AND ADDRESS OF	
Specify Web Profile:	Chrome HTML	
	Chrome HTML	
	Firefox HTML Internet Explorer HTML Selenium - Chrome Selenium - Firefox	

You can run your recording in a different browser than the one in which it was recorded.

Selecting a new Playback Browser

First, open the script for your test using the <u>Test Files Dialog</u>. Locate the line where **g_load_libraries** is initialized.

Under the Hood

It is possible to have more control about cross browser execution using available APIs and configuration variables You can also run the recording in multiple browsers in succession. Both options require modification of the script. The necessary modifications are described below. First, open the script for your test using the <u>Test Files Dialog</u>. Locate the line where **g_load_libraries** is initialized.

If you recorded your script in IE you will see:

g_load_libraries=["%g_browserLibrary:Internet Explorer HTML%"];

If you recorded it in Firefox, you will see:

g_load_libraries=["%g_browserLibrary:Firefox HTML"];

This line tells Rapise to use the browser library specified in the special g_browserLibrary variable

setting, and if no value is set, default to the named browser (Internet Explorer or Firefox in this example).

Changing the Playback Browser

Go to the Main <u>Start Page</u> of Rapise. Under the 'Current Test' section you will see the list of available web browsers:

Current Test		
Name:	Scenarios Test	Test Description
Scenario:	Test V Play	
User Functions:	Scenarios Test.user.js	
Script:	<u>Scenarios Test.js</u>	
Parameters Spira Properties		
Browser:	Selenium - Chrome	\checkmark
Record Title:	Chrome HTML	1
	Firefox HTML	
	Internet Explorer HTML Selenium - Chrome	
	Selenium - Firefox	
	Selenium - Internet Explorer	
🕫 📄 🔷 🗎	Selenium - Opera Selenium - Safari	

Change the browser to either one of the <u>native browsers</u> such as **Firefox**, **Internet Explorer** or **Chrome**, or one of the <u>Selenium WebDriver</u> based browser profiles.

Once you have changed this setting, <u>Playback</u> the script normally and it will playback in the selected browser.

Changing this setting will effectively set the value of the **g_browserLibrary** global variable.

Playback in Multiple Browsers - SpiraTest

Executing a test in multiple browsers is slightly more complicated. We recommend that you use **SpiraTest 'Test Sets'** where you may define multiple test cases pointing to the same Test with a different **g_browserLibrary** parameter value.

The separate help document "Using SpiraTest with Rapise" provides specific instructions on using Rapise with SpiraTest to handle the specific case of cross-browser testing as well as more general support for parameterized testing.

See the <u>SpiraTest Integration</u> topic for more general information on using Rapise with SpiraTest.

Playback in Multiple Browsers - SubTests

As another option, it is also possible to use <u>sub-tests</u> to organize multi-browser testing where a single test executes itself in different browsers one after another.

- 1. Record <u>base</u> test. Put all the recorded actions into a User-defined function and place it into <testname>user.js file. For example, function Login() inside file MyTest.user.js.
- 2. <u>Create Sub-Test</u> for IE re-using objects and functions from the base test
- 3. Modify script file in sub-test as follows:

```
function Test()
{
    // Re-use 'Login()' scenario from parent test
    Login();
}
```

g_load_libraries=["Internet Explorer HTML"];

Create Sub-Test for Firefox re-using objects and functions from parent test
 Modify script file in subtest as follows:

```
function Test()
{
    // Re-use 'Login()' scenario from parent test
    Login();
}
g_load_libraries=["Firefox HTML"];
```

As a result you have a test for 2 browsers: **IE** an **Firefox**. Each browser is defined by a library in a corresponding sub-test. Rapise contains the <u>Cross Browser</u> sample using this approach.

2.6.1.2 Setting Up Web Browsers

Before you can use Rapise with certain web browsers to do web testing, you will need to install plugins so that Rapise can communicate with them.

Preparing Internet Explorer (IE)

Unlike the other web browsers, there are actually no steps needed to configure IE. Once Rapise is installed, it is ready to connect to IE for recording, playback and learning without any configuration steps.

Preparing Firefox

In order to test web applications using the Firefox web browser, you will need to install an extension for Firefox that allows Rapise to interact with it:

 Launch Firefox. Navigate to the following URL: <u>http://www.inflectra.com/Rapise/</u> <u>Downloads.aspx</u>:

Available Downloads

The following downloads are currently available for Rapise:

Web-Browser	Extensions	
⊌ኅ v2.0.0 ⊾ <u></u>	Rapise Add-On For Firefox Allows Rapise to record, learn and playback automated tests using the Mozilla Firefox web browser. Compatible with Mozilla Firefox 20+ and Rapise v2.0+	*
©ኅ v2.0.3 ፝⊾ <u>₽</u>	Rapise Extension for Google Chrome, Allows Rapise to record, learn and playback automated tests using the Google Chrome browser. Compatible with Google Chrome 32.0+ and Rapise v2.0+	*

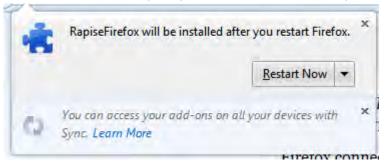
2. Click on the 'Rapise Add-In For Firefox' hyperlink and it will take you to the Firefox Add-Ons page:

RapiseFirefox 2.0 by Inflectra Corporation Firefox connector for Rapise	Not yet rated O user reviews 1 user
+ Add to Firefox End-User License Agreement This add-on has been preliminarily reviewed by Mozilla. Learn more	□ Add to collection < Share this Add-o
	3

- 3. Click on the 'Add to Firefox' button to install the Add-On into your instance of Firefox.
- 4. Click "Install Now" in the software installation dialog that appears:

tware Inst	allation			
À		dd-ons only from a software can damage yo		
You have	asked to insta	all the following item:		
		: (Author not verified) s.mozilla.org/firefox/dow	vnloads/file/254042/ra	pisefirefox-2.0-fx-windov
			F	stall Now Cancel

5. When installation is complete press "Restart Now" to complete the installation:



6. Once Firefox is started again, you should now see the RapiseFirefox Add-On listed in the 'Extensions' section of the Firefox Add-ons Manager:



7. Firefox is ready to be used with Rapise for automated testing.

Note: Rapise requires localhost port 4247 to be accessible for correct operation. Please, make sure that this port is unblocked in your Firewall.

Preparing Google Chrome for Rapise®

In order to test web applications using the Google Chrome browser, you will need to install an extension for Chrome that allows Rapise to interact with it:

1. Launch Chrome. Navigate to the following URL: <u>http://www.inflectra.com/Rapise/</u> <u>Downloads.aspx</u>:

Available Downloads

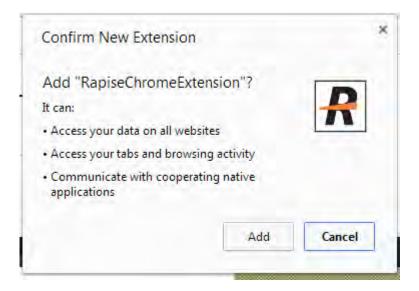
The following downloads are currently available for Rapise:



2. Click on the 'Rapise Extension for Google Chrome' hyperlink and it will take you to the Chome store:

OVERVIEW	DETAILS	REVIEWS	RELATED	g
	Tests Description Description Anticipation De			This extension allows the Rapis test automation platform to test web applications running in the Google Chrome browser. This extension allows the Rapise test automation system to record, learn and playback automated tests using the Goog
	 C. Self-Linear Department of the self-self-self-self-self-self-self-self-		(i) a set of the constrained operator (ii) a set of the constrained operator (iii) a set of the constrained (iii) a set of the constrained	
and an and	R 8			

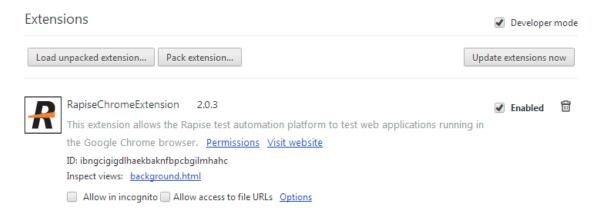
3. Click on the 'Free' button to install the Rapise extension into your instance of Chrome. You will be asked to confirm that you want to Add the extension:



4. Once the extension has been installed, you will see the following confirmation dialog box:

R	RapiseChromeExtension has been added to Chrome.	
	Sign in to Chrome to get this extension, your history, and other Chrome settings on all your devices.	

 To view the extension, click on the "wrench" icon next to the omnibox and choose Tools > extensions. You should now see the chrome extension manager with the Rapise extension listed:

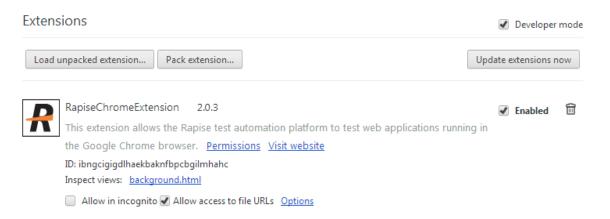


 It is recommended to set the "Allow access to file URLs" checkbox so that web pages opened from the local folder (using the file:// protocol) can be also accessed by Rapise when running automated browser tests.

Testing Local Applications

The Rapise Chrome extension is not loaded by Chrome if a web page is opened from the local folder (file://). To overcome this restriction make sure that "Allow access to local URLs" checkbox is set.

- 1. In Chrome type following URL: chrome://extensions If you don't see the omnibox (the field where to type URL) please press Ctrl+T to show new chrome window.
- 2. 2. Set "Allow access to local URLs" checkbox:



Testing Chromium Applications

The Rapise extension installation procedure differs in the case of testing Chromium applications. Below are the recommended steps to enable automation for such applications:

- Refer this <u>http://www.chromium.org/administrators/pre-installed-extensions</u> to install an extension for a Chromium application. Getting the ID of current version of the Chrome Extension is easy. Install it into a regular Chrome browser following the instructions above, and then check the information in chrome by following this URL: <u>chrome://settings/ extensionSettings</u>
- Choose the "Chrome HTML" library in the "Select an Application for Record" dialog in Rapise when recording tests rather than "Auto". The Rapise library auto-detection logic may fail because the executable name is not "chrome.exe".
- 3. After following the steps below to modify the Windows registry, Rapise should be able to record and learn the application correctly.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Google\Chrome\Extensions
\malnpehoofemabajdignefgdoppndaeb\path
TYPE: REG_SZ
VALUE: "C:\path\to\your\RapiseChrome.crx"
```

```
HKEY_LOCAL_MACHINE\SOFTWARE\Google\Chrome\Extensions 
\malnpehoofemabajdignefgdoppndaeb\version
```

TYPE: REG_SZ VALUE: "1.6.0"

4. For the playback to execute correctly you need to set the following line in the beginning of your <testname>.user.js file:

```
g_browserExecutablePath='C:\\the\\path\\to\chomium-based
\app.exe';
```

With that change in place, the recorded test should be able to play.

Testing Chrome Frame Applications

The following configuration enables testing of Chrome Frame applications by Rapise:

- A 'packed' copy of the RapiseChrome.crx extension is provided with the Rapise installation. It can be located in the "C:\Program Files (x86)\Inflectra\Rapise\Extensions\Chrome" folder of your installation.
- 2. Enter the following data to the Windows registry:

```
[HKEY_CURRENT_USER\Software\Google\ChromeFrame]
"EnableGCFProtocol"=dword:00000001
"IsDefaultRenderer"=dword:00000001
"AllowUnsafeURLs"=dword:00000001
[HKEY_LOCAL_MACHINE\SOFTWARE\Google\Chrome\Extensions
\malnpehoofemabajdignefgdoppndaeb]
"path"="c:\\path\\to\\RapiseChrome.crx"
"version"="1.6.0"
```

- 3. In the Rapise "Select Application to Record" dialog box, explicitly choose the "Chrome HTML" library (and some or all of DOM libraries: YUI DOM, GWT DOM, etc) rather than using "Auto".
- Record the test script as you would normally. Note that there should be no Chrome browser or any Chrome applications running when IE with Chrome Frame is started and recording is being performed.

You will need to make some changes to the recorded test to ensure that it plays correctly. Insert this line in the beginning of the <test>.user.js file:

```
g_browserExecutablePath="iexplore.exe";
```

Check to make sure you have Internet Explorer browser executable available at the specified path on your PC and correct it if necessary.

Using Other Browsers with Rapise

If you would like to test your applications using **other web browsers (Safari, Edge or Opera)** you can use the <u>Selenium WebDriver</u> libraries that are provided with Rapise. To <u>setup the Selenium web browser</u> libraries, please refer to <u>this topic</u>.

2.6.1.3 XPath

Purpose

When testing web applications you will often need to use XPath to query the browser DOM for elements based on the scenario under test. This section explains how you can use XPath queries with Rapise to make your browser testing more flexible and adaptive to changes on the screen.

XPath Fundamentals

XPath uses path expressions to select nodes in an XML document such as HTML. The node is selected by following a path or steps. The most useful path expressions are listed below:

Expression	Description
nodename	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
	Selects the current node
	Selects the parent of the current node
@	Selects attributes

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
bookstore	Selects all the child nodes of the bookstore element
/bookstore	Selects the root element bookstore
	Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore

Path Expression	Result
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang
// book[@lang='Engli sh']	Selects all book elements that have a lang attribute equal to 'English'
//book[text() ='Oliver Twist']	Selects all book elements that have the text 'Oliver Twist' as their inner content (i.e. <book>Oliver Twist</book>)

Rapise XPath Extensions

Web pages sometimes use HTML frames. The XPath works inside the frame contents. Rapise has a special syntax (that is not part of standard XPath) to combine multiple XPath statements into a single line:

```
//frame[@name='main']@@@@//a[3]
```

The special statement:

@@@

Is used as a separator for XPath statements pointing to constituent frames. The top-level frame is found by name 'main'

```
//frame[@name='main']
```

Then the frame's contents are searched for the third <a> element (i.e. 3rd link on a page).

There are several different ways to use XPath queries in Rapise

Using Web Spy

To most easily use XPath queries in Rapise, we recommend using the Web Spy tool:

//table//td[@style]	Test XPath	Learn
	Test CSS	Learn

Locators

If you enter in the XPath query at the top, when you click **Test XPath** it will display all of the DOM elements that match the query:

```
Test Results

▲ Locator: //table//td[@style], found: 4

...

...

...
```

You can now refine the query to only find the items you want to test.

Learning Objects

When you have created the query in the Web Spy that returns the HTML elements that you were expecting, you can click on the **Learn** button to learn that object. What this will do is create a new Rapise object in the <u>Object Tree</u> that maps to this specific XPath. That means that the "object" in Rapise is effectively a pointer to this specific XPath query.

For example, if you want to find a specific book in a grid of books, you can search by its name using XPath and the text() XPath expression described above, then learn this object as "Oliver_Twist" so that you can access it in your code as SeS("Oliver_Twist"). Every time you call a function on "Oliver_Twist", Rapise will use the learned XPath expression and use that to evaluate which HTML element in the web page to access.

Dynamic Queries

In addition to learning objects based on specific XPath, there are a set of general functions that can be used to query for objects in the web page:

SeS('Book_Management').DoDOMQueryXPath('tr/td[text()="Oliver Twist"]');

Will dynamically query for any HTML element that is a child of the learned "Book Management" object that matches the XPath. In this example it will look for any table cell in a table row that has the content of the book name.

2.6.1.4 CSS

Purpose

When testing web applications you will often want to use Cascading Style Sheets (CSS) selectors to query the browser DOM for elements based on the scenario under test. This section explains how you can use CSS selectors with Rapise to make your browser testing more flexible and adaptive to changes on the screen.

CSS is an alternative to <u>XPath</u> that is often better at selecting multiple elements from across different parts of the DOM Tree, unlike XPath which is strictly hierarchical. However since CSS is not always able to uniquely locate an object, when Rapise is used in recording mode, it will learn objects automatically using XPath.

CSS Fundamentals

In CSS, selectors are patterns used to select the element(s) you want to style. Here are the different operators that you can use in CSS selectors:

Selector	Example	Example description	CSS
<u>.class</u>	.intro	Selects all elements with class="intro"	1
<u>#id</u>	#firstname	Selects the element with id="firstname"	1
*	*	Selects all elements	2
<u>element</u>	р	Selects all elements	1
element,element	div, p	Selects all <div> elements and all elements</div>	1
element element	div p	Selects all elements inside <div> elements</div>	
<u>element>element</u>	div > p	Selects all elements where the parent is a <div> element</div>	
<u>element+element</u>	div + p	Selects all elements that are placed immediately after <div> elements</div>	
<u>element1</u> ~element2	p ~ ul	Selects every element that are preceded by a element	
[attribute]	[target]	Selects all elements with a target attribute	2
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"	2
[<i>attribute~=value</i>]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
[attribute=value]	[lang =en]	Selects all elements with a lang attribute value starting with "en"	2
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"	3
[<u>attribute\$=value</u>]	a[href\$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"	3
[<u>attribute*=value</u>]	a[href*="w3schoo ls"]	Selects every <a> element whose href attribute value contains the substring "w3schools"	3
:active	a:active	Selects the active link	1
::after	p::after	Insert content after every element	2
::before	p::before	Insert content before the content of every element	2
:checked	input:checked	Selects every checked <input/> element	3
:disabled	input:disabled	Selects every disabled <input/> element	3
:empty	p:empty	Selects every element that has no children (including text nodes)	3
:enabled	input:enabled	Selects every enabled <input/> element	3
:first-child	p:first-child	Selects every element that is the first child of its parent	
::first-letter	p::first-letter	Selects the first letter of every element	1
::first-line	p::first-line	Selects the first line of every element	1
:first-of-type	p:first-of-type	Selects every element that is the first element of its parent	

Selector	Example	Example description	CSS
:focus	input:focus	Selects the input element which has focus	2
:hover	a:hover	Selects links on mouse over	1
<u>:in-range</u>	input:in-range	Selects input elements with a value within a specified range	3
:invalid	input:invalid	Selects all input elements with an invalid value	3
<u>:lang(<i>language</i>)</u>	p:lang(it)	Selects every element with a lang attribute equal to "it" (Italian)	
:last-child	p:last-child	Selects every element that is the last child of its parent	
:last-of-type	p:last-of-type	Selects every element that is the last element of its parent	
<u>:link</u>	a:link Selects all unvisited links		1
:not(selector)	:not(p)	Selects every element that is not a element	3
:nth-child(n)	p:nth-child(2)	Selects every element that is the second child of its parent	3
:nth-last-child(n)	p:nth-last-child(2)	Selects every element that is the second child of its parent, counting from the last child	3
<u>:nth-last-of-</u> type(n)	p:nth-last-of- type(2)	Selects every element that is the second element of its parent, counting from the last child	
:nth-of-type(n)	p:nth-of-type(2)	Selects every element that is the second element of its parent	3
:only-of-type	p:only-of-type	Selects every element that is the only element of its parent	3
:only-child	p:only-child	Selects every element that is the only child of its parent	3
:optional	input:optional	Selects input elements with no "required" attribute	3
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range	3
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified	3
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified	3
:required	input:required	Selects input elements with the "required" attribute specified	
<u>:root</u>	:root	Selects the document's root element	3
::selection	::selection	Selects the portion of an element that is selected by a user	
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)	3
<u>:valid</u>	input:valid	Selects all input elements with a valid value	3
:visited	a:visited	Selects all visited links	

One limitation (as compared to XPath) is that there is not a way to select an element based on its contents. So it would not be possible to locate a cell in a grid (for example) based on the contents of the

cell. For that you would need to use XPath.

Rapise CSS Extensions

Since Rapise uses XPath as its primary means of locating an HTML element, when you **Learn** an object using CSS, Rapise will prefix the Locator (listed under the XPath property for that object in the <u>Object</u> <u>Tree</u>) with <u>CSS=</u> to let Rapise know that the locator is actually using a CSS selector.

css=html > body > form#ctl01 > div:nth-of-type(3) > div:first-of-type > div:first-of-

There are several different ways to use CSS selectors in Rapise

Using Web Spy

To most easily use CSS selectors in Rapise, we recommend using the Web Spy tool:

	Test XPath Lea	arn	
table td[style]	Test CSS Lea	arn	
Locators			

If you enter in the CSS selector at the top, when you click **Test CSS** it will display all of the DOM elements that match the selector:

Test Results	
▲ Locator: CSS=table td[style], found: 4	

You can now refine the query to only find the items you want to test.

Learning Objects

When you have created the query in the Web Spy that returns the HTML elements that you were expecting, you can click on the **Learn** button to learn that object. What this will do is create a new Rapise object in the <u>Object Tree</u> that maps to this specific CSS. That means that the "object" in Rapise is effectively a pointer to this specific CSS selector.

For example, if you want to find a specific book in a grid of books by its CSS class, style, ID or other attribute, you can search using the appropriate CSS selector, then learn this object as "Book_1" so that you can access it in your code as SeS("Book_1"). Every time you call a function on "Book_1", Rapise will use the learned CSS selector and use that to evaluate which HTML element in the web page to access.

Dynamic Queries

In addition to learning objects based on specific CSS selector, there are a set of general functions that can be used to query for objects in the web page:

```
SeS('Book_Management').DoDOMQueryCss('tr td[data=book1]');
```

Will dynamically query for any HTML element that is a child of the learned "Book Management" object that matches the CSS selector. In this example it will look for any table cell in a table row that has the attribute data="book1".

2.6.2 Selenium WebDriver

When developing and testing a web application you naturally need to test it with <u>different web browsers</u> and multiple version of each web browser. With Rapise <u>natively</u> you can record a test script using one browser and then play it back using **Mozilla Firefox**, **Google Chrome or Microsoft Internet Explorer**.

In addition, you can use Rapise with the open-source **Selenium WebDriver framework** to play back the same tests against other browsers such as **Apple Safari and Opera** (as well as IE, Firefox and Chome). You can also use Rapise to write <u>native Selenium code</u> for cases where you want to use existing Selenium WebDriver logic.

Playing & Recording Tests

Once you have <u>installed and configured the integration between Rapise and Selenium</u>, we shall discuss how to use Selenium with Rapise to record and play tests.

Now one of the important points is that there are some limitations as to the operations that can be performed using Selenium-based web browsers as opposed to the native browsers supported by Rapise:

Feature	Rapise Native Browser	Selenium Browser
Learn HTML Objects	Yes	(Only in Web Spy)
Record HTML Events	Yes	No
Playback HTML Events	Yes	Yes
Web Spy	Yes	Yes
Learn Java Applets	Yes	No
Learn Flex Controls	Yes	No
Learn Silverlight Controls	Yes	No
Manual Testing	Yes	No

So if you are planning on using Rapise to record a test script by clicking HTML objects and having Rapise create the script using the learned objects and adding the events (DoClick, SetText, etc.) then you will need to use one of the native browsers (Chrome, IE, Firefox) to create the test script. You can then playback the same test in either the native or Selenium browsers.

If you are planning on using Rapise to learn objects using the Web Spy, and then create the test script from those objects by either dragging the object methods and properties from the Object Tree into the test script or just using Intellisense to type the methods (DoClick, SetText, etc.) then you can use either a native or Selenium web browser just as easily.

For most users, the primary reason for using the Selenium web browsers will be to playback their tests on a greater number of browsers or to leverage existing Selenium WebDriver scripts created outside of Rapise (see section 3).

Managing the Selenium Profiles

Rapise allows you to maintain different profiles for your different installed Selenium web browsers (both on the same machine as Rapise and also those running on a remote Selenium WebDriver server), To see the different <u>Selenium profiles</u>, click on **Options > Tools**:

Options			
General Tools			
	Build OCR Font Database		
	Reset Layout		
	Java Settings		
	Mobile Settings		
	Selenium Settings		
	Spira Connection Settings		
	Web Spy Settings		
	Browser Settings		
		ОК	Cancel

When you click on the [Selenium Settings...] button, it will bring up the Selenium profile manager:

elect Profile:	Chrome	Save	Test Duplicate	Create Re	ename Delete
	Chrome Firefox Internet Explorer				
Connecti Uri					
⊿ Details					
Primary C	apabilities		(1)		
brows Common C	erName		Chrome (1)		
Primary Cap Desired capab	ilities are a set of keys and va	ilues (i.e., a map or hash) s	ent to the Selenium server to t	ell the server what kind	of automation
esired capab	ilities are a set of keys and va	liues (i.e., a map or hash) s re are also various capabil	ent to the Selenium server to t ties which can modify the beh	ell the server what kind avior of the server durir	l of automation ng automation.

By default there is one profile for each of the Selenium WebDriver supported web browsers (Chrome, Firefox, Internet Explorer, Opera, Safari). However you can clone and change the profiles if you want to have different versions of the browsers (e.g. a local instance of Firefox and one running on a remote Selenium server).

Most users will only need to change the **Uri** field of the Safari web browser since the defaults are typically sufficient for most testing needs.

Recording using Selenium

To start <u>recording</u> a web testing using a Selenium WebDriver based browser, make sure you change the test's web browser parameter to one of the Selenium profiles:

Parameters	
Spira Properties	
Browser:	Selenium - Firefox
Record Title:	Chrome HTML Firefox HTML Internet Explorer HTML Selenium - Chrome Selenium - Firefox Selenium - Internet Explorer Selenium - Opera Selenium - Safari

When you click the **Record/Learn** button in the main Test ribbon you will see the following <u>Recording</u> <u>Activity Dialog</u>:

Object	Action [Data	Comment		
Verify (Ctrl+1)	Learn (Ctrl+2	2)	Spy(Ctrl+5)	Pick Object	Resume

You will notice that the **Verify**, **Learn** and **Pick Object** options are not available. If you want to use these tools you will need to use a <u>native browser</u> (non-Selenium) instead.

When using a Selenium profile for recording, you will need to use the <u>Spy (Ctrl+5)</u> tool to do the learning of objects on the web page. This brings up the <u>Web Spy</u>:

0											_	-
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,			Rapise Sj	oy [http://wwv	v.libraryinformatio	nsystem.org/]					<u>↑</u> –	□ ×
Spy												
	📋 Highlighting Mode	- 🚵	- ² 0						Test XPat	h Learn		
	🔇 Go To Url	🏙 🍈										
Get Snapshot	• • • • • • •		Options						Test CSS	Learn		
(Ctrl-G)		Learn										
	Browser	Object	Tools				Locators					^
🚮 Selenium												
DOM Tree					DOM Element							
A <html xml:lang="er</td><td>n" xmlns="</p></td><td>http://www.w3.org/1999/xhtm</td><td>l"></html>		4										
♦ <head><</head>	/head>				innerHTML				Welcome to the L	ibrary Informat	ion System	
_₄ <body></body>					tagName			H2				
	method="post" action="./" id=	"ctl01">										
	iv class="aspNetHidden"><				height			26				
	criptsrc="/WebResource.axd		Rawwl "typ	e="text/iavasci	width			920				
	iv class="aspNetHidden"><	_			x			49 123				
	iv class="page">				y ⊿ Selectors			123				
	<pre><div class="header"> </div></pre>	~			CSS			html >	body:first-of-type	> form first of	tune > diventh	of three (3) >
· · · ·	<div class="main"></div>				xpath				n[@id="ctl01"]/di			or type(0) >
-	<h2>Welcome to the Lit</h2>	vrany Informati			xpath aria				n[@id="ctl01"]/di			
	<pre>div style="margin-righ"</pre>				xpath_full				/body[1]/form[1]/			
	a <uv style="margin-right</p"></uv>	L TOUPX >			xpath_indexed	d		(//h2)[1]			
			des		xpath_min				n[@id="ctl01"]/di			
	To view the libration of the libratio	ary catalog or t	t		xpath_std				n[@id="ctl01"]/di			
					xpath_text			(normalize-space(t		me to the Libr	arv Informatic
	<pre><dr><dr><dr><dr><dr><dr><dr><dr><dr><dr< td=""><td></td><td></td><td></td><td>Test Results</td><td></td><td></td><td>Record</td><td>ing Activity for</td><td></td><td></td><td></td></dr<></dr></dr></dr></dr></dr></dr></dr></dr></dr></pre>				Test Results			Record	ing Activity for			
	clable>							#	Object	Action	Data	Comm
	<h2>Other Samples</h2>								,			
		\$71122										
	<pre>ctable></pre>											
	<pre><div class="clear"></div></pre>											
	iv class="footer">	0.1415										
<s< td=""><td>cripttype="text/javascript">ne</td><td>ew Sys.WebFo</td><td>orms.Menu({</td><td>element:<td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></td></s<>	cripttype="text/javascript">ne	ew Sys.WebFo	orms.Menu({	element: <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>								
I												

When using the <u>Web Spy</u> with a Selenium profile you will notice that the web browser icon / name shows "Selenium" rather than the browser name and the option to Track an item (CTRL+T) is not present. That means you need to **select the HTML DOM object in the DOM Tree and learn it from there** (rather than clicking on the web page itself which is possible when using a native browser profile).

When you choose to Learn an object in the DOM tree it will be displayed in the <u>Recording Activity</u> <u>Dialog</u> as a new Learned Object:

#	Object	Action	Data	Comment		
Ø1 Welco Ø2 Home		Leam Leam	Welcome to Home	Learned Welcome to the Library Information Syste Learned Home		
Veni	y (Ctrl+1)	Learn (Ct	rl+2)	Spy(Ctrl+5)	Pick Object	Resume

Objects Learned using a Selenium profile will be added to the Rapise <u>Object Tree</u> in the usual way and as is typical with <u>Learning</u>, you have the option to specify an Action in the Recording Activity Grid (e.g. change Learn to Click) in which case test script code is also generated.

Tip: Due to the inherent limitations in recording using a Selenium browser profile (vs. a native browser profile) most users will record their scripts using a native browser and then use Selenium primarily for debugging using the Web Spy and playback.

Playback using Selenium

To playback a web test using a Selenium web browser profile, simply choose the appropriate profile in the test Start Page:

Parameters	
Spira Properties	
Browser:	Selenium - Firefox
Record Title:	Chrome HTML Firefox HTML Internet Explorer HTML Selenium - Chrome Selenium - Firefox Selenium - Internet Explorer Selenium - Opera Selenium - Safari

Then click the **Play** button on the main Test ribbon. The test will now start <u>execution</u>. Unlike recording there is nothing different in the way Rapise handles the <u>playback</u> of a Selenium test. The only difference will be that if the test uses non-HTML technologies such as Flex, Java, etc. those parts of the test will fail.

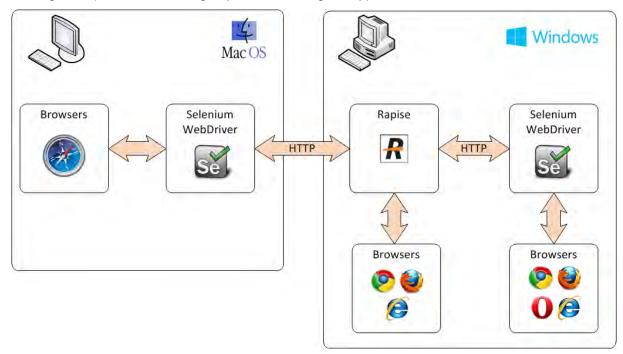
#	Туре	Start	 Name 	Status	Comment	Iteration
* 🛛		-	B			-
	Message	12:32:24.739	Starting scenario: Test	Info		
	Assert	12:32:35.183	Log In.DoClick([])	Pass	Returned Value: true	0
	Assert	12:32:36.432	Username:.DoSetText(["librarian"])	Pass	Returned Value: true	0
	Assert	12:32:37.587	Password:.DoSetText(["librarian"])	Pass	Returned Value: true	0
	Assert	12:32:38.976	ctl00\$MainContent\$LoginUser\$LoginButton.DoClick([])	Pass	Returned Value: true	0
	Assert	12:32:40.595	Book Management.DoClick([])	Pass	Returned Value: true	0
	Assert	12:32:42.003	Log Out.DoClick([])	Pass	Returned Value: true	0
•	Test	12:32:42.008	Demo Test 2 TC163	Pass	Passed:6 Failed:0	

2.6.2.1 Setting Up Selenium

This section describes the process for setting up Rapise to work with <u>Selenium</u>. Since Rapise is a Windows® application, you can use a single computer running Rapise to use the following web browsers:

- Internet Explorer
- Google Chrome
- Mozilla Firefox
- Opera
- Microsoft Edge

However because Safari only runs on Apple Mac computers, you will need to use two computers (a Mac running Safari) and a PC running Rapise to test using the Apple Safari web browser:



Configuring Selenium on a PC

Once you have installed Rapise on your local computer, you need to perform the following steps to configure each of the web browsers to use Selenium and Rapise:

Firefox

Unlike the other web browsers, Firefox does not require anything special to be done, it already includes a built-in plugin for use by Selenium WebDriver. However if you start using Rapise with Firefox and you see the following issue inside of Firefox:

Firefox WebDriver could not be verified for use in Firefox and has been disabled. More Information
 Firefox WebDriver (disabled)
 WebDriver implementation for Firefox More

Then it means that the version of Selenium WebDriver that shipped with Rapise is no longer compatible with the installed version for Firefox. The solution is straightforward, just go to the main Selenium website: http://www.seleniumhq.org/download/ and then download the **C# WebDriver Bindings**:

Language	Client Version	n Release Date	9		
Java	2.50.1	2016-01-29	<u>Download</u>	Change log	<u>Javadoc</u>
C#	2.50.1	2016-01-28	<u>Download</u>	Change log	API docs
Ruby	2.50.0	2016-01-27	<u>Download</u>	Change log	API docs
Python	2.50.0	2016-01-27	<u>Download</u>	<u>Change log</u>	API docs
Javascript (Node) 2.48.2	2015-10-15	<u>Download</u>	<u>Change log</u>	API docs

Download the **Selenium-dotnot-x.x.x.zip** file from the website. Proceed to unzip the archive and then look in the **net40** subfolder and extract the following two files and copy into the **C:\Program Files (x86) \Inflectra\Rapise\Bin** folder (or wherever you installed Rapise):

- WebDriver.dll
- WebDriver.Support.dll

Note: You will need to close Rapise before copying these files into the Bin folder.

Microsoft Edge

To use Selenium with Microsoft Edge, you will need to download the latest version of the Edge Driver from the Microsoft website:

https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/#downloads

By downloading and using this software, you agree to the license terms below

Release 14393

Version: 3.14393 | Edge version supported: 14.14393 | License terms

Insiders

Version and Edge Version Supported: Current Insiders Fast Ring Build License terms | Privacy Statement

Release 10586

Version: 2.10586 | Edge version supported: 13.10586 | License terms

Create a new folder on your local PC called:

C:\Program Files (x86)\Microsoft Web Driver

in to Quick Copy Paste access	Move to •	Copy to *	Delete Rename	New item *	Properties	Sele		
Clipboard		Orga	nize	New	Open	Se	elect	
				<u>^</u>				
🔉 🧧 Microsoft Visual Studio 12.0		^	Name	~	Date mo	dified	Туре	Size
Microsoft Visual Studio 14.0		^		^ ftWebDriver.exe		dified 16 4:16 PM	Type Application	Size 126 KB
		^						
Microsoft Visual Studio 14.0		^						
Microsoft Visual Studio 14.0 Microsoft WCF Data Services		^						

Download the MicrosoftWebDriver.exe to this local folder you just created:

Internet Explorer

To use Selenium with Internet Explorer, you will need to download the latest version of the Internet Explorer IE Driver:

http://selenium-release.storage.googleapis.com/index.html

The list of versions at time of writing was:

<u>Name</u>	Last modified	Size	ETag
<u>2.48</u>	-	-	_
<u>2.49</u>	-	-	-
<u>2.50</u>	-	-	-
icons	-	-	-
index.html	2014-01-13 22:12:39	0.01MB	704b0f841aad1b1428481b7ff3c759c0

When you click on the folder for the latest version you will see the various files that can be downloaded:

	Name	Last modified	Size	ETag
٩	Parent Directory		-	
10 01 10	IEDriverServer Win32 2.50.0.zip	2016-01-27 23:51:11	0.95MB	cf6850b0ceae8498e1952f6dead9d80b
10 01 10	IEDriverServer x64 2.50.0.zip	2016-01-27 23:51:12	1.11MB	7229088ae632893579004388bb20c5d3
10 01 10	selenium-dotnet-2.50.0.zip	2016-01-27 23:51:17	6.52MB	53c2bfb0545beceba48e7d2dd847f2a2
10 01 10	selenium-dotnet-2.50.1.zip	2016-01-28 18:06:48	6.13MB	cbc94b85ee75686ce6c3cf103670150d
	selenium-dotnet-strongnamed-2.50.0.zip	2016-01-27 23:51:15	3.83MB	55712f98477707fbeb41e84a325b40a0
10 01 10	selenium-dotnet-strongnamed-2.50.1.zip	2016-01-28 18:06:50	3.85MB	ca7bf7ebd7873ff80b48f3d74b569b3f
10 01 10	selenium-java-2.50.0.zip	2016-01-27 18:46:51	21.45MB	f1243239575d8a32e96decb4ccba6847
10 01 10	selenium-java-2.50.1.zip	2016-01-29 19:12:11	21.45MB	7648ad9f89428a443c8aff5bb5cc6885
	selenium-server-2.50.0.zip	2016-01-27 18:46:33	26.79MB	dee71d9814589c2cf3bd2e31c6710359
10 01 10	selenium-server-2.50.1.zip	2016-01-29 19:11:54	26.79MB	2f9ffd31e3b824e95395cba80dab9d02
10 01 10	selenium-server-standalone-2.50.0.jar	2016-01-27 18:46:10	29.52MB	65d4c900eef25184215326fd58c36f30
10. 01 10	selenium-server-standalone-2.50.1.jar	2016-01-29 19:11:30	29.52MB	bd291ba0e26f486ff12b45a627ecdc80

Download the IEDriverServer_XXXX_X.X.X.zip to your local PC:

- IEDriverServer_Win32_X.X.X.zip (for 32-bit Internet Explorer)
- IEDriverServer_x64_X.X.X.zip (for 64-bit Internet Explorer)

The file inside the zip archive is called **IEDriverServer.exe** and you need to copy it into the **C: \Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise).

Chrome

To use Selenium with Google Chrome, you will need to download the latest version of the Chrome Driver:

http://chromedriver.storage.googleapis.com/index.html

The list of versions at time of writing was:

	Name	Last modified	Size	ETag	
<u>2.6</u>		-			-
<u>2.6</u> 2.7		-			-
<u>2.8</u>		-			-
<u>2.9</u>		-			-

When you click on the folder for the latest version you will see the various files that can be downloaded:

	Name	Last modified	Size	ETag
2	Parent Directory		-	
	chromedriver linux32.zip	2016-01-26 06:47:39	2.64MB	d0a589f70e53774db95bf6f46972837c
	chromedriver linux64.zip	2016-01-26 15:51:03	2.57MB	06e57f4c411e1135c6277d17ea8390fd
	chromedriver mac32.zip	2016-01-26 07:59:08	3.55MB	452d8c9cba353ba366d15fbeba013943
	chromedriver win32.zip	2016-01-26 06:47:03	2.48MB	8a93dc3ff02ef9bc3161dd4b20f87215
	notes.txt	2016-01-28 23:25:03	0.00MB	d8d67de107327522f0728fb389fee377

Download the chromedriver_win32.zip to your local PC.

The file inside the zip archive is called **chromedriver.exe** and you need to copy it into the **C:\Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise).

Opera

To use Selenium with Opera, you will need to download the latest version of the Opera Driver:

https://github.com/operasoftware/operachromiumdriver/releases

This page will list the latest version of the driver at the top of the page:

v0.2.2 by v0.2.2 by v0.2.2 by v0.2.2 by v0.2.2 constant of the second s	
•• 26ae344 Fix for #10.	
Downloads	
⑦operadriver_linux32.zip 3.	1 MB
Toperadriver_linux64.zip 2.8	5 MB
Toperadriver_mac64.zip 3.4	5 MB
Toperadriver_win32.zip 2.6	4 MB
Transformation 3.1	7 MB

Download the operadriver_winXX.zip to your local PC:

- operadriver_win32.zip (for 32-bit Opera)
- operadriver_win64.zip (for 64-bit Opera)

The file inside the zip archive is called **operadriver.exe** and you need to copy it into the **C:\Program Files (x86)\Inflectra\Rapise\Bin** folder (or wherever you installed Rapise).

Installing Selenium on a Mac

The reason for using Selenium running on a Mac is to be able to execute tests against the Safari web browser. So although you can also use the Mac to test with Firefox, Opera and Chrome, we do not recommend this as it adds needless complexity.

Safari

The first thing you need to do is download the latest version of the Selenium server for Apple Mac computers:

http://selenium-release.storage.googleapis.com/index.html

The list of versions at time of writing was:

<u>Name</u>	Last modified	Size	ETag
<u>2.48</u>	-	-	-
<u>2.49</u>	-	-	-
2.50	-	-	-
icons	-	-	-
index.html	2014-01-13 22:12:39	0.01MB	704b0f841aad1b1428481b7ff3c759c0

When you click on the folder for the latest version you will see the various files that can be downloaded:

	Name	Last modified	Size	ETag
2	Parent Directory		-	
	IEDriverServer Win32 2.50.0.zip	2016-01-27 23:51:11	0.95MB	cf6850b0ceae8498e1952f6dead9d80b
	IEDriverServer x64 2.50.0.zip	2016-01-27 23:51:12	1.11MB	7229088ae632893579004388bb20c5d3
	selenium-dotnet-2.50.0.zip	2016-01-27 23:51:17	6.52MB	53c2bfb0545beceba48e7d2dd847f2a2
10 01 10	selenium-dotnet-2.50.1.zip	2016-01-28 18:06:48	6.13MB	cbc94b85ee75686ce6c3cf103670150d
	selenium-dotnet-strongnamed-2.50.0.zip	2016-01-27 23:51:15	3.83MB	55712f98477707fbeb41e84a325b40a0
	selenium-dotnet-strongnamed-2.50.1.zip	2016-01-28 18:06:50	3.85MB	ca7bf7ebd7873ff80b48f3d74b569b3f
10 01 10	selenium-java-2.50.0.zip	2016-01-27 18:46:51	21.45MB	f1243239575d8a32e96decb4ccba6847
10 01 10	selenium-java-2.50.1.zip	2016-01-29 19:12:11	21.45MB	7648ad9f89428a443c8aff5bb5cc6885
10 01 10	selenium-server-2.50.0.zip	2016-01-27 18:46:33	26.79MB	dee71d9814589c2cf3bd2e31c6710359
10 01 10	selenium-server-2.50.1.zip	2016-01-29 19:11:54	26.79MB	2f9ffd31e3b824e95395cba80dab9d02
10 01 10	selenium-server-standalone-2.50.0.jar	2016-01-27 18:46:10	29.52MB	65d4c900eef25184215326fd58c36f30
10 01 10	selenium-server-standalone-2.50.1.jar	2016-01-29 19:11:30	29.52MB	bd291ba0e26f486ff12b45a627ecdc80

Download the selenium-server-standalone-X.XX.X.jar to the Mac.

Run this Java application by double clicking the downloaded .JAR file in Finder. This will startup the Selenium server.

Note: You will need to have the Java (ideally the latest version) installed on the Mac first.

Once you have this running, you will need to then download the actual Safari WebDriver plugin. This can be found at the following location:

https://github.com/SeleniumHQ/selenium/wiki/SafariDriver

	Name	Last modified	Size	ETag
	Parent Directory			
	IEDriverServer_Win32_2.45.0.zip	2015-02-27 18:18:08	0.81MB	dde210e04e5c1b0d6019fd8a1199df18
	IEDriverServer_x64_2.45.0.zip	2015-02-27 18:18:09	0.90MB	fc9e083200dfdc35d837a586927a1f86
	SafariDriver.safariextz	2015-02-27 00:07:39	0.21MB	8f6c341f8fb6a8b89801ae532c68e1b1
	selenium-dotnet-2.45.0.zip	2015-02-27 18:18:07	10.13MB	9b172ad6a96cf497867be0efbe9acac8
	selenium-dotnet-strongnamed-2.45.0.zip	2015-02-27 18:17:55	7.87MB	ff51ed60c1b04255649f6f28e13e4207
	selenium-java-2.45.0.zip	2015-03-05 23:12:19	23.90MB	5adf84e7eb9f7b32e1b6a1d59cb93769
8	selenium-server-2.45.0.zip	2015-02-27 00:07:36	32.17MB	5034f099c70533fbac38f0c246101b9b
8	selenium-server-standalone-2,45,0,jar	2015-02-27 00:07:33	33.64MB	a62db4c36e230a936455aaacda9340a8

Download the SafariDriver.safariextz file to the local computer and the double-click to install in Safari:

Safari File Edit View History Bookm	arks Window Help		
0 O Saf	ariDriver – selenium – Information about the	Safari Driver - Browser autom	ation framework - Google Project Hosting
	ads/SafariDriver.safariextz		
selenium-re/index.html Apple iCloud Face	book Twitter Wikipedia Yahoo News 🔻	Popular =	
Selenium Browser automation framework	to GitHub		
arch Current pages ÷ for arcDriver mation about the Safari Driver Driver	Search	Updated Feb	Are you sure you want to install the extension "WebDriver"? Install extensions only from sources you trust. Extensions can modify the appearance and behavior of Safari, and access your private information at websites.
The SafariDriver The SafariDriver		_	Don't Install Install

Once that has been installed, you are now ready to test web applications running on Safari. The final step is to tell Rapise where it can find that instance of Selenium. To do that, open up Rapise (on your PC) and click on **Options > Tools** and then click on the **'Selenium Settings...'** entry:

elect Profile: Safari	Save Test Duplicate Create Rename Dele
≣ 2↓ ©	
Connection	
Uri	http://192.168.0.1:4444/wd/hub
Details	
 Primary Capabilities 	(1)
browserName	Safari
Common Capabilities	(1)
javascript Enabled	
Primana Canadalitian	
Primary Capabilities	(i.e., a map or hash) sent to the Selenium server to tell the server what kind of automation
ession we are interested in starting up. There are	e also various capabilities which can modify the behavior of the server during automation.
and the second second second second	

Now you need to change the Uri field to point to your Mac. The format of the URI will be:

• http://<IP or DNS name of MAC computer>:4444/wd/hub

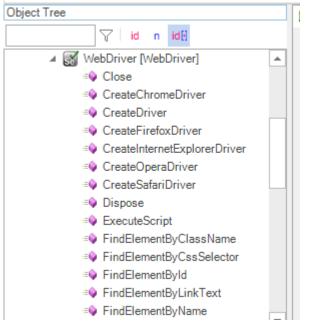
(for example it could be http://test-mac01.local:4444/wd/hub or http://192.168.0.52:4444/wd/hub)

2.6.2.2 Using Native Selenium Code

When using the standard Rapise <u>Record, Learn</u>, and <u>Web Spy</u> tools you can create your web tests using the Rapise built-in <u>Object Tree</u>. This lets you design your tests using a higher level of abstraction than working directly with <u>Selenium WebDriver</u>.

For example you can learn an object Ses("EditButton") that points to a dynamic <u>XPATH</u> or <u>CSS</u> query that the automation engineer knows will be accurate even if the data on the page changes (for example). The QA analyst can then simply drag and drop these Rapise objects from the <u>Object Tree</u> into the test script (e.g. Ses("EditButton").DoClick()) to perform the desired action.

Sometimes however you will want to be able to run standard Selenium raw WebDriver code inside Rapise using the standard <u>Selenium WebDriver</u> API functions (<u>http://www.seleniumhq.org/</u><u>docs/03 webdriver.jsp</u>). Rapise allows you to do this using the special **WebDriver** global object:



The **WebDriver** object implements the various standard Selenium WebDriver API calls for automating the web browser. There is a sample available for Rapise called "UsingSelenium" that illustrates using the WebDriver code directly, but for completeness, here is a sample that uses the <u>www.libraryinformationsystem.org</u> same web site and performs some simple actions:

```
//First create the Firefox driver
WebDriver.CreateFirefoxDriver();
//Open the URL for the www.libraryinformationsystem.org website:
WebDriver.SetUrl('http://www.libraryinformationsystem.org');
//Find the body element and verify the text in it
var el = WebDriver.FindElementByXPath("//body");
Tester.Assert("Text found in BODY", el.GetText().indexOf("Library Information
System") != -1);
//Click on the login link
var logInLink = WebDriver.FindElementById('HeadLoginView_HeadLoginStatus');
logInLink.Click();
//Make sure the input textbox is as expected
var userName = WebDriver.FindElementByCssSelector("html > body > form > div:nth-
of-type(3) > div:nth-of-type(2) > div:nth-of-type(2) > fieldset > p:first-of-type
> input");
Tester.AssertEqual("class is 'textbox'", "textbox",
userName.GetAttribute("class"));
//Go to a different URL (http://libraryinformationsystem.org/HtmlTest.htm)
WebDriver.SetUrl('http://www.libraryinformationsystem.org/HtmlTest.htm');
//Click on the Alert box
var alertBtn = WebDriver.FindElementById("btnAlert");
```

```
alertBtn.Click();
```

```
//Switch to this alert box and close
var alertElement = WebDriver.SwitchToAlert();
alertElement.Accept();
```

```
//Shut down Selenium
WebDriver.Quit()
```

When you click '**Play**' to playback your Selenium script, make sure you have selected one of the Selenium web browser profiles. If you have selected a native browser profile (e.g. "Firefox HTML" instead of "Selenium – Firefox") you will get the error message "WebDriver" is not defined.

Code Completion for the Selenium WebElement Objects

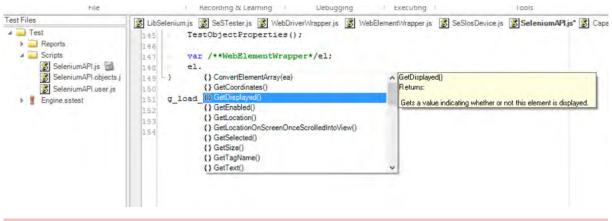
When you are using functions such as FindElementsById() in your code, the returned object will be a Selenium Web Element.

If you want to have intellisense and code-completion for the returned WebElement objects, you need to do two things:

- Click on CTRL+E to include the Rapise engine in your project.
- Prefix the variable with /**WebElementWrapper*/, so instead of just:

```
var el = WebDriver.FindElementById(...) you use:
```

var /**WebElementWrapper*/el = WebDriver.FindElementById(...) instead



For example:

Interoperability with Rapise Objects

In addition to being able to use raw Selenium code on its own, you can also use a mixture of Rapise object-based code and raw Selenium WebDriver code.

For example, you are using the standard Rapise approach (using learned Ses('object') objects) for testing but at some point want to switch to Selenium API to call a couple of WebElement functions on a learned object, you can use the special '**element**' property:

```
var webElement = SeS('MyObject').element;
```

If you want to the reverse and be able to create a Rapise SeS('object') 'on the fly' from a physical object on the web page, you can do the MakeObjectForXPath(xpath) function that returns a Rapise SeSObject, in the same way that SeS('id') does normally:

var sesObj = MakeObjectForXPath("//body//div[@id='logArea']");

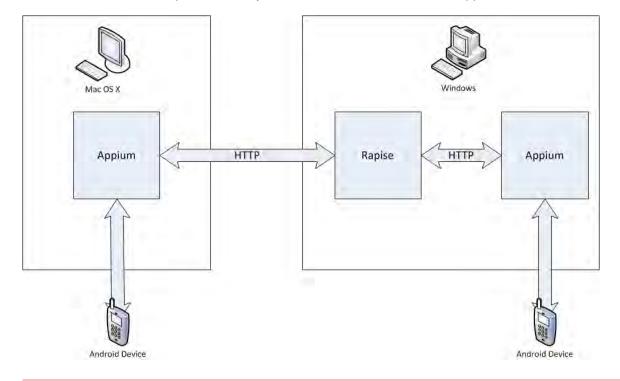
2.6.3 Mobile Testing

Purpose

Rapise lets you record and play automated tests against native applications on a variety of mobile devices using either <u>Apple iOS</u> or <u>Android</u>. Rapise gives you the flexibility to test your applications on either real or simulated devices.

This section explains **how to setup your environment for mobile testing**, once that is done, you can the go to <u>the section that explains the process</u> for using Rapise to actually perform <u>mobile testing</u>.

Rapise uses a third-party open-source tool called **Appium** (<u>http://appium.io</u>) that is used to actually host the mobile devices and Rapise essentially communicates to the device via. Appium:



Testing Architectures

Rapise runs on Windows computers (PC) and Android devices (both real and simulated) can be tested on either an Apple Macintosh (Mac) computer or a PC. Conversely, iOS devices (both real and simulated) can only be tested on an Apple Macintosh (Mac) computer. So this means that there are three separate possible testing environments that you may need to setup:

- Using a Mac to Host iOS Devices. It will be necessary to install Appium and Apple Xcode onto the Mac and connect to Appium over the network from Rapise running on your PC.
- Using a Mac to Host Android Devices. It will be necessary to install Appium and Android Studio onto the Mac and connect to Appium over the network from Rapise running on your PC.
- Using a PC to Host Android Devices. You can either install Appium and Android Studio onto a separate PC or you can simply use the same PC that is running Rapise. The only difference will be whether the URL used to connect to Appium is a localhost URL or one pointing to the other PC.

The steps for setting each of these will be described separately below:

1) Using a Mac to Host iOS Devices

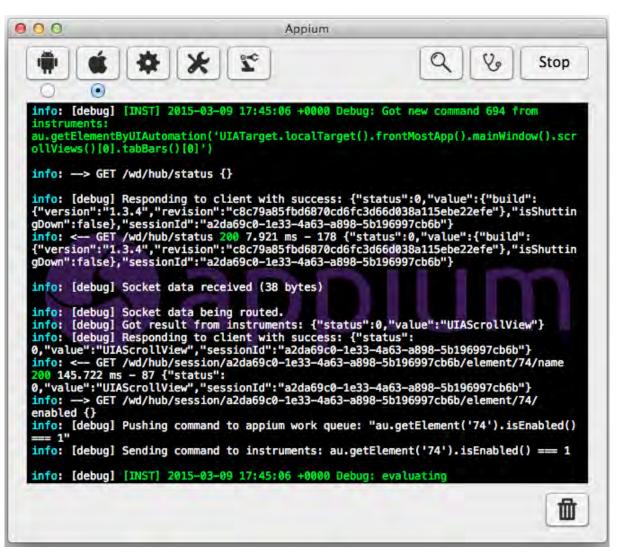
The first thing you need to do is install Xcode from the Apple Mac app store. Make sure you include the **iOS SDK**, and also the **iOS Simulator** if you intend to test simulated iOS devices.

×.	Xcode File Edit Vi	iew Find	Navigate	Editor	Product	Debug	Source Control Window Help
-	About Xcode						
	Preferences Behaviors	₩,					AUTIOS ~/Documents/AUTIOS
	Open Developer Tool		Instrum	nente		200	SafariLauncher ~/Documents
	Services	Þ	iOS Sir				run-webkit-proxy.sh
	Hide Xcode Hide Others Show All	₩₩ 十業プ	G Access	rge			-/Documents UICatalog ingUIKitControlsObj-CandSwift/Objective-C
	Quit Xcode	жQ	More Dev	eloper To	ols	P	hybrid.md ~/node_modules/appium/docs/en
	Versio	on 6.1 (6A1	052d)				InternationalMountains ~/Documents
	Get started with a playground Explore new ideas quickly and easily.						mobile-web.md ~/node_modules/appium/docs/en troubleshooting.md
	Create a new Start building a			applicatior	ı.	ZIP	~/node_modules/appium/docs/en InternationalMountains.app.zip ~/Documents/ios-driver
	Check out an Start working o			repository			
						Ope	n another project
							A THE CONTRACT OF A DESCRIPTION OF A DES

(Please refer to the Apple tutorial <u>https://developer.apple.com/library/ios/referencelibrary/</u> <u>GettingStarted/RoadMapiOS/</u> if you are writing your first iOS application and need an introduction into how to develop for iOS).

Since configuring Xcode to build and deploy an application to a physical or simulated iOS device is quite involved, we have created a <u>separate topic that explains the process</u>.

Once you have the iOS environment configured, you need to do is go to the **Appium** website (<u>http://appium.io</u>) and install the latest version of Appium. Once it is installed, you need to select the option for

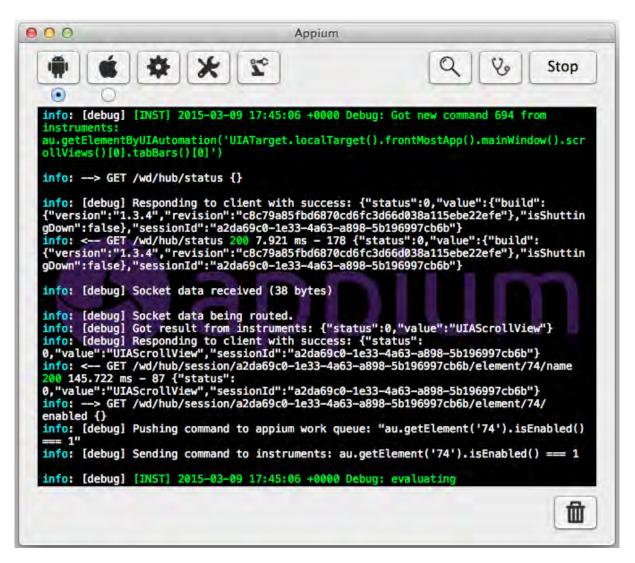


iOS and click the Play button to start the Appium server:

You are now ready to start mobile testing of your iOS device.

2) Using a Mac to Host Android Devices

The first thing you need to do is go to the **Appium** website (<u>http://appium.io</u>) and install the latest version of Appium. Once it is installed, you need to select the option for **Android** and click the Play button to start the Appium server:



Once that is installed, you will then need to download and install the latest version of Java SE Development Kit (JDK) from the Oracle website (<u>http://www.oracle.com/technetwork/java/javase/downloads/index.html</u>). Once that has been installed, make sure that the JAVA_HOME environment variable has been set.

Once that is installed, you will then need to install the Android SDK (you may already have it installed if you are doing Android development). You can download it from: <u>https://developer.android.com/sdk</u>.

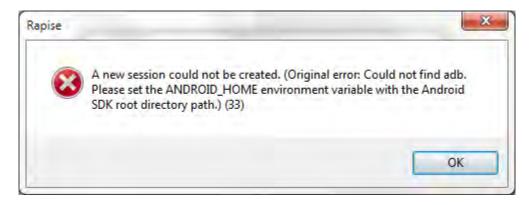
Once it has installed, you will use the **Android SDK Manager** to download and install the necessary packages:

Name		API	Rev.	Status	
Tools					
Android SDK Too	ols		24.0.2	📑 Update available: rev. 24.1	
Android SDK Pla	tform-tools		21	📑 Update available: rev. 22	
Android SDK Bui	ild-tools		22.0.1	Not installed	
Android SDK Bui	ld-tools		21.1.2	😿 Installed	
Android SDK Bui	ild-tools		21.1.1	Not installed	
📌 Android SDK Bui	Android SDK Build-tools		21.1	Not installed	
Android SDK Bui	Android SDK Build-tools		21.0.2	21.0.2 Not installed	
📌 Android SDK Bui	📌 Android SDK Build-tools		21.0.1	Not installed	
Android SDK Bui	ild-tools		21	21 Not installed	
📌 Android SDK Bui	ild-tools		20 Not installed		
Android SDK Bui	ild-tools		19.1 Not installed		
📌 Android SDK Bui	ild-tools		19.0.3 Not installed		
Android SDK Bui	ild-tools		19.0.2	Not installed	
Android SDK Bui			19.0.1	Not installed	
w: 🗹 Updates/New 🗹 I		ites		Install packages	
Obsolete	Deselect All			Delete packages	

If you are going to be testing a physical Android device, you will need to do the following:

- 1. Make sure you have enabled Developer mode in the Android device itself:
 - a. Open Settings> About on your Android phone or tablet.
 - b. If you have a Samsung Galaxy S4, Note 8.0, Tab 3 or any other Galaxy device with Android 4.2, open Settings> More tab> About and tap it.
 - c. If you have Galaxy Note 3 or any Galaxy device with Android 4.3, go to Galaxy Note 3 from Settings> General> About and tap the Build version 7 times.
 - d. Now scroll to Build number and tap it 7 times.
 - e. After tapping the Build Number 7 times, you will see a message "You are now a developer!" If you have a Galaxy S4 or any other Samsung Galaxy device with Android 4.2, the message reads as follows- "Developer mode has been enabled".

Now when you try and <u>connect to the device</u> using the <u>Rapise mobile spy</u>, you may get the following message:



This means you need to use a MacOS X Shell window to add a **environment variable** called **ANDROID_HOME** and set it to the path of the installed Android SDK (typically something like /Users/my.user/Downloads/android-sdk-macosx).

If you want to test using the Android simulator, make sure you have installed it using the SDK manager. Then you can launch (from the main menu of the Android SDK Manager) the **Android Virtual Device (AVD) Manager**:

ist of existing	Android Virtual Devices loo	cated at /Users/	elise.broo	ks/.android/avd		Rev. Status
VD Name	Target Name	Platform	API Level	CPU/ABI	Create	incv. Status
emulato	r64-Android 4.3.1	4.3.1	18	ARM (armeabi-v7a)	Start	24.0.2 Update available: rev. 24.1 21 Update available: rev. 22 22.0.1 Not installed
					Edit	21.1.2 👼 Installed
					Repair	21.1.1 Not installed 21.1 Not installed
						21.1 Not installed 21.0.2 Not installed
					Delete	21.0.1 Not installed
					Details	21 Not installed
						20 Not installed
						19.1 Not installed
						19.0.3 🗌 Not installed
						19.0.2 Not installed
						19.0.1 Not installed
						Install 27 packages
					Refresh	Delete 14 packages

In this case you can just create the Android Virtual Device, Start it and then connect to it using Rapise.

You are now ready to start mobile testing of your Android device.

3) Using a PC to Host Android Devices

The first thing you need to do is go to the **Appium** website (<u>http://appium.io</u>) and install the latest version of Appium. Once it is installed, you can start it up and click the Play button to start the Appium server:



Once that is installed, you will then need to download and install the latest version of Java SE Development Kit (JDK) from the Oracle website (<u>http://www.oracle.com/technetwork/java/javase/downloads/index.html</u>). Once that has been installed, make sure that the JAVA_HOME environment variable has been set.

Once that is installed, you will then need to install the Android SDK (you may already have it installed if you are doing Android development). You can download it from: <u>https://developer.android.com/sdk</u>.

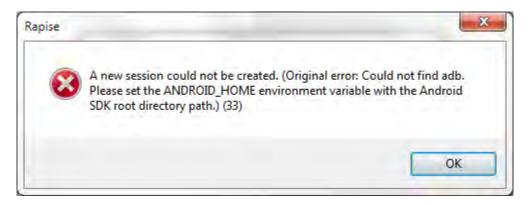
Once it has installed, you will use the **Android SDK Manager** to download and install the necessary packages:

Packages Tools					
DK Path: C:\Program Files (x86)\Android\android-sdk					
Packages					
iii Name	API	Rev.	Status		*
a 🔲 🦲 Tools					
Android SDK Tools		24.1.2	Tinstalled		
Android SDK Platform-tools		22	🔯 Installed		
🔲 🥜 Android SDK Build-tools		22.0.1	🔯 Installed		
🔲 🧚 Android SDK Build-tools		21.1.2	Not installed		
🔲 🥕 Android SDK Build-tools		20	Not installed		
🔲 🧚 Android SDK Build-tools		19.1	Not installed		
Android 5.1.1 (API 22)					
Documentation for Android SDK	22	1	🔯 Installed		
🔲 🃫 SDK Platform	22	2	🔯 Installed		
🔄 🚣 Samples for SDK	22	5	🔯 Installed		
🔄 📑 Android TV ARM EABI v7a System Image	22	1	🔯 Installed		
🔲 🌆 Android TV Intel x86 Atom System Image	22	1	🔯 Installed		
🕅 🌆 ARM EABI v7a System Image	22	1	🔯 Installed		Ŧ
Show: Vpdates/New VInstalled Select New or Updates				Install packages	
Obsolete Deselect All			L	Delete packages	
	_				

If you are going to be testing a physical Android device, you will need to do the following:

- 2. Locate the **Google Android USB drivers** that came with the Android SDK (C:\Program Files (x86)\Android\android-sdk\extras\google\usb_driver) and when you connect your Android device to the PC, choose to install these drivers rather than the standard ones.
- 3. Make sure you have enabled Developer mode in the Android device itself:
 - a. Open Settings> About on your Android phone or tablet.
 - b. If you have a Samsung Galaxy S4, Note 8.0, Tab 3 or any other Galaxy device with Android 4.2, open Settings> More tab> About and tap it.
 - c. If you have Galaxy Note 3 or any Galaxy device with Android 4.3, go to Galaxy Note 3 from Settings> General> About and tap the Build version 7 times.
 - d. Now scroll to Build number and tap it 7 times.
 - e. After tapping the Build Number 7 times, you will see a message "You are now a developer!" If you have a Galaxy S4 or any other Samsung Galaxy device with Android 4.2, the message reads as follows- "Developer mode has been enabled".

Now when you try and <u>connect to the device</u> using the <u>Rapise mobile spy</u>, you may get the following message:



This means you need to use the Windows control panel to add a **System environment variable** called **ANDROID_HOME** and set it to the path of the installed Android SDK (typically C:\Program Files (x86)\Android\android-sdk).

If you want to test using the Android simulator, make sure you have installed it using the SDK manager. Then you can launch (from the Windows Start Menu) the **Android Virtual Device (AVD) Manager**:

ls			
droid Virtual Dev	vices Device Definitions		
st of existing An	droid Virtual Devices located a	t C:\Users\adam.sandman\.android\avd	
AVD Name	Target Name	Platfor API Le CPU/ABI	Create
14	No AVD available		Start
			Edit
			Repair
			Delete
			Details
			Refresh
			Keffesh

In this case you can just create the Android Virtual Device, Start it and then connect to it using Rapise. You are now ready to start mobile testing of your Android device.

See Also

- Mobile Testing, for an overview of mobile testing with sub-sections on testing using iOS and Android.
- Mobile Testing Tutorial for a simple introduction to mobile device testing.
- <u>Mobile Settings Dialog</u> for information on setting up the different mobile profiles for the mobile devices you will be testing
- <u>Mobile Object Spy</u> for information on how Rapise connects to the device and lets you view the objects in the application being tested
- Mobile Testing: iOS Setup the steps for setting up Xcode and the iOS SDK for testing iOS devices

2.6.3.1 Mobile Testing: iOS Setup

Purpose

This section describes how to setup Apple Xcode for developing and deploying iOS applications to a real or simulated device so that they can be tested by Rapise.

Make sure you have already installed XCode and the iOS SDK onto your Apple Mac as described in the <u>Mobile Testing parent topic</u>.

This topic describes the process for building and deploying the sample AUTIOS application that comes with Rapise, however it can be used equally well with your in-house application.

1) Get the AUTiOS Source Code

When you install Rapise, the sample AUT for iOS (AUTiOS) is placed in the following folder on your PC:

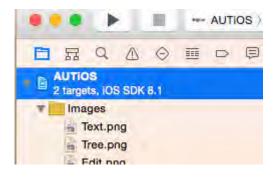
C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTiOS

You will need to copy this folder across onto your Mac so that you can open it in Xcode.

Once you have done that, launch Xcode on the Mac:



Open the AUTIOS project and select the root node:



Before you can actually build and deploy this project, you will need to register for an **Apple ID** and setup an Apple Developer account. You should check with your company to see if they have already joined the **Apple iOS Developer Program**, if not, you will need to join yourself and become a member. You can learn more about this at the Apple developer website: <u>https://developer.apple.com</u>.

2) Join Your iOS Development Team

Assuming that either you or your company already has signed up for the iOS Developer Program, you will need to ask the administrator of your account (it might be you) to send an invitation to you if you are not already a member. The link for accepting such an invitation is typically:

https://developer.apple.com/programs/start/jointeam/index.php?success=%2Fios%2Finvitation% 2Faccept.action

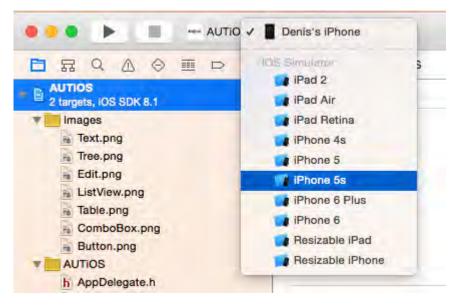
Click on this link and accept the invitation.

Meanwhile, back in XCode Use the 'Add an Account...' to login with your Apple ID:

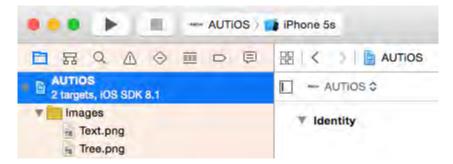
- AUTIOS \$	General Capabilitie	s Info	Build Settings	Build Phases	Build Rules
▼ Identity					
	Bundle Identifier	com.inflectra.A	UTIOS		
	Version	1.0		1	
	Build	1			
	Tear	Inflectra Corp	oration (markovtsevé	gmail.com)	
V Deployment Info		None		-	
Deployment into	1	Add an Accou	int		
	Deployment Target	B.(~		
	Devices	Universal	•		
	Main Interface	Main			
	Device Orientation	Portrait			
		Upside Dow	n		
		Landscape I	_eft		
		🕗 Landscape I	Right		
	Status Bar Style	Default	3		
		Hide status h	par		

3) Building and Deploying on a Simulated Device

Now that you have signed into Xcode using your developer account, you can select a simulated device and run the project on it:



Once you have selected the simulated iOS device you want to use, click the **Product > Build** option to build the app for the targeted device. You can use the **Run** option to make sure that the app actually launches on this device before testing it with Rapise.



Assuming that this is successful, you will see the AUTiOS running in the iOS Simulator:

Carrier	•	1:57 PM	-
Edit		AUT Items	+
ab	Buttor	í:	
abl	Text		
00	Table		
	Comb	o Box	
Cont	tents		

If you are only going to use Simulated devices (not recommended) then you can skip the next section and just continue with setting up **Appium**, as described in the main <u>Mobile Testing topic</u>.

4) Building and Deploying on a Physical Device

Login with your Apple ID to http://developer.apple.com

Choose Certificates, Identifiers & Profiles:

	People	Programs & Add-ons	Your Account
ization: Infle	ectra Corporation		
Develop	er Program Resourc	es	
Techni	cal Resources and To	pols	
	Dev Centers Quickly access a	range of technical	Certificates, Identifiers & Profiles
	iOS Mac Sat	fari	Manage your certificates, App IDs, devices, and provisioning profiles.
App St	ore Distribution		
6	App Store Res	source Center	iTunes Connect
1	Learn about how App Store Submit		Submit and manage your apps on the App Store.

Select Devices:

	iOS Apps
6	Certificates
	Identifiers
	Devices
Ē	Provisioning Profiles
Lea	urn More
D	App Distribution Guide

Add your device's UDID to the list of registered iOS devices in the developer account:

Certificates, Identifiers &	& Profiles		Denis Markovtsev 🔻
iOS Apps 🔹		iOS Devices	+ 2
🎸 Certificates	You can register 98 additio	nal devices.	
a All	Name	UDID	
Pending	Denis's iPhone	cd68f859a6ff27fc68dc1642cb9	88db065137f89
Development	Inflectra	b6789598c42703429379d147a	6f81ecea95edb66
Production			

You can find out the UDID by connecting it to the Mac and viewing the device inside Xcode.

000		Device	5
DEVICES My Mac	Device Information		
<u>≥</u> 10.9.5 (13F34)	Name	Goshen iphon	e
Goshen iphone 7.1.1 (11D201)	Model	iPhone 4 (CDM	(AA)
	Capacity	13.52 GB (12.	89 GB available)
SIMULATORS	iOS	7.1.1 (11D20)	1)
7.1 (11D167)	Identifier	b54cfb1c05a4	1438eb95e322a7ed8df3e45978654
iPad 2 8.1 (128411)	View Device Logs	Take Screenshot	
iPad Air 7.1 (11D167)			
iPad Air 8.1 (128411)	Installed Apps		
iPad Retina 7.1 (110167)	Name	Version	Identifier
iPad Retina	- AUTIOS	1	com.inflectra.AUTiOS

Then, back in Xcode choose your physical device, and use the **Product > Build and Run** option to test that the app launches on the device:

Xcode		Product Debug Run Test	Source 第R 第U	Control Window Help
•	O O O AUTIOS) 📕 Goshen iphone	Profile Analyze	策I 企業B	AUTiOS.xc
m		Archive		AUTIOS.xcodeproj
3	By File By Type	Build For	•	General Capabilities
	AUTIOS 3 issues	Perform Action		Combined Levels +
	V AUTIOS.xcodeproj	Build	ЖB	
	A Retina 4-inch Support Applications using Launch Screen Files and targetting earlier need to also include a Launch Image in an Ass ▼ m AUTMainController.m ► △ Semantic Issue Conflicting return type in implementation of 'number PickerView:': 'NSInteger' (aka 'int') vs 'long'	Clean Stop	企第K 策.	ting
		Scheme	E.	ditional SDKs
		Destination	•	hitectures
		Create Bot		e SDK
	Semantic Issue Conflicting return type in implementation of 'pickerVie			Id Active Architecture Only
the second s	RowsinComponent': 'NSinteger' (aka 'int') ve 'iong'	wanumberor		Debug

Example

You can find the iOS sample tests and sample Application (called AUTiOS) in your Rapise installation at the following locations:

Sample iOS Tests:

- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AppiOS (testing a native App)
- C:\Users\Public\Documents\Rapise\Samples\UsingMobile\WebiOS (testing a web app)

Sample Application (AUTiOS)

• C:\Users\Public\Documents\Rapise\Samples\UsingMobile\AUT\AUTiOS

See Also

• Mobile Testing, for an overview of mobile testing with sub-sections on testing using iOS and Android.

- Mobile Testing Tutorial for a simple introduction to mobile device testing.
- <u>Mobile Settings Dialog</u> for information on setting up the different mobile profiles for the mobile devices you will be testing
- <u>Mobile Object Spy</u> for information on how Rapise connects to the device and lets you view the objects in the application being tested

2.6.4 Microsoft Dynamics

Overview

Microsoft Dynamics is a line of **enterprise resource planning (ERP)** and **customer relationship management (CRM)** software applications from Microsoft. The Microsoft Dynamics focus industries are retail, service, manufacturing, financial services, and the public sector. Microsoft Dynamics offers support for small, medium, and large businesses.

Microsoft **Dynamics ERP** comprises a group of enterprise-resource-planning products primarily geared toward midsize organizations with simple corporate structures and low-to-moderately complex production models. Microsoft Dynamics ERP includes three primary products:

- Microsoft Dynamics AX (formerly Axapta) multi-language, multi-currency enterprise resource planning (ERP) business software with global business management features for financial, human resources, and operations management as well as additional industry capabilities for retailers, professional service industries, financial service businesses, manufacturers, and public-sector organizations
- Microsoft Dynamics GP (formerly Great Plains Software) ERP software for small and midsize businesses: helps manage financials, supply chain, and employees
- Microsoft Dynamics NAV (formerly Navision) business management solution that helps small and mid-sized businesses manage their financials, supply chain, and people. It features multiple languages and multiple currencies.

Microsoft Dynamics CRM is a customer relationship management application from Microsoft, that provides sales, service, and marketing capabilities. Microsoft sells Microsoft Dynamics CRM separately from the ERP products. CRM is available either as on-premises software or as a software-as-a-service offering called "Microsoft Dynamics CRM Online".

Starting in late 2016, Microsoft has created a new cloud-based SaaS-only combined ERP/CRM solution called **Microsoft Dynamics 365**. This provides a new web-based version of Dynamics AX (renamed Dynamics 365 for Operations), combined with a new web based version of Dynamics NAV (called Dynamics 365 for Financials) and an updated Microsoft Dynamics CRM Online. This new integrated, ERP/CRM is provided solely through Microsoft Azure and is completely web-based.

Rapise Support

Rapise provides out of the box support for the following different versions of Microsoft Dynamics:

- <u>Dynamics AX</u> Rapise provides specialized support for the Dynamics AX core user interface plus add-ons such as the Management Reporter.
- <u>Dynamics CRM</u> Rapise uses its native web library support for Dynamics CRM, with Dynamics-CRM specific extensions included for unique CRM components (e.g. data grids)
- <u>Dynames 365</u> Rapise includes built-in support for Dynamics 365 including both its ERP and CRM components.

2.6.4.1 Dynamics AX

Overview

Microsoft Dynamics AX (usually referred to as just "AX") - is an ERP system for mid-size to large enterprises. It is the most robust, scalable, and functionally rich enterprise resource planning system in the Microsoft Dynamics family of products. The system was originally known as Axapta, owned by the Danish software company Damgaard.

Dynamics AX consists of a client-server architecture with a thick-client ERP interface for entering/ adding data in the system:

	Vendor Procur	ement 1	nvoice General				-			
Vendor New	Edit in grid Delete Edit 10 On hold T Maintain	Invoices	Add vendor to another legal entity* Copy	Contacts	update ons	s Transactions Transactions	Balance Balance	Vendor Vendor requests search Related information	Refresh Export to Microsoft Excel List	Generate from Att template * Att
Favorit	es *	All yendo	ors •				Typ	pe to filter	Vendor account * ->	7232
My favorites Accounts payable Area page		Vendor a	ccount Name		Vendor hold	Phone	Extension		- Income and a second sec	
		BRMF-00	0003 Advent	ure Works Cycles Brasil L	No					
		DRME-00001 Fabricat		am Brasil Ltda No ic Design Brasil Ltda No	No					
					No					
4 Com		s BRMF-000002 Pr BRMF-000008 R		ura de São Paulo	No					
4 Ve	ndors Ill vendors			are Brasil Ltda	No					
	endors on hold				No No					
Ho Ho	me	BRMF-00	0006 SEFAZ	- Sao Paulo	No					
Aci	counts payable	e BRMF-000004 Wide World Importers		/orld Importers	No 111	111-555-0037				
Ac	counts receivable									
Ge	neral ledger									
Fise	cal books									
Bu	daetina									

Rapise includes specialized libraries for testing Dynamics AX applications that are built-upon the standard <u>Microsoft Windows</u> **UIAutomation** library with special extensions for handling unique AX controls such as treeviews, the navigation explorer and the various grids used to edit data. In addition Rapise can test the following extensions to Dynamics AX:

- Dynamics AX Management Reporter Rapise can test this extension using its Generic Windows library
- Dynamics AX Web Portals Rapise can test the various web portals using its <u>web browser</u> <u>libraries</u>.

Start Recording a New Test

First you need to create a new Basic test and start recording session. Choose Dynamics AX from the list of applications:

R Select an Application to Record		- • ×
Select From Running Applications Run Applic	ation	
Drag the Finder Tool over a window select it from the list below Applications	or	
Available Applications	PID	Path
Microsoft Dynamics AX? - ??Contoso? [??	3760	C:\Program Files (x8)
<		•
+ Libraries		
Cancel	2 Refre	sh Select

Then press Select button to start recording. Rapise will automatically plug the UIAutomation and DynamicsAX libraries.

- Microsoft UI Automation is the new accessibility framework for Microsoft Windows, available on all operating systems that support Windows Presentation Foundation (WPF). UI Automation provides programmatic access to most user interface (UI) elements on the desktop, enabling assistive technology products such as screen readers to provide information about the UI to end users and to manipulate the UI by means other than standard input. UI Automation also allows automated test scripts to interact with the UI.
- **DynamicsAX** library supports a set of controls specific to the Microsoft Dynamics AX 2012 application.

When a recording is completed you can see the attached libraries in the code of test.js file:

g_load_libraries=["UIAutomation", "DynamicsAX"];

Automatic Adjustment of Window Title Object Property

The main window title of Dynamics AX is dynamic by nature.

Microsoft Dynamics AX - Contoso [WIN-DR7HHF5ESQT: Session ID - 3] - [1 - dat]

It contains not only the application name but also domain and AOS information and session Id. So it can be different at the time of test recording and test playback. To cope with this problem Rapise automatically replaces the actual window title with a regular expression in object properties. Here it is:

```
regex:.Microsoft Dynamics AX.*
```

So the recorded object properties are automatically adjusted as

Pro	perties					
] ≵ ↓ 💼					
	Ignore Object Nam	False	*			
	Name	AddressBarContainer				
	Role	param:object_class				
	Text					
	Туре	DynamicsAXAddressBar	_			
⊿	Window					
	Window Class	AxMainFrame	Ξ			
	Window Text	regex:.Microsoft Dynamic				
	Window Title	regex:.Microsoft Dynamic	Ŧ			
Class						

Also Rapise automatically sets the Record Title in the test settings to the same regular expression so you do not need to choose the Dynamics AX main window during subsequent recording sessions.

_
• E
-

Titles of Child Windows

Child windows of Dynamics AX also may have dynamic titles. Rapise does not know all the available patterns, so for child windows you will need to write regular expressions yourself. But the good news is you need to do this for one object only in every such window. For further learned objects, Rapise will change the window title property automatically. In other words when Rapise learns a new object and it's window title is matched by a regular expression of a previously learned object then the title property is automatically replaced by this regular expression.

Object Location

The object location property has the form of

id1/id2/id3...

where ids refer to parent objects along the path to the top window object. Sometimes such ids also can be dynamic, e.g.:

Dimensions display (1 - k13) - Sales order: S30014323, CE15P, Sales

```
order: S30014323/InventDimParmFixed/Pane/View/Style
```

In this case change the dynamic parts with corresponding regular expressions. In the above example, the updated location looks like:

```
regex:Dimensions display.*/InventDimParmFixed/Pane/View/Style
```

Here is an example of the updated location in the property grid:

Pro	perties	
	₽↓	
4	Locators	
\triangleright	Locators	(1)
4	Object	
	ID	Style1
	Location	regex:Dimensions display.*/InventDimParmFixed/Pane/View/Style
	Version	0
4	Recognition	n
	Class	param:object_flavor
	Flavor	CheckBox

How to Launch Dynamics AX Client

If in your test you want to check that the Dynamics AX application is installed and running use the code:

var fso = new ActiveXObject("Scripting.FileSystemObject");

```
var pfFolder = Global.GetSpecialFolderPath("ProgramFilesX86");
var dynamicsPath = pfFolder + "\\Microsoft Dynamics AX\\60\\Client\\Bin\
\Delta x32.exe"
if(!fso.FileExists(dynamicsPath))
ł
    Tester.Message("Dynamics AX Client is not installed on this computer");
    return;
}
var windows = q util.FindWindows("reqex:.Microsoft Dynamics AX.*",
"AxMainFrame");
if (windows.length == 0)
    Tester.Message("Dynamics AX Client is not started. Please start it
manually and re-run the test.");
    return;
}
To start the application use
Global.DoLaunch(dynamicsPath);
```

Recording Actions and Learning Objects

During recording while you interact with Dynamics AX controls Rapise captures actions and displays them in the recording dialog.

#	Object	Action	Data	Comment			
18	Hire new wo	LClick	28,30	User clicks at: 28	User clicks at: 28, 30 in 'Hire new worker'		
2	First name	LClick	20,8	User clicks at: 20	User clicks at: 20, 8 in 'First name'		
1 3	First name	SetText	John	Do SetText("Joh	Do SetText("John") on First name		
iii 4	Middle name	LClick	26,11	User clicks at: 26	User clicks at: 26, 11 in 'Middle name'		
🖄 5	Middle name	SetText	Α	Do SetText("A")	Do SetText("A") on Middle name		
🖄 🌀	Last name	LClick	22,10	User clicks at: 22	User clicks at: 22, 10 in 'Last name'		
107	Last name	SetText	Smith	Do SetText("Smit	Do SetText("Smith") on Last name		
8 🗐	Job	SelectIt	СТО	Item selected:'CT	FO'		
Verif	y (Ctrl+1)	Learn (Ctrl	+2)	Spy(Ctrl+5)	Pick Object	Resume	
Analog (Ctrl+4)			_Simulated	•	Cancel	Finish (Ctrl+3)	

After this recording session, the corresponding UI area looks as follows:

Create new worker					
First name:	John				
Middle name:	Α				
Last name:	Smith				
Personnel number:					
Job:	сто 👻				
Employment start date:	10/4/2016 12:00:00 am				
Employment end date:	Never				
	Hire new worker				

When the recording is finished Rapise automatically generates the test code:

```
function Test()
{
   //User clicks at: 28, 30 in 'Hire new worker'
   SeS('Hire_new_worker').DoLClick(28, 30);
   //User clicks at: 20, 8 in 'First name'
   SeS('First_name').DoLClick(20, 8);
   //Do SetText("John") on First name
```

```
SeS('First_name').DoSetText("John");
//User clicks at: 26, 11 in 'Middle name'
SeS('Middle_name').DoLClick(26, 11);
//Do SetText("A") on Middle name
SeS('Middle_name').DoSetText("A");
//User clicks at: 22, 10 in 'Last name'
SeS('Last_name').DoLClick(22, 10);
//Do SetText("Smith") on Last name
SeS('Last_name').DoSetText("Smith");
//Item selected:'CTO'
SeS('Job').DoSelectItem("CTO");
}
```

If Rapise does not capture any interaction or captures it wrongly then try to learn the object. In this case Rapise will add it to the object tree but will not capture the action and you'll add the code to the test manually later. To learn an object during recording session place mouse cursor over it and press Ctrl-2 shortcut. It makes sense to pause recording before learning objects. This will prevent Rapise from intersecting mouse and keyboard and attempting to record interactions you do. Pause/Resume button is located at the right side of the Recording dialog.

Tips for Interacting with Objects

Text Box

To allow Rapise to capture the entered text interact with a text box in two steps: 1. Click into the edit box 2. Type text using keyboard

ComboBox

Dynamics AX has several types of combo boxes.

• Standard combo box like Job in the Hire New Worker example above. Rapise treats such combo boxes as atomic objects. To set a value in such a combo use DoSelectItem action:

SeS('Job').DoSelectItem("CTO");

• **Table fiter combo** is recognized as a pair of objects: MenuItem and DropDown. The MenuItem is used to open the DropDown.

		Pers	sonnel number 🔻 🗦	
ie	Extension	*	Name Search name Personnel number	Worker typ
		-	More	

Here is the code generated on recording:

```
//User clicks at: 115, 7 in 'Scope'
SeS('Scope').DoLClick(115, 7);
//DropDown item selected:'Personnel number'
SeS('DropDown').DoSelectItem("Personnel number");
```

• **Lookup field** consists of an edit box, open button and a dropdown table. This is the case when Rapise can record edit box interaction only. To make a choice from the table learn open button and then learn the table.

Account type:	Active Directory user 👻
Default company:	DAT
Enabled:	Company 🔺 Company name
Current partition is default partition:	DAT Company accounts d
External:	
📕 Edit role	

This is the code to lookup a value:

```
// Press open button
SeS('Default_company_lookup_button').DoClick();
// In the first column select a cell with value 'DAT'
SeS('Grid').DoClickCell("DAT", 0);
```

```
// Or alternatively select the first row in the first column
SeS('Grid').DoClickCell(0, 0);
```

Address Bar

Learn the Dynamics AX address bar control using Ctrl-2 shortcut.

Microsoft E	Dynamics AX - Contoso [WIN-DR7HHF5ESQT: Session ID - 3] - [1 - dat]	
00-	DAT	-

Then you can set the path using DoSetText action in your code:

```
SeS('AddressBarContainer').DoSetText('DAT/Retail essentials/Employees/
Users');
```

Menu

Rapise supports both recording and learning of the main menu. When recording make sure you click on every component along the path. E.g. you want to navigate to File > View > Modules > General ledger. During recording click on File, View, Modules and General ledger. Generated code looks like:

```
//Menu item selected:'General ledger'
SeS('File').DoMenu("File;View;Modules;General ledger");
```

Rapise captures menu as top level object (File in the example above). Notice that menu path components are separated with :. If you want for example to open menu File > Tools > Telephone list then write:

```
SeS('File').DoMenu("File;Tools;Telephone list");
```

Table

To work with a table/grid in AX, learn it first and then write the code.

ι	Jsers 👻	7	Type to filter	Account typ	e ▼ →	7 🖫	7
	Account type	Alias	Network domain	User ID 🔺	User name	Company	Enabled
	Active Directory user	Administrato	r CONTOSO	Admin		DAT	1
	Active Directory user			Guest		DAT	
	Active Directory user	squirrel	contoso.com	squirrel	squirrel	DAT	V

// To click on the cell at first column and first row // First column contains checkboxes. First row is 'Administrator' record. SeS('Grid').DoClickCell(0, 0);

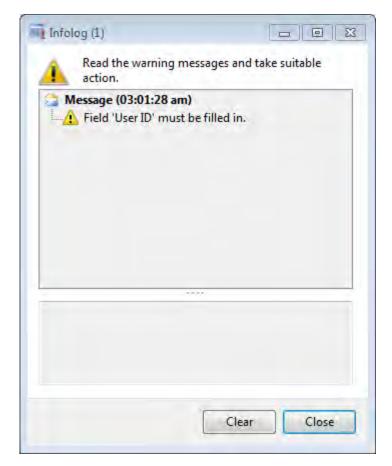
// To click on 'contoso.com' in 'Network domain' column
SeS('Grid').DoClickCell("contoso.com", "Network domain");

```
// Get column name by index (returns 'Network domain')
var columnName = SeS('Grid').GetColumnName(3);
```

```
// Click on column header (clicks on the checkbox to select all rows in the
table)
SeS('Grid').DoClickColumn(0);
```

Infolog

In some case Dynamics AX can report an error using its Infolog window:



To obtain text of the messages in this window learn the Tree object - place cursor over the error text and press Ctrl-2shortcut. In the case of infolog tree - all tree nodes are immediate children of the root. So in the example shown on the image above the tree contains two child nodes of the tree node.

// Get the number of messages in the infolog, 0 - means the root node. // For the presented example it returns 2. var messageCount = SeS('Tree').GetChildrenCount(0);

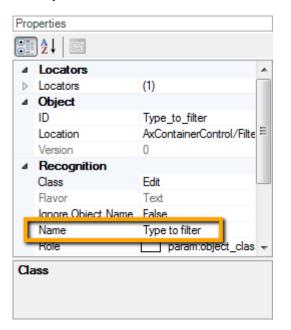
```
// Get second message text, returns "Field 'User ID' must be filled in."
var messageText = SeS('Tree').GetChildAt(0, 1);
```

Type to filter Field

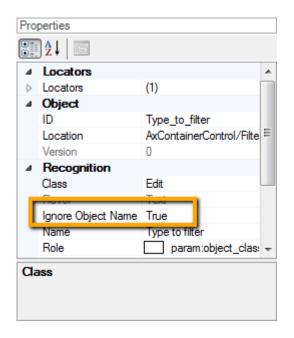
Almost every standard grid in Dynamics AX has a text field for filtering grid data:

Тур	e to	filter	Alias		+
User ID	+	User name	Company	Enabled	External
Admin			DAT	147	
Guest			DAT		
squirrel		squirrel	DAT	12	

When you learn this field it has a name "Type to filter".



However when this field gets focus it's name changes to Text box. To enable Rapise to find this field during playback set the Ignore Object Name property of the object to True.



Dynamics AX Useful Tips & Tricks

Maximize/Minimize/Restore Window

You can maximize a window using any object inside it as a starting point.

```
SeS('AddressBarContainer').getDesktopWindow().Maximized = true;
```

To minimize use

```
SeS('AddressBarContainer').getDesktopWindow().Maximized = false;
// or
SeS('AddressBarContainer').getDesktopWindow().Iconic = true;
```

For restoring (to go back to a smaller window from maximized or minimized state) use

SeS('AddressBarContainer').getDesktopWindow().Iconic = false;

Scroll to the Bottom of a Grid

If your grid looks like this and you want to scroll to the last record of it learn the Last Record button and click on it during test playback:

Unread	Notification type	Subject Due d	ate Notified fc *
4	i)	The batch job Report data clean up comple	Job descrip
1	1	The batch job Report data clean up comple	Job descrij
1	4	The batch job Report data clean up comple	Job descrij
1	0	The batch job Report data clean up comple	Job descrij
1	Ú.	The batch job Report data clean up comple	Job descrij
	4	The batch job Report data clean up comple	Job descrij
)	i)	The batch job Report data clean up comple	Job descrij 🖛

```
SeS('Last_Record').DoAction();
// or
SeS('Last_Record').DoClick();
```

Scroll and Click on a Cell in a Grid

If you know the value of a cell in a grid then Rapise will automatically scroll the grid before clicking:

Workers -		Туре	to filter	Name		▼ →	T 7
Name 🔺	Search name	Personnel num	ber Telephone	Extension	E-mail	Worker type	-
W1 W1	W1 W1	00000009				Employee	
W10 W10	W10 W10	00000027				Employee	=
W11 W11	W11 W11	00000007				Employee	
W2 W2	W2 W2	00000011				Employee	
W3 W3	W3 W3	00000013				Employee	
W4 W4	W4 W4	00000015				Employee	
W5 W5	W5 W5	00000017				Employee	
W6 W6	W6 W6	00000019				Employee	-

Let's assume that you want to click on a cell with value W9 W9 in a column Search name. This cell is not visible on the picture and requires scrolling to show up. The following code performs the click:

SeS('Grid').DoClickCell("W9 W9", "Search name");

If you do not know the value of a cell you can get it this way:

var cellValue = SeS('Grid').GetCell(9, "Search name");

Where 9 is row number.

2.6.4.2 Dynamics CRM

Overview

Microsoft Dynamics CRM is a customer relationship management application from Microsoft, that provides sales, service, and marketing capabilities. Microsoft sells Microsoft Dynamics CRM separately from the ERP products. CRM is available either as on-premises software or as a software-as-a-service offering called "Microsoft Dynamics CRM Online".

Rapise can be used to test installations of Dynamics CRM both on-premise and online.

Dynamics CRM On-Premise

	3100/demo/main.aspx		▼ ++ ×	Bing
Fle Dashboards	icrosoft Dynamics CRM Overview	and the second se	Dynamics CRM	First name Last name @ demo _
Save As Dashboard Management Norkplace	Default Dashboard Action	Lefresh Advanced Find Query Velcome to Microsoft Dynam	ics CRM	
My Work Dashboards		1. Explore	ed by providing a high-level view of your inf 2. Extend	3. Use
Activities Calendar Imports Duplicate Detection		라 About this Get Started Pan 라 More Access to CRM 라 About Dashboards	e - 라 About Customizing 라 Dynamics Marketplace 라 Add More Languages	 → Manage Sample Data → Import Data → Add Users
 Queues Articles Reports Announcements Customers Accounts Contacts 	Dashboard: Microsoft I Sales Pipeline Open Opportunities	L	eads by Source Campaign Il leads in current fiscal year	
Workplace Sales Marketing Service Settings Resource Center		51 0100000	(blank) 0 2 4 6 8 Count:All (Name)	There is no data to create the Cases By Priority (Per Day) chart.

Dynamics CRM Online

Microsoft Dyna	mics CRM 🗸 🏦 SALES 🗸	Accounts	 Contoso Pharmaceu 	Create		n Voorhis 3 Sample	ø
NEW C DEACTIVA	ATE 📓 CONNECT 👻 📑 ADD TO	MARKETING LIS	ST 🚔 ASSIGN 🚥			Ť	4
Account Con	toso Pharmace	utical	s (sample)	Annual Revenue \$60,000.00	No. of Employe 1,500	es Owner* John Vo	oorhis
ummary							
ACCOUNT INFORM	ATION Contoso Pharmaceuticals (samp	POSTS Enter po	ACTIVITIES NOTES	Primary Cor Robert Lyo			
Phone	555-0156		uto pasts. User posts	Em.	ail some	one_g@example	.com
Fax Website	 http://www.contoso.com/	2,	Shipping time information (sample) Case: Closed by First name Last name for Account Contoso Pharmaceuticals			0106	+
Parent Account Ticker Symbol	-		(sample). On Shipping time information (sample)'s wall Today 8/9/2013 9-23 AM	Full Name		mail	-
ADDRESS 9906 Oak Grove Roa Redmond, WA 8080 U.S.		2	100ay 69/2013 %2.2 AM Shipping time information (sample) Case: Created by First name Last name for Account Contoso Pharmaceuticals (sample). On Shipping time information (sample)'s wall Today 8/9/2013 9:23 AM	Robert Lyo	n (sample) sc	omeone_g@exam	ple.c.,
152nd Ave NE	and m15 h 81 h	* [®]	Maintenance time information required (sample) Case: Created by First name Last name for Account Contoso Pharmaceuticals (sample). On Maintenance time information required (sample)'s wall Today 8/97/2013 9:23 AM	< C			>
hst	SAMMAMISH TRAIL Bdg NE 83rd St NE 83rd St Pad		They sell many of the same items that we do - need to follow up (sa, Opportunity: Created by First name Last name for Account Contoso Pharmaceuticals (sample). On They sell many of the same items that we do - need to follow up (sample)'s	RECENT O	PPORTUNITIES	+	
1 1 16 1	3		wall	Title	S	tatus 🛧	Actual
bing	@2013 Nokia @ 2013 Microsoft Corporati		Today 8/9/2013 9:23 AM				

Recording a Dynamics CRM Test

Both versions of Dynamics CRM (server and online) are completely web-based and use a web browser to access the user interface. Therefore when recording a test using Rapise, you use the same web browser libraries that you use to record other <u>web tests</u>:

- Most of the Dynamics CRM user interface will be tested using the **standard browser library** for your web browser of choice (e.g. Internet Explorer HTML, Firefox HTML, and Chrome HTML).
- In addition, there are special controls inside CRM that Rapise has specialized support for. For that reason you'll also see the DomDynamicsCrm library added to your test as well as the browser one. This DomDynamicsCrm library adds additional rules that identify certain CRM objects to make testing easier.

When you record your first test, you'll the following library selection code generated automatically by Rapise:

g_load_libraries=["%g_browserLibrary:Internet Explorer HTML%", "DomDynamicsCrm"];

If you don't see the **DomDynamicsCrm** library listed in your test, then you will need to manually add it.

Recording and Learning Objects

During recording while you interact with Dynamics CRM, Rapise captures actions and displays them in the recording dialog:

Objects	
id n id∏	
🔺 💟 Object Tree C:\Users\Public\Documents\Rapise\Sa	
Microsoft Dynamics CRM	
A Contacts [Contacts]	
Grid [Grid]	
Morkplace [Workplace]	
Contact	
Ifirstname [firstname]	
Iastname [lastname]	

Some of these objects will be standard HTML DOM objects (e.g. hyperlink):

Pro	perties		
•	Ag↓ □		
v	HTML		^
	Page Title	Microsoft Dynamics CRM	
	URL		
	XPATH	/html/body/div[5]/div[@id='crmA	4
v	Locators		
>	Locators	(1)	
v	Object		
	ID	Contacts	
	Version	0	
v	Recognition		
	Class	Internet Explorer_Server	
	Flavor	Link	
	Ianore Object Name	False	
	Library	Internet Explorer H I ML	
	Name	Contacts	
	Role	ROLE_SYSTEM_CLIENT	
	Text	param:object_name	
	Туре	HTMLObject	\mathbf{v}

and others will be specific to Dynamics CRM:

Pro	operties					
	2↓ □					
×	HTML		~			
	Page Title	Microsoft Dynamics CRM				
	URL					
	XPATH	//iframe[@id='contentlFrame']@@				
v	Locators					
>	Locators	(1)				
v	Object					
	ID	Grid				
	Version	0				
v	Recognition					
	Class	Internet Explorer_Server				
	Flavor	Grid				
	Ignore Object Name	False				
	Name	Grid				
	Role	ROLE_SYSTEM_CLIENT				
	Text	Full NameE-mailParent Custome				
	Type	DomDynamicsCrmGrid				

Tips for Interacting with Objects

The following unique objects are available within Dynamics CRM that Rapise has special support for:

Dynamics CRM Grid

One of the the most important UI elements inside Dynamics CRM is the sortable, filterable grid:

Contacts: My Active Contacts -		Sear	ch for records	
🗖 🛛 Full Name 🔺	E-mail	Parent Customer	Business Phone	8
🗖 🚂 John Smith	john@contoso.com			

The Dynamics CRM Grid is used in lots of different screens inside Dynamics CRM (e.g. Contacts, Accounts, etc.) and it is very common to need to interact with it in test scripts.

When you record operations on such a grid or simply learn the entire grid using CTRL+2 you will learn the DynamicsCrmGrid object:

Objects	
v id n id[]	
Øight and State (State State) A state of the state of	
Microsoft Dynamics CRM	
A Contacts [Contacts]	
🔺 🛺 Grid [Grid]	
=le DoClickCell	
=le DoFullText	
GetSelectedRowCount	
GetSelectedRows	
Generic	
HTMLObject	

In additional to the standard HTML object methods and properties, you have the following special functions that you can perform on the grid:

- DoClickCell() Clicks the specified cell when you specify the x-index, y-index, the type of click (left-click, right-click, etc.)
- o DoFullText() Returns the textual representation of the entire table
- o GetCell Gets the text of the specified cell.
- o GetColumnCount Gets the number of columns in grid
- GetColumnName Gets the caption of a column.
- GetRowCount Gets the number of rows in grid
- o GetSelectedRow Gets the index of the selected row.
- o GetSelectedRowCount Gets the number of selected rows.
- o GetSelectedRows Gets the selected rows.

2.6.4.3 Dynamics 365

Overview

Microsoft has created a new cloud-based SaaS-only combined ERP/CRM solution called **Microsoft Dynamics 365**. This provides a new web-based version of <u>Dynamics AX</u> (renamed Dynamics 365 for Operations), combined with a new web based version of Dynamics NAV (called Dynamics 365 for Financials) and an updated Microsoft Dynamics CRM Online.

This new integrated, ERP/CRM is provided solely through Microsoft Azure and is completely web-based. There are two main modules that Rapise has specialized support for:

- 1. **Dynamics 365 for Operations** this is the subject of this section, please read on if you are testing these modules.
- 2. **Dynamics 365 for Sales** this is a rebrand of <u>Dynamics CRM</u> and is covered in section 2 above.

Dynamics 365 v Operations	is.cibualik.dynamics.co P + 🗎 C 👔	Dashboard Operations ×			
Welcome to Microso	ft Dynamics 365 for Op	erations			
			/		and P.V.
🔹 April 2017 🗲	Bank management	Customer credit and collections	Ledger budgets and forecasts	Product readiness for process manufacturing	Reservation management
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	Budget planning	Customer invoicing	Manager self service	Product variant model definition	Resource lifecycle management
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	Catalog management	Customer payments	B Master planning	Production floor management	Retail IT
30 Work items assigned to me • •	Category and product management	Data management	Cutbound work monitoring	Project management	Retail store financials
Expense reports (Record estimate Exturing r catalog apprové) Catalog ("catalog apprové)	Channel deployment	Electronic reporting	Outbound work planning	Purchase order confirmation	Retail store management
	Compensation management	Pb Employee self service	Payroll management	Purchase order preparation	Sales order processing and inquiry
	Cost administration	Financial period close	Personnel	Purchase order receipt	Sales return processing

Recording a Dynamics 365 for Operations Test

Dynamics 365 for Operations is completely web-based (unlike Dynamics AX) and you use a web browser to access the user interface. Therefore when recording a test using Rapise, you use the same web browser libraries that you use to record other web tests:

- Most of the Dynamics 365 user interface will be tested using the **standard browser library** for your web browser of choice (e.g. Internet Explorer HTML, Firefox HTML, and Chrome HTML).
- In addition, there are special controls inside Dynamics 365 that Rapise has specialized support for. For that reason you'll also see the DomDynamicsAx library added to your test as well as the browser one. This DomDynamicsAx library adds additional rules that identify certain Dynamics 365 objects to make testing easier.

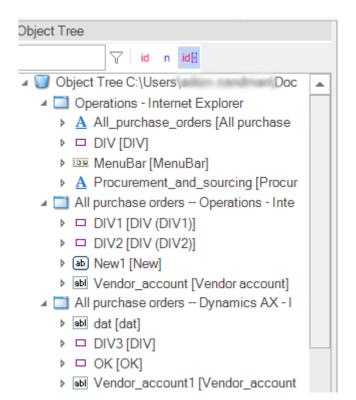
When you record your first test, you'll see the following library selection code generated automatically by Rapise:

g_load_libraries=["%g_browserLibrary:Internet Explorer HTML%", "DomDynamicsAx"];

If you don't see the **DomDynamicsAx** library listed in your test, then you will need to manually add it.

Recording and Learning Objects

During recording while you interact with Dynamics 365, Rapise captures actions and displays them in the recording dialog:



Some of these objects will be standard HTML DOM objects (e.g. hyperlink):

Pro	operties		
•	A↓		
×	HTML		^
	Page Title	Operations	
	URL	https://d365try-6.trial.operat	
	XPATH	/html/body/div[@id='mainC	
>	Locators		
>	Object		
×	Recognition		
	Class	Internet Explorer_Server	
	Flavor	Link	
_	lanore Obiect Na	False	
	Library	Internet Explorer H I ML	
	Name	All purchase orders	
	Role	ROLE_SYSTEM_CLIENT	
	Text	param:object_name	
	Type	HTMLObiect	
v	Window		~

and others will be specific to Dynamics 365:

ro	perties		
•	A ↓		
v	HTML		^
	Page Title	Operations	
	URL	https://usnconeboxax1aos.	
	XPATH	//div[@class='modulesPar	
>	Locators		
>	Object		
Y	Recognition		
	Class	Internet Explorer_Server	
	Flavor	MenuBar	
	Ignore Object Na	False	
	Name	MenuBar	
	Role	ROLE_SYSTEM_CLIENT	
	Text		_
	Туре	DomDynamicsAXMenuBar	г
~	window		
	Window Class	IEFrame	$\mathbf{\mathbf{v}}$

Tips for Interacting with Objects

One of the most important UI elements inside Dynamics 365 is the multi-level menu bar:

Bana management	Quatories predir and collections	Cadge bulgets and Reprivats	Constant matteries for process manufacturing	Contraction management
Budget planning	Customer Invocang	Manager out service	Pindust variant model defention	Basauco Mecycie europerent
Cathlog management	Customer payments	Matter planning	Production foor management	(B) sewill
Category and product manuplement	Data management	Cultured work manifolding	Prost management	B testiter function
Curved Selectment	Taxtori aportog	Cutourities	Purchase order confernation	B Real store
Compensation management	Contraction and service	Parel nearened	Purchasa order presentation	Sides order processing and maxing

The Dynamics 365 menu bar is used to quickly and easily navigate between different parts of the application and it is very common to need to interact with it in test scripts.

When you click on entries in the menu bar or simply learn the entire menu using CTRL+2 you will learn the **DomDynamicsAXMenuBar** object:

- ▲ Image MenuBar [MenuBar]
 - DoMenu
 - Generic
 - HTMLObject

In additional to the standard HTML object methods and properties, you have the following special functions that you can perform on the grid:

 DoMenu(path, separator) – selects the menu entries in specified path, using the specified separator (or semicolon if none specified).

Here is a sample test that was recorded using Dynamics 365 for Operations and Rapise:

```
function Test()
      //Get the sample data from the database
      var conn = 'Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=SampleData.accdb';
      var sql = 'SELECT TEST VALUE FROM TEST DATA';
     Database.DoAttach(conn, sql);
      do
      {
            var testValue = Database.GetValue('TEST VALUE');
            if (testValue)
            {
                  SeS('MenuBar').DoMenu("Modules;Procurement and
sourcing;Purchase orders;All purchase orders");
                  StartNewPurchaseOrder();
                  FillPurchaseOrderForm(testValue);
SeS('MenuBar').DoMenu('', '');
      }
      while (Database.DoSequential())
 }
g load libraries=["%g browserLibrary:Internet Explorer HTML%",
"DomDynamicsAX"];
/** @scenario StartNewPurchaseOrder*/
function StartNewPurchaseOrder()
{
      Global.DoWaitFor('New1');
      SeS('New1').DoClick();
}
/** @scenario FillPurchaseOrderForm*/
function FillPurchaseOrderForm(data)
ł
      Global.DoWaitFor('DIV1');
      SeS('DIV1').DoClick();
      SeS('DIV2').DoClick();
      //Click on DIV
      SeS('DIV3').DoClick();
      //Learned Vendor account
      SeS('Vendor_account1').DoClick();
      //Set Text datss in datss
      SeS('dat').DoSetText(data);
      SeS('OK').DoClick();
```

}

2.6.5 Windows Applications

Overview

Rapise provides comprehensive support for testing Microsoft Windows GUI applications, including applications written using **Win32**, **MFC**, **ATL**, Windows Forms, Visual Basic 6, Microsoft **.NET**, **ActiveX**, and Windows Presentation Framework (**WPF**).

Specifically, Rapise has support for the following technologies used to build Windows applications:

○ Win32 Applications

- Microsoft Foundation Classes (MFC)
- ActiveX Template Library (ATL)
- Visual Basic 6
- ActiveX / COM

• Microsoft .NET Applications

- WinForms
- .NET 1.1,
- .NET 2.0
- .NET 4.x
- Windows Presentation Framework (WPF)
 - Silverlight
 - XAML
- Modern / Metro Apps
- o Third Party Component Libraries
 - Infragistics WinForm Controls
 - Telerik RadControls
 - DevExpress Controls
 - ComponentOne ActiveX Controls
 - SyncFusion Windows Form Controls
 - FarPoint Spreadsheet Control

Choosing the Right Windows Library

Since applications are often built using a mixture of different Windows technologies and component frameworks, we recommend that you use the following approach when testing Windows desktop applications with Rapise:

- 1. First try recording using the "Auto" option, this will let Rapise inspect the application and use the most appropriate libraries. Usually Rapise finds the correct libraries using its auto-detection, but sometimes an application is unusual and the auto-detection fails. If auto-detection is not reliable for your application, move on to step 2.
- Select the "UlAutomation Object" Spy and try to inspect the application. If the Spy shows the real content of the application (i.e. you see object names and IDs) then it is worth trying to record a test with the corresponding library called "UlAutomation" - <u>using the record application dialog</u> <u>box</u>. This library is best for modern Windows applications.
- 3. If the recording is not picked up or does not play back correctly then the next step is to try

recording with the "**Generic**" library. This is designed for older Windows Win32 applications. There is also a Spy tool for this library called the "**Accessible Object**" Spy.

- 4. If you have **third-party components** from Infragistics, Telerik, DevExpress, ComponentOne, SyncFusion or FarPoint, you may also need to select those libraries in addition to "Generic".
- 5. If the recording is not sufficient then the last step is to try and record with the "Advanced Accessibility" library. This library contains definitions of accessible controls at a very basic level. It is not very sophisticated, but it is often sufficient for many cases and works across a wide range of applications.
- 6. If neither of options below satisfy then it may be worth to try low level recording (<u>https://www.inflectra.com/Support/KnowledgeBase/KB114.aspx</u>).

Sample Applications

In the <u>Samples index</u> of this manual we provide a full list of all the samples included with Rapise, however for Windows applications there are a couple of key ones that are worth exploring:

a) Two Dialogs

This is a simple Win32 MFC application that uses the **Generic** library and lets you try some basic testing scenarios.

1
1
el

b) Sample ATM

This is a more robust Win32 MFC application that is based on a banking ATM. You should use the **Generic** library with this application:

SampleATM		_		×
Account Help		*		
Account Transaction				
From Account:		To Account:		_
00000004 0000005 0000009 00000011 00000012 00000013 00000014 00000016	~	00000004 0000005 0000009 00000011 00000012 00000013 00000014 00000016	~	
Amount:				
Submit Transaction		Cancel Tran		

c) WPF Sample Application

This application is a good test of the **UIAutomation** library and is a simple test application with all of the different types of control available for testing:

sicControls	Shapes and	d Containers	More Containers					
Button		ListBox	ListView	Slider		RichTe:	xtBox	
CheckBo		listBoxItem1 listBoxItem2 listBoxItem3	listViewItem2 listViewItem3	TextBlock				
ComboBox	×	listBoxItem4 listBoxItem5		Label		ScrollBar	ScrollVie	ew
 RadioBu RadioBu Status Bar 		Ra	pise	TextBox PasswordBc	x	-	LongConten LongConten LongConten LongConten LongConten	itStri itStri itStri itStri
ProgressBa	ar	_		Tree	 Root Node1 Noc 	de1Node1]
ToolBar	in ToolBarPa	nel	÷			Node1Node Node1Node de1Node2		
ToolBar i	n ToolBarTray	r		7 .	Node3			

2.6.6 Salesforce.com

Overview

Salesforce.com is a cloud-based Customer Relationship Management (CRM) and Sales Force Automation (SFA) platform that is delivered as a Software as a Service (CRM) solution. It provides companies with an interface for case management and task management, and a system for automatically routing and escalating important events.

Home Chatter Campaigns	Leads Accounts Contacts Opportunitie	s Forecasts Contracts	Cases Solutions Produc	ts Reports Dashboards +	
Create New 👻	Accounts Home				Tell me morel Help for this Page
Shortcut					
Unresolved Items	View: All Accounts - C	ot Edit Create New View			
Officioned Roma	and a second	Now			Recently Viewed ~
Recent Items	Recent Accounts	New			Recently viewed V
No records to display	Account Name		Billing City	Phone	
	AB Drivers Limited		San Jose	(244) 982-7016	
Recycle Bin	Reports			Tools	
Quick Create	Active Accounts			Import Accounts & Contacts	
Account Name	Accounts with last activity > 30 days			Mass Delete Accounts	
18	Account Owners			Transfer Accounts	
Phone	Contact Role Report Account History Report			Merge Accounts	
Website	Partner Accounts			Sales Methodologies	
Save	Go to Reports *				

Rapise provides out of the box support for testing Salesforce.com and provides <u>additional libraries</u> for making the testing even easier and more productive.

Recording a Salesforce.com Test

Salesforce.com is completely web-based and you use a web browser to access the user interface. Therefore when recording a test using Rapise, you use the same web browser libraries that you use to record other web tests:

- Most of the Salesforce.com user interface will be tested using the standard browser library for your web browser of choice (e.g. Internet Explorer HTML, Firefox HTML, and Chrome HTML).
- In addition, there are special controls inside Salesforce.com that Rapise has specialized support for. For that reason you'll also see the DomSalesforce library added to your test as well as the browser one. This DomSalesforce library adds additional rules that identify certain Salesforce.com objects (e.g. the main Grid) to make testing easier.

When you record your first test, you'll see the following library selection code generated automatically by Rapise:

g_load_libraries=["%g_browserLibrary:Internet Explorer HTML%",
"DomSalesforce"];

If you don't see the **DomSalesforce** library listed in your test, then you will need to <u>manually change it</u> using the **[] Libraries** button on the tool menu.

Tips for Interacting with Objects

Many of the screens in Salesforce.com use simple HTML form fields such as text boxes, dropdown lists, etc.:

New Lead						Help for this Page
ead Edit	S	ave Save & Nev	Cancel			
Lead Information						e Required Information
Lead Owner	Denis Markovtsev			Phone	100	
First Name	None v			Mobile		
Last Name	1			Fax		
Company	1			Email		
Title				Website		
Lead Source	None ~			Lead Status	Open - Not Contacted v	
Campaign		-		Rating	None v	
Industry	None ~			No. of Employees		
Annual Revenue						
Address Information						
Street						
		al				
City						
State/Province						
Zip/Postal Code						
Country						
Additional Information						
Product Interest	None			Current Generator(s)		C
SIC Code				Primary		Chat

Rapise will recognize these objects using the standard web browser HTML libraries.

Salesforce.com Grid

One of the the most important UI elements inside Salesforce.com is the sortable, filterable grid:

All Open	Leads v Edit D	elete Create New View					TITT List	Feed 📄
New Lead Char	nge Status Change Owner A	dd to Campaign		ABC	DEFGHIJ	K L M N O F	P Q R S T U V W	X Y Z Other
Action	Name +	Company	State/Province	Email	Lead Status	Created Date	Owner Alias	Unread By Owne
Edit Del 🕀	Akin, Kristen	Aethna Home Prod	VA	kakin@athenahom	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🔁	Braund, Mike	Metropolitan Hea	MD	likeb@metro.com	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🕀	Bunny, Bugs	MerriMelodies Inc.		b.bunny@merriem	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🔀	Centeno, Morgan	Nelson Inc	CA	im.im@example	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🔀	Crenshaw, Carolyn	Ace Iron and Steel	AL	carolync@aceis	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🔂	Dadio Jr, Bill	Zenith Industrial Pa	OH	bill dadio@zenith.c	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🚯	Daigle, Jaclyn	Nelson Inc	CA	the97@example.it	Open - Not Contacted	7/30/2014	samt	1
Edit Del 😲	Deaton, Luann	Nelson Inc	CA	kid43@example.de	Open - Not Contacted	7/30/2014	samf	~
Edit Del 😲	Doo, Scooby	Doo Inc.		scooby@doo.com	Open - Not Contacted	7/30/2014	samf	1
Edit Del 😳	Duck, Donald	Duck Tales Inc		d.duck@ducks.com	Open - Not Contacted	7/30/2014	samf	1
Edit Del 😲	Eberhard, Sandra	Highland Manufa	CA	sandra e@highlan	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🜐	Feager, Patricia	International Shi	NC	patricia feager@is	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🜐	Flinstone, Fred	Stone Age		f.flinstone@stonea	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🚭	Gesr, Gesr	ex.grinmark.com		gesr@ex.grinmark	Open - Not Contacted	4/13/2017	gadmin	
Edit Del 🚭	Glimpse, Jeff	Jackson Controls		jeffg@jackson.com	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🚭	J, Danny	Brix			Open - Not Contacted	7/30/2014	samf	1
Edit Del 😗	James, Tom	Delphi Chemicals	MN	tom.james@delphi	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🚭	Kasten, Kara	Nelson Inc.	CA	dev.support.qa@ex	Open - Not Contacted	7/30/2014	samf	1
Edit Del 🚭	Ladouceur, Grover	Hammer Group Inc	CA	ga69@example.de	Open - Not Contacted	7/30/2014	samf	1
Edit Del 😲	Luce, Eugena	Pacific Retail Group	MA	eluce@pacificretail	Open - Not Contacted	7/30/2014	samf	1
Edit Del G	Maccleod, Violet	Emerson Transport	GA	violetm@emersontr	Open - Not Contacted	7/30/2014	samf	1

The Salesforce.com Grid is used in lots of different screens inside Salesforce.com (e.g. leads, quotes,

etc.) and it is very common to need to interact with it in test scripts.

When you record operations on such a grid or simply learn the entire grid using CTRL+2 you will learn the SalesforceGrid object

In additional to the standard HTML object methods and properties, you have the following special functions that you can perform on the grid:

- DoClickCell() Clicks the specified cell when you specify the x-index, y-index, the type of click (left-click, right-click, etc.)
- o DoClickText() Clicks on a specific text inside the grid.
- o GetCell Gets the text of the specified cell.
- o GetColumnCount Gets the number of columns in grid
- o GetColumnName Gets the caption of a column.
- GetRowCount Gets the number of rows in grid

2.6.7 Qt Framework

Purpose

Rapise includes support for testing applications written using the Qt Framework written using QWidget controls.

Usage

Rapise fully supports the test automation of Qt based applications. To ensure that Rapise can access the UI elements and properties in the Qt application, MSAA (Microsoft Active Accessibility) support for your Qt application must be enabled. This provides additional information on Qt UI elements to automation software like Rapise and can be accomplished by shipping and loading the "Accessible Plug-in" included in the Qt SDK (Software Development Kit) with the Qt application under test (see below).

Loading Accessible Plug-in for your Qt application:

1. Copy the "accessible" directory (and all its contents) from the Qt SDK (used to build the application under test) installation folder to the folder of the automated application (e.g. "Program Files/Your-Application/plugins"). If you do not have access to the Qt SDK which the Qt application is developed with, please contact the developer of the application and request the "accessible" directory from him.

2. Create a file called "**qt.conf**" (or append if the file already exists) in the root directory of the automated application (e.g. "**Program Files/Your-Application**") with following content (copy and paste the following two lines):

[Paths]
Plugins = plugins

Sample

In the Samples section of the Rapise Start Page you should see our sample Qt Framework AUT:

🔃 Sample (QtWidget Application		- 0	×
<u>F</u> ile <u>V</u> iew				
Ш				
Sample	• OtWidget Application for R			
Name:		Filename:		
Description:		Font:	Choos	e
		Color:	Change Color	
Location:	Headquarters	Print:	Print	
Options:	• Option 1 C Option 2	Input:		
Active:	Is Active?	Errors:	, Display Errors	
Font:	MS Shell Dlg 2	Enors.		
Date/Time:	12:00:00 AM 🔆 1/1/2000 📩	Message:	Display Message	
Date/Time:	1/1/2000 12:00:00 AM			
Notes:				
	OK Cancel]		
				1

You can use this application with Rapise to try out the Qt Framework testing capabilities:

- o You should record tests using the special Qt Framework library
- You should use the Accessible Spy for inspecting objects.

2.6.8 Java AWT/Swing

Purpose

Rapise supports the testing of Java applications using either the Abstract Window Toolkit (AWT) or Swing graphic user interface toolkits. For maximum flexibility, Rapise can connect to your choice of JVM.

Java Bridge Installation

In order to use a particular Java Virtual Machine (JVM) with Rapise you need to install Java Bridge into it. Installation process consists of several simple steps:

1. Click the Options icon in the Tools group of the main Rapise ribbon. That will bring up the Options dialog.

2. Click on the Tools > Java Settings button. This will launch the Java Bridge installation dialog:

Java virtual mac	hines found in comm	on places:	
	(ja∨a\jdk1.6.0_21\bin (x86)\ja∨a\jre6\bin		
Install in All	Install	Search disks	Cancel

- 3. Choose target JVM in the list of available Java machines and press Install button
- 4. Verify that the installation is successful

To verify that the bridge installed correctly, check that the following files have been installed inside your Java VM (typically found at C:\Program Files (x86)\Java\jre1.x.x_xxx):

- o lib\accessibility.properties
- o lib\ext\jaccess.jar
- o lib/ext/smartestudio-bridge.jar

If you don't see **all three of these files** then it means the bridge was not installed correctly.

Troubleshooting the Configuration

To help you make sure that your environment is correctly setup and also to help you try out Rapise, we have a sample application called AUTJava (AUT = Application Under Test) that can be found in the folder: C:\Users\Public\Documents\Rapise\Samples\Java\AUTJAVA

To run the application, right-click on the x86run.cmd file and choose Run as Administrator.

🛓 AUTJAVA		_		\times
AWT label1				
AWT TextArea	button1	AWT Li AWT Li AWT Li	stitem1 stitem2 stitem3 stitem4 stitem5	
open dialog Adv	vanced AWT Control	s		
Swing jButton2 jToggleButt jCheck	Item 1 ^ Item 2 v Item 3 v	JTree Colors Sports food	-	
) jRadioButton1) jRadioButton2	tab1 tab2	Advanced	Swing Co	nt
Item 1 v jTextField1		jButton3		
Swing TextArea Second Line	2			^ ~

If the application doesn't appear correctly then you may need to set the **JAVA_HOME** environment variable.

To do this, open up the Windows control panel and choose **System > Advanced System Settings**:

System Propertie	S						\times
Computer Name	Hardware	Advanced	System Prote	ection	Remote		
You must be lo Performance Visual effects					virtual mer	-	
– User Profiles – Desktop setti	ngs related t	to your sign-i	n		S <u>e</u> tti	ngs	
Startup and Re		ilure, and de	bugging inform	nation	Se <u>t</u> ti	ngs	
			[Envir	o <u>n</u> ment Va	ariables]
		(ОК	Cano	cel	<u>A</u> pply	

Click on the Environment Variables button:

Variable	Value				
Path	%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;				
TEMP	%USERPROFILE%\AppData\Local\Temp				
TMP	%USERPROFILE%\AppData\Local\Temp				
	New Edit Delete				
	<u>N</u> ew <u>E</u> dit <u>D</u> elete				
	<u>N</u> ew <u>E</u> alt <u>D</u> elete				
uctem variables	<u>N</u> ew <u>E</u> ait <u>D</u> eiete				
ystem variables	<u>N</u> ew <u>E</u> ait <u>D</u> eiete				
ystem variables Variable	Value				
Variable	Value				
Variable ANDROID_HOME	Value C:\Program Files (x86)\Android\android-sdk				
Variable ANDROID_HOME ComSpec	Value C:\Program Files (x86)\Android\android-sdk C:\WINDOWS\system32\cmd.exe				
Variable ANDROID_HOME ComSpec JAVA_HOME	Value C:\Program Files (x86)\Android\android-sdk C:\WINDOWS\system32\cmd.exe C:\Program Files (x86)\Java\jre1.8.0_112 C:\Program Files\Microsoft MPI\Bin\				
Variable ANDROID_HOME ComSpec JAVA_HOME MSMPI_BIN	Value C:\Program Files (x86)\Android\android-sdk C:\WINDOWS\system32\cmd.exe C:\Program Files (x86)\Java\jre1.8.0_112 C:\Program Files\Microsoft MPI\Bin\				
Variable ANDROID_HOME ComSpec JAVA_HOME MSMPI_BIN NUMBER_OF_PROCESSORS	Value C:\Program Files (x86)\Android\android-sdk C:\WINDOWS\system32\cmd.exe C.\Program Files (x86)\Java\jre1.8.0_112 C:\Program Files\Microsoft MPI\Bin\ 8				
Variable ANDROID_HOME ComSpec JAVA_HOME MSMPI_BIN NUMBER_OF_PROCESSORS OS	Value C:\Program Files (x86)\Android\android-sdk C:\WINDOWS\system32\cmd.exe C.\Program Files (x86)\Java\jre1.8.0_112 C:\Program Files\Microsoft MPI\Bin\ 8 Windows_NT				
Variable ANDROID_HOME ComSpec JAVA_HOME MSMPI_BIN NUMBER_OF_PROCESSORS OS Path	Value C:\Program Files (x86)\Android\android-sdk C:\WINDOWS\system32\cmd.exe C\Program Files (x86)\Java\jre1.8.0_112 C:\Program Files\Microsoft MPI\Bin\ 8 Windows_NT C:\Program Data\Oracle\Java\javapath;C:\Program Files\Micro				

Click on the option to add a **System Variable** and then add the following:

Variable = JAVA_HOME Value = C:\Program Files (x86)\Java\jre1.x.x_xxx

(you will need to match the location of your actual Java VM)

Now you should be able to launch the AUTJava sample application.

AUTJAVA – C × AWT label1 AWT TextArea button1 AWT Listthem1 AWT Listthem2 AWT Listthem3 AWT Listthem4 AWT Listthem4 AWT Listthem4 AWT Listthem4 AWT Listthem4 AWT Listthem4 AWT Listthem4 AWT Listthem4 AWT Listthem5 Open dialog Advanced AWT Controls Swinq Item 1 jToggleButt jToggleButt jCheck jCheck jRadioButton2 tab1 tab2 Advanced Swing Cont jButton3 Swing TextArea Second Line					
AWT TextArea button1 AWT ListItem1 AWT ListItem2 AWT ListItem3 AWT ListItem4 AWT ListItem5 Open dialog Advanced AWT Controls Swing jButton2 Item 1 jToggleButt jCheck jRadioButton1 jRadioButton2 item 1 jButton2 item 1 jRadioButton2 item 1 jButton2 item 1 jRadioButton1 jRadioButton2 item 1 jButton3 Advanced Swing Cont	🛓 AUTJAVA		_		\times
Swing jButton2 jToggleButt jCheck jRadioButton1 jRadioButton2 item 1 item 1 item 1 jButton2 item 3 item 1 jButton2 item 1 jButton2 item 3 item 1 jButton3	AWT label1				
jButton2 jToggleButt jCheck jRadioButton1 jRadioButton2 Item 1 v jTextField1 Swing TextArea	< >	Choice1	AWT Lis AWT Lis AWT Lis AWT Lis AWT Lis	stitem2 stitem3 stitem4	
jButton2 jToggleButt jCheck jRadioButton1 jRadioButton2 Item 1 v jTextField1 Swing TextArea			_		
O jRadioButton2 tab1 tab2 Advanced Swing Cont Item 1	jButton2 jToggleButt	Item 2	colors		
Item 1 v jButton3 jTextField1 Swing TextArea	<u> </u>	tab1 tab2	Advanced	Swing Cor	ıt
	Item 1 v		jButton3		
~ ~	-				^
					~

To verify that Rapise is configured correctly, click on the SPY menu in Rapise and choose Java Spy.

Then click on the main \boldsymbol{Spy} icon and the Java Spy will start up:

Rapise Spy		
Start Tracking (Ctrl-G)		· ⊞
- Java Object Tree	Properties	
HWND Tree	- Properties	
-		

Click the CTRL+G button combination to start tracking and then move the mouse over one of the buttons in the sample application and click CTRL+G again.

You should see the following:

Rapise Spy		- 🗆 X
Start Tracking (Ctrl-G)		\boxplus
Java Object		
Tree	- Properties	
advAwtControls[java.awt.Button]	Y Primary	
	Handle	659416
	IndexInParent	5
	Name	advAwtControls
	VmID	13046412
	Secondary	
	accessibleContext	java.awt.Button\$AccessibleAW
	actionCommand	Advanced AWT Controls
	actionListeners	[Ljava.awt.event.ActionListener
	background	java.awt.Color[r=240,g=240,b=2
	enabled	true
	focusable	true
	font	java.awt.Font[family=Dialog,na
- HWND		
Tree	 Properties 	
Advanced AWT Controls[Button]	PosX	120
	PosY	160
	PPID	14040
	> Process	
	ProcessName	C:\Program Files (x86)\Java\jre
	> Size	140, 24
	✓ Primary	
	AccessibleObject	Advanced AWT Controls[ROLI
	ClassName	Button
	> Rect	120, 160, 140, 24
	Text	Advanced AWT Controls
	Visible	True

Which shows that Rapise is able to see the AWT button (in this example) and its properties. However if you see the following instead:

Apise Spy	- 0	×
Start Tracking (Ctrl-G)		- 11
Java Object Tree	- Properties	
HWND Tree button1[Button]	Properties PosX 180 PosY 60	^
	PPID 21976 Process ProcessName C:\Program Files (x86) Size 80, 24 Primary AccessibleObject button1[ROLE_SYSTED ClassName Button Rect 180, 60, 80, 24	
	Text button1 Visible True WindowStyle 1409286155	~

it means that you didn't run the sample application using "Run as Administrator", close the application and try again using "Run as Administrator" and you will see:

🔎 Rapise Spy	- 0	×
Start Tracking (Ctrl-G)		÷ 1
- Java Object		
- Tree	Properties	
advAwtControls[java.awt.Button]	✓ Primary	^
	Handle 659416	
	IndexInParent 5	
	Name advAwtControls	
	VmID 13046412	
	✓ Secondary	
	accessibleContext java.awt.Button\$Accessi	bleAW
	actionCommand Advanced AWT Controls	5
	actionListeners [Ljava.awt.event.ActionL	istener
	background java.awt.Color[r=240,g=2	40,b=2
	enabled true	
	focusable true	
	font java.awt.Font[family=Dia	log,na 🗸
- HWND		
- Tree	Properties	
Advanced AWT Controls[Button]	PosX 120	^
	PosY 160	
	PPID 14040	
	Process	
	ProcessName C:\Program Files (x86)\J	ava∖jre
	ProcessName C:\Program Files (x86)\J Size 140, 24	ava∖jre
		ava\jre
	> Size 140, 24	-
	 Size 140, 24 Primary 	
	Size 140, 24 Primary AccessibleObject Advanced AWT Controls	-
	 > Size 140, 24 Primary AccessibleObject Advanced AWT Controls ClassName Button 	s[ROLI
	 > Size 140, 24 Primary AccessibleObject Advanced AWT Controls ClassName Button Rect 120, 160, 140, 24 	s[ROLI

You are now ready to start testing your real application. Make sure to also start it using **"Run as Administrator**".

Analyzing the Java Application using the Java Spy

With Spy you can walk along the tree of Java objects in your application.

🕎 Start Tracking (Ctrl-G)	H •
 Java Object Tree panel0[java.awt.Panel] Iabel0[java.awt.Label] textArea1[java.awt.TextArea] list1[java.awt.List] button1[java.awt.Button] 	Properties Primary Handle 984994 IndexInParent 3 Name button 1 VmID 198698
- button2[j; Parent - advAwtC Highlight - checkbo: - textField Refresh - choice2[java.awt.cnoicej	Secondary accessibleConte java.awt.Button\$Access actionCommand button1 actionListeners [Ljava.awt.event.Action background java.awt.Color[r=240,g= enabled true focusable true font java.awt.Font[family=Dia foreground java.awt.Color[r=0,g=0,b label button1 name button1 visible true

Save to File

You can save the Spy data for a particular node and all its descendants to a text file.

👽 SeS Spy					x
💩 Start Tracking (Ctr	I-G)				•
 Java Object Tree button1[java.awt.Bu^{***} 			J		
Dutton njava awi bu	Parent Highlight		Handle IndexInParent Name	133978 3 button 1	
	Refresh Save To File	E	VmID Secondary accessibleContext	68416	
	Reflection		actionCommand actionListeners	button1 [Ljava.awt.event.Action]	
HWND			Properties		
····· button1[Button]					

button1 - Notepad	-
File Edit Format View Help	
<pre>button1 java.awt.Button accessibleContext java.awt.Button\$AccessibleAWTButton@34a1fc actionCommand button1 actionListeners [Ljava.awt.event.ActionListener;@12a54f9 background java.awt.Color[r=240,g=240,b=240] enabled true focusable true font java.awt.Font[family=Dialog,name=Dialog,style=plain,size=12] foreground java.awt.Color[r=0,g=0,b=0] label button1 name button1 visible true</pre>	*
۲	

Reflection Information

You can save reflection information for a java class used to implement a GUI control.

👽 SeS Spy			
💩 Start Tracking (C	Ctrl-G)		
🗐 Java Object			
Tree		Properties	
button1[java.awt.B u	Parent Highlight Refresh Save To File	Primary Image: Arrow of the secondary Handle 133978 IndexInParent 3 Name button 1 VmID 68416	
	Reflection	accessibleContext java.awt.Button\$Accessil actionCommand button1 actionListeners [Ljava.awt.event.ActionLi +	
HWND Tree Wutton1[Button]		Properties	

🖉 java.awt.Button - Notepad
File Edit Format View Help
<pre>java.awt.Button field: label java.lang.String java.awt.Button.label field: actionCommand java.lang.String java.awt.Button.action field: actionListener transient java.awt.event.ActionListener field: base private static final java.lang.String java.awt. field: nameCounter private static int java.awt.Button.name field: serialVersionUD private static final long java. field: buttonSerializedDataVersion private int java.awt.Bu method: writeObject private void java.awt.Button.writeObject method: initIDs private static native void java.awt.Button.init method: constructComponentName java.lang.String java.awt.Butto getAccessibleContext public java.awt.Button.eventEnabled(ja method: getListeners public java.awt.Button.eventEnabled(ja method: getActionCommand public java.lang.String java.awt.Button method: getActionListener public java.lang.String java.awt.Button method: getActionListener public java.lang.String java.awt.Button method: setActionListener public java.lang.String java.awt.Button method: getActionListener public java.lang.string java.awt.Button method: getActionListener public java.lang.string java.awt.Button method: setActionListener public java.lang.string java.awt.ev method: getActionListener public java.lang.string java.awt.ev method: setActionCommand public java.lang.string java.awt.button method: setActionListener public synchronized void java.awt.ev method: setActionListener public synchronized void java.awt.ev method: setActionCommand public void java.awt.Button.set method: setActionCommand publ</pre>

2.6.9 Java SWT

Overview

The **Java SWT** GUI library is an alternative to Swing developed by the **Eclipse** foundation and it provides Java applications the ability to access the **native GUI libraries** of the operating system using JNI (Java Native Interface) in a manner that is similar to those programs written using operating system-specific APIs. Programs that call SWT are portable, but the implementation of the toolkit, despite part of it being written in Java, is unique for each platform.

Rapise supports the testing of applications written using Java **Standard Widget Toolkit (SWT)** using its **JavaSWT** extensions library (which is based on the UI Automation technology in Windows). Since SWT displays applications using native Windows controls it doesn't need the Java Access Bridge to be installed (unlike Java AWT/Swing applications).

Sample Application

In the Samples section of the Rapise Start Page you should see our sample Java SWT AUT:

💶 Test Jav	va SWT A	Applicati	ion						_		×	
Sample J Sample	ava SWT Text Box		ntion									
Tab 0	Tab 1	Tab 2										
Buttor		Button	Sim	ple B	utton		Check	This	○ Rad ○ Rad			
Comb	os and S ~	liders		~	Thi	s a ve	ery sin	nple <u>I</u>	<u>ink</u> wid <u>c</u>	get.		
<									>			
	1 1	1	I	I	I	1	I	1	1			
			•									

You can use this application with Rapise to try out the Java SWT testing capabilities:

- You should record tests using the special SWT library
- o You should use the UIAutomation Spy for inspecting objects.

2.7 Extensibility

The Extensibility section is for experienced Rapise users who want to extend capabilities of the tool.

2.7.1 Tutorial: Custom Library

In this section, you will learn how to create a **Custom Library** and add support for a third-party GUI control to Rapise. We will be using a demo application called **CustomControlApp**. Our Custom Library will be simple. It will allow to Record and Learn objects of **CustomListboxControl** type and also Playback actions for this type of objects. This tutorial is complemented by a ready test **CustomControlTest** which you'll be able to examine and run.

Tutorial Data

- CustomControlApp folder: C:\Program Files\Inflectra\Rapise\Samples\Extensibility\CustomLibrary \CustomControlApp. You may build this application yourself in Microsoft Visual Studio (C++) or use ready executable: <CustomControlApp folder>\Release\CustomControlApp.exe
- CustomControlTest folder: C:\Program Files\Inflectra\Rapise\Samples\Extensibility\CustomLibrary \CustomControlTest
- CustomLibrary file: C:\Program Files\Inflectra\Rapise\Samples\Extensibility\CustomLibrary \CustomLibrary.js

If you prefer active experimentation learning style you may first skip to subsection 9 and after playing with the ready test and library start reading from the beginning.

1. Application Under Test

CustomControlApp contains an object of type CustomListboxControl. The control is similar to a singleselect listbox, but each line item has a corresponding **progress bar** indicator indicating a current value. Using the left/right cursor keys you can change the value of the currently focused item.

CustomControlApp	
String 1 String 2 String 3 String 4 String 5 String 6 String 7	
Sound La	

If you will try to record a test for CustomControlApp using just Generic library you'll see that CustomListboxControl is treated as Simulated Object and all interactions with it are recorded as mouse clicks and key presses. For some tests such functionality is sufficient, but if you want to be able to recognize CustomListboxControl as a list, get its items, select an item by name, set value for a particular item you need to create a Custom Library.

Recordin	ng activity for "C	ustomCont	rolApp"			
#	Object	Action	Data	Comment		
1	String 2	LClick	189,5	Mouse click at:	189, 5 in 'String 2'	
102	Global	SendK	{RIGHT}	Туре		
10 3	Global	SendK	(RIGHT)	Туре		
1 4	Global	SendK	{RIGHT}	Туре		
-						
L						
	y (Ctrl+1)	Learn (Ctr	l+2) 🕥	Spy(Ctrl+5)	Pick Object	Pause
Analo	ig (Ctrl+4)		_Simulated	-	Cancel	Finish (Ctrl+3)
Last capt	Last captured: SeSSimulated (CustomControlApp)					

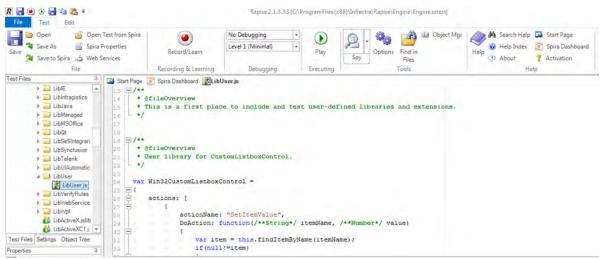
2. LibUser

A good place to start implementing a Custom Library is empty LibUser library included into Rapise. All Rapise libraries live in *C:\Program Files\Inflectra\Rapise\Engine\Lib* folder and LibUser is not an exception. LibUser library consists of two files:

- 1. C:\Program Files\Inflectra\Rapise\Engine\Lib\LibUser.jslib which is a library declaration file.
- 2. C:\Program Files\Inflectra\Rapise\Engine\Lib\LibUser\LibUser.js which is a library definition file.

3. Open Engine.sstest

Open the <u>Engine.sstest</u> project in Rapise (it is usually located in the *C:\Program Files (x86)\Inflectra \Rapise\Engine* folder). Then find LibUser.js in the project tree and open it. You are about to start implementing a Custom Library to support CustomListboxControl.



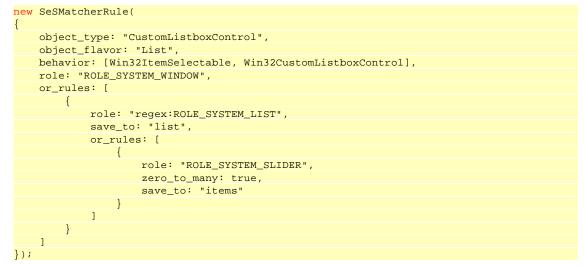
4. Analyze CustomListboxControl in Spy

Launch CustomControlApp and open <u>Spy</u>. Using the **Accessible** option in the Spy tool, spy on the CustomListboxControl. It is easy to see that CustomListboxControl has the following accessibility tree: ROLE_SYSTEM_WINDOW top node contains ROLE_SYSTEM_LIST child that in its turn may contain zero to many ROLE_SYSTEM_SLIDER nodes.

es ext focusable y unt 7 CHILDID_SELF Action tion
ext focusable : Y unt 7 CHILDID_SELF Action tion
Y T T T T T T T T T T T T T T T T T T T
[CustomListboxControl] [Parent 0 nRECT {X=83,Y=105,Width=484,Height=262}
ies
dth 484 83 105 2992
IS C:\Users\Public\Documents\Rapise\Sample: 484, 262

5. Create Matcher Rule for CustomListboxControl

With knowledge of CustomListboxControl accessibility tree we can create a **matcher rule** that will make CustomListboxControl recognizable by Rapise. Write the following code into LibUser.js:



Each matcher rule (instance of SeSMatcherRule) is a tree like structure that describes a particular GUI control type. Each node in this tree is a rule object that is defined by the following simplified grammar:

```
or_rules: (rule)+
and_rules: (rule)+
```

rule:		
role	ble	
[save	save_to]	
[zer	zero_to_many]	
[or_:	or_rules]	
[and	and_rules]	

- object_type: the string that uniquely identifies this matcher rule and designates type of the control
- object_flavor: visual type of the control, it is used to show an appropriate icon in the <u>Object Tree</u> and to filter actions and properties in composite behavior patterns (like in Adobe Flex, see FlexActions.js)
- behavior: array of behavior patterns that define object actions, properties and events.
- **role**: accessibility role of the corresponding node in the accessibility tree of the control. The role equals to a Role of the accessible element as displayed in the Spy.
- or_rules: array of rules (defining child nodes) joined with logical OR. Any OR rule can be satisfied to consider child nodes matched.
- and_rules: array of rules (defining child nodes) joined with logical AND. All AND rules must be satisfied to consider child nodes matched.
- save_to: SeSObject created for accessibility tree node corresponding to this rule is assigned to the field with "save_to" name of the top level SeSObject. I.e. if rule has save_to: "items" element then you can access learned element using SeS('ObjID').items. In many cases such named fields are used in behavior patterns.
- zero_to_many: if this property is present in the rule and set to 'true' then it means that parent rule may contain from zero to many of child nodes that match this rule.

6. CustomListboxControl Behavior

After defining the matcher rule we can proceed to **behavior patterns**. Behavior patterns operate with **SeSObject** contents, so they should not be aware about accessibility tree of the underlying GUI control and thus the same behavior pattern can be assigned to different matcher rules. There are a plenty of behavior patterns defined in SeSBahavior.js. After looking at those patterns it is possible to notice that Win32ItemSelectable pattern is the one that perfectly suites for capturing selection accessibility events and for selecting list items. This pattern contains OnSelect event that is called during recording when an item is selected in list and DoSelectItem action used to select desired item during playback.

But using just Win32ItemSelectable behavior pattern is not sufficient. It does not support recording of progress bar value change events and it does not support setting progress bar value during playback. That is why we need to define new behavior pattern: Win32CustomListboxControl. Look at its code:

ar Win32CustomListboxControl =					
{					
actions: [
{					
actionName: "SetItemValue",					
DoAction: function(/**String*/ itemName, /**Number*/ value)					
<pre>var item = this.findItemByName(itemName);</pre>					
if(null!=item)					
{					
item.getTopObject().instance.HWND.SetForegroundWindow();					
<pre>item.instance.Value = value;</pre>					
return true;					
}					
return false;					
}					
},					
{					
actionName: "GetItemValue",					



During recording process OnValueChange function captures progress bar change events and calls RegisterAction function that adds SetItemValue action to the test.

7. CustomListboxControl Specific Accessibility Events

What accessibility events are fired when a user changes the progress bar value? You can use Spy to find out. Launch CustomControlApp and open Spy window. Spy on CustomListboxControl. Choose Monitor Events...

🥺 SeS Spy			6	- 0 🗙
살 Start Trackin	ng (Ctrl-G)			
- Accessible Ob	ject			
- Tree		- Proper	rties	
(ROLE_SYSTEM String 1[R String 2[R String 3[R String 4[R String 4[R	OLE_SYSTEM_ OLE_SYSTEM_ OLE_SYSTEM_ OLE_SYSTEM_ OLE_SYSTEM_		Paren 0 nREC {X:	ustomListboxC =411,Y=328,\ DLE_SYSTEN T
- Tree		- Proper	rties	
[CustomL	Parent	jar		4; 264 🔺
	Mouse Click Highlight		ame Cu	DLE_SYSTEP stomListboxC 1; 328; 484; 2
	Monitor Events		Tru	Je v

You will see Accessible Events dialog:

A	cessible E	vents			
	#	Туре	Aac		h∿
	📃 Skip			📝 Ignore Mouse Move	Clear

Select an item in CustomControlApp and advance its progress bar using right key. Accessible Events dialog will show you captured events:

#	Туре	Aac	hWnd	
1	EVENT_SYSTEM_FOREGROUND	ROLE_SYSTEM_WINDOW/sizeable moveable focusable	0x00170a38	0x0
2	EVENT_OBJECT_FOCUS	ROLE_SYSTEM_CLIENT/focusable	0x00170a38	
3	EVENT_OBJECT_FOCUS	ROLE_SYSTEM_LIST/focused focusable	0x0007188c	
4	EVENT_OBJECT_SELECTION	ROLE_SYSTEM_SLIDER/selected focused focusable selectable	0x0007188c	
5	EVENT_OBJECT_FOCUS	ROLE_SYSTEM_SLIDER/selected focused focusable selectable	0x0007188c	
6	EVENT_OBJECT_VALUECHANGE	ROLE_SYSTEM_SLIDER/selected focused focusable selectable	0x0007188c	
7	EVENT_OBJECT_VALUECHANGE	ROLE_SYSTEM_SLIDER/selected focused focusable selectable	0x0007188c	
8	EVENT_OBJECT_VALUECHANGE	ROLE_SYSTEM_SLIDER/selected focused focusable selectable	0x0007188c	
9	EVENT_OBJECT_NAMECHANGE	ROLE_SYSTEM_CURSOR/floating	0x00000000	
•		m		Þ

You can see that changing progress bar leads to generation of EVENT_OBJECT_VALUECHANGE events.

Not all accessibility events a processed and propagated by Rapise engine. EVENT_OBJECT_VALUECHANGE is one of such events. To consume this event and make an appropriate call to OnValueChange of Win32CustomListboxControl you need to add and register custom accessibility event handler:

```
function CustomRegisterAccessibleEvent(evt, etxt)
{
    if(etxt.indexOf("EVENT_OBJECT_VALUECHANGE")>=0)
    {
        var ao;
        try
        {
           ao = evt.AccessibleObject;
           if(!_SeSisValidObject(ao)) return false;
        }
        catch(e)
        {
           Log("Error getting event object:"+e.Description+"/"+etxt);
           return false;
        }
        var ro = SeSCacheAccessibleObject(ao);
        if (13 && ro) Log3("CustomListboxControl: " + ro.toString());
        if (ro != null && ("OnValueChange" in ro))
        {
            ro.OnValueChange();
        }
        return true;
    }
    return false;
```

g_customEventHandlers.push(CustomRegisterAccessibleEvent);

8. Record and Playback

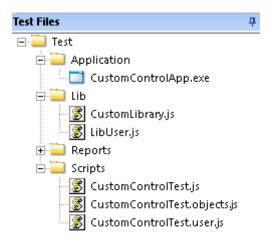
Now you are ready to record and playback a test. Just remember that in Select an Application to

Library	Description
🔽 User	Default user-defined library
🗖 Console	Console Application
🗹 Generic	Generic library contains basic definitions for most comm
MSOffice	Microsoft Office with Accessiility

Record dialog you need to uncheck Auto library and select User and Generic libraries.

9. CustomControlTest

This tutorial is complemented by a ready test CustomControlTest which you can examine and run. Open CustomControlTest in Rapise and place contents of CustomLibrary file into LibUser.js file (C: \Program Files\Inflectra\Rapise\Engine\Lib\LibUser\LibUser.js). LibUser.js is added to CustomControlTest, so you can populate it with CustomLibrary code right in Rapise.



Tip: It is possible to launch CustomControlApp right from Rapise, just double click on CustomControlApp.exe in the project tree.

10. Wrap-up: Implementation Sequence

Full support for a custom object requires support for Record, Learn and Playback. Let's go over created library and specify the purpose of each component in it.

- Matcher Rule:- it is used to recognize the object inside an application, required for Record, Learn and Playback.
- Events in Behavior Patterns: handling events is required for Record.
- Actions in Behavior Patterns: actions are used to examine or change state of the control, required for Playback.
- Custom Accessibility Event Handler: required for Record if some important events are not processed by Rapise engine as needed.

2.8 Glossary

This glossary presents a list of terms and their definitions as they are used in this guide.

API - Application Programming Interface

AUT - Application Under Test

DOM - Document Object Model

GUI - Graphical User Interface

GWT - Google Web Toolkit

IDE - Integrated Development Environment

JSON - JavaScript Object Notation

REST - REpresentation State Transfer

SOAP - Simple Object Access Protocol

UI - User Interface

XML - eXtensible Markup Language

YUI - Yahoo! User Interface (library)

Index

- A -

About this Guide 7 203 Accessing Functions Action 165 Add File 417 Add Web Service 320 Analog Recording 158 Assert 208 Automated Reporting 192

- B -

Breakpoints 219

- C -

Call Stack 419 Checkpoint 302 Create 208 Checkpoints 420 Code Completion 224 Code Folding 222 COM Testing 240 Command Line 190 Component Object Model 240 Content View 328 Control Execution 218 Create a new Recording Library 164 Create a New Test 428 Create File 417 Create New Test 322 Create Sub-Test 327 Cross-Browser Testing 451 Custom Library 164 Custom Recording Library 164 Custom Strings 241, 350

- D -

Data External 208 Data-Driven Testing 208 Debugger External 220 217 Internal Default Layout 435 **Defining Functions** 203 7,320 Dialogs **DLL** functions invoking 240 **DLL** objects creating and using 240 **DLL** Testing 240

- E -

Engine 212 Enter Filter Criteria 329 Entry Point 436 Errors View 331 Examples 10 Execution 189, 190 Execution flow 218 Exeuction Pause 219 External Data 208 External Debugger 220 External Files 206

- F -

Features 7, 150 Filter Group 419 Filter Report View 195 Find 332, 333, 334 Find and Replace Dialog 332 Find Results View 333 Find Text Dialog 334 Functions 203

- G -

Getting Started7, 8Global Variables206Guide Overview7

- | -

IDE 217

552

553

Include External Files206Including Functions203Internal Debugger217

- J -

Java Testing 528 Javascript IDE 217

- L -

Learning 154 Library 162

- M -

Menus 7, 320 Meta Data 241 Multiple Recordings 166 Multiple-Browser Testing 451

- N -

NameValue Collection Editor 350 Naming Conventions 203 New Group 417 New Test 322, 428 NUnit 231

- 0 -

Object Learning 154 **Object Locator** 191 **Object Manager** 180 **Object Properties** 359 **Object Recognition** 191 Object Spy 167.408 **Object Tree** 354 202 **Objects File** Open 427 427 Open a Test **Open File** 417 **Options Dialog** 355 **Output Verbosity** 221 Output View 358 **Override** Action 165 Overview 8

- P -

Pause Execution 219 Playback 189 Properties Dialog 359

- Q -

Qt Framework 527

- R -

Rapise Overview 8 Recording 152 Recording Activity Dialog 360 Recording Library 162 Regex 206 **Regression Testing** 302 206 **Regular Expressions** Replace 332, 365 Replace Text Dialog 365 Report 192, 373 Filtering 195, 329 Writing 194 **Report Viewer** 365 166 Re-record **REST Web Services** 244 **REST Definition Editor** 366 Tutorial: REST Web Services 65 Restore Default Layout 435 Restore Layout 435 Ribbon Debugger 378 Edit 377 Report 373 REST 381 Spreadsheet 376 Test 370

- S -

Sample Projects 10 Sample Tests 10 Screen Capture 395 Scripting 201 ScriptPath 396 Select an Application to Record Dialog 387 SeS Spy Dialog 408 Settings View 393 Simulated Object 161 Source Editor 400 Spira Dashboard 403 SpiraTest Integration 289 Spreadsheet Viewer 401 Spy 167, 408 Start Page 402 Sub-Test 327 Syntax Checking 223 Syntax Errors 422 Syntax Highlighting 222

- W -

Warning View 422 Watch View 422 Web Service Testing 242 REST Web Services 244

- T -

TAP 241 Test Anything Protocol 241 Test Entry Point 436 Test Files View 417 Test Function 202, 436 Test Script 202 TestFinish Function 202 Testing DLLs 240 TestInit Function 202 TestPrepare Function 202 Text Editor 400 The Test Script 202 218 Tooltips Tutorial 13

- U -

Unit Testing 227 User File 202

- V -

Variable View 419 variables query value 218 view values 419 Verbosity 221 Verify Object Properties Dialog 420 Views 7, 320