Rapise® | Quick Start Guide

Testing Qt Framework Applications with Rapise

Date: May 8th, 2017

Contents

Introduction

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

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, 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).

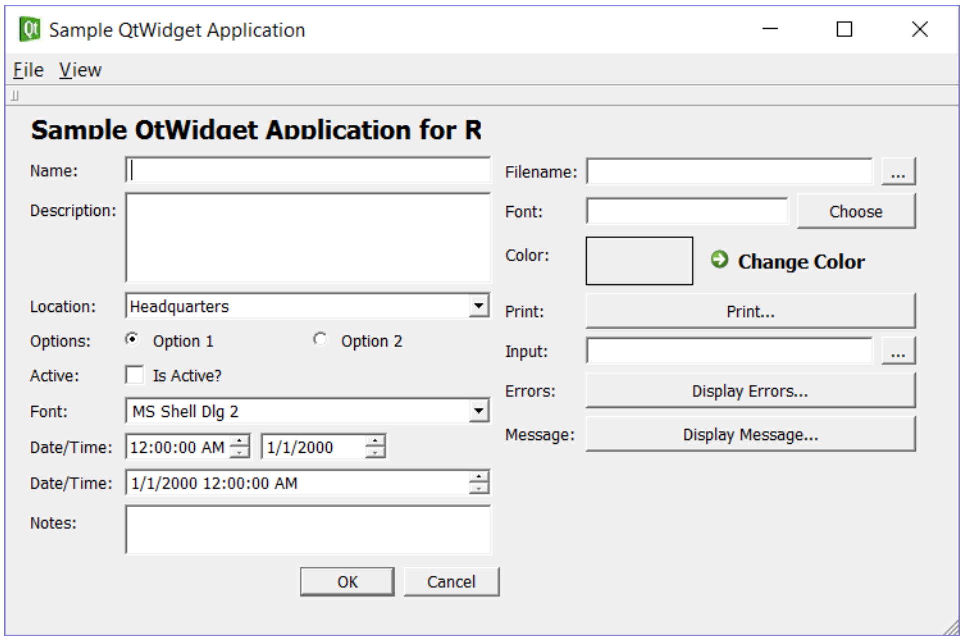
This tutorial illustrates the ability of Rapise to test such Qt applications using a sample application that already has the MSAA support added.

1. Testing the Sample Qt Application

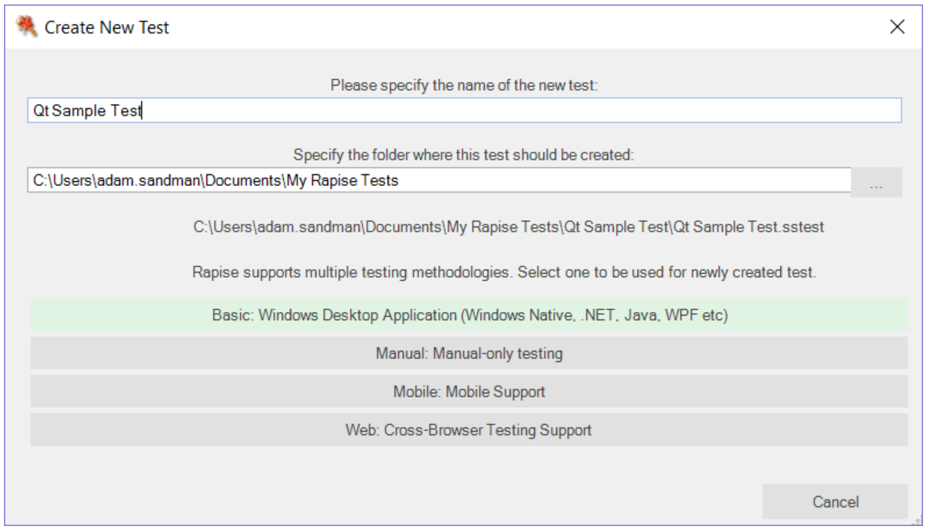
On the Start Page of Rapise, click on the **Fetch Samples** button to make sure you have all of the latest samples available.

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:



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:

Qt Sample Test

Specify the folder where this test should be created:

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

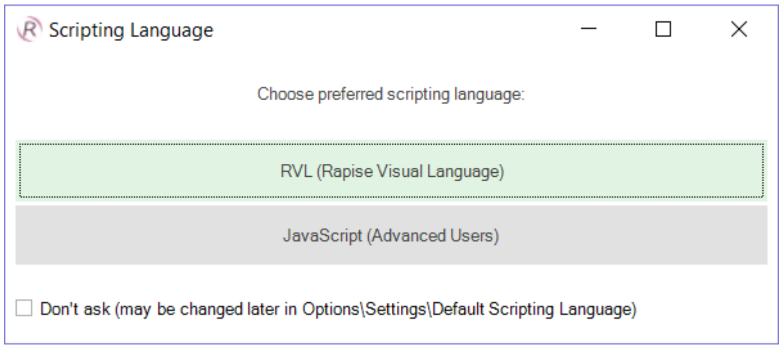
C:\Users\adam.sandman\Documents\My Rapise Tests\Qt Sample Test\Qt Sample Test.sstest

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

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:



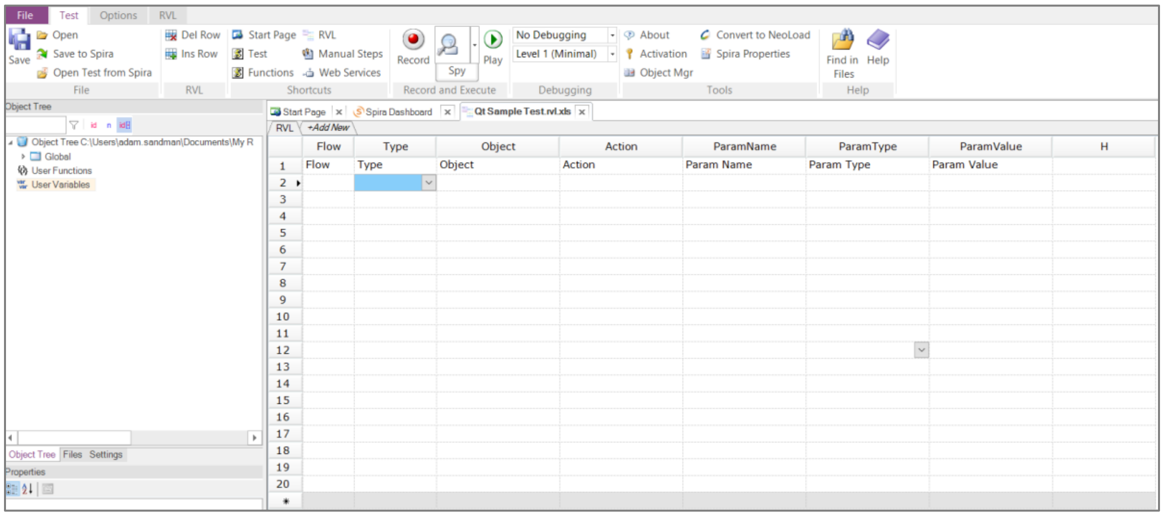
Choose preferred scripting language:

RVL (Rapise Visual Language)

JavaScript (Advanced Users)

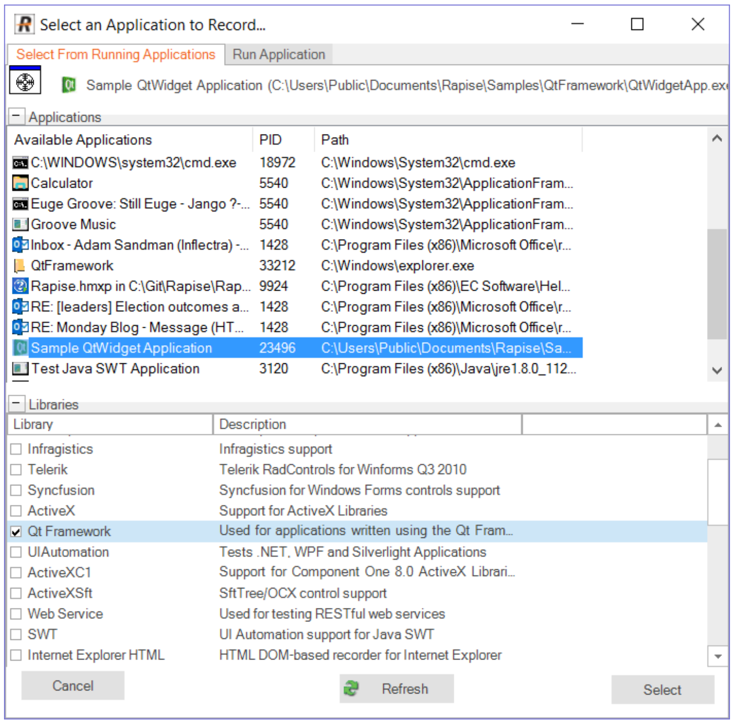
☐ Don't ask (may be changed later in Options\Settings\Default Scripting Language)

Once the test is created, you will see:



	Flow	Type	Object	Action	ParamName	ParamType	ParamValue	H
1	Flow	Type	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 an Application to Record...

Select From Running Applications | Run Application

Sample QtWidget Application (C:\Users\Public\Documents\Rapise\Samples\QtFramework\QtWidgetApp.ex

Applications

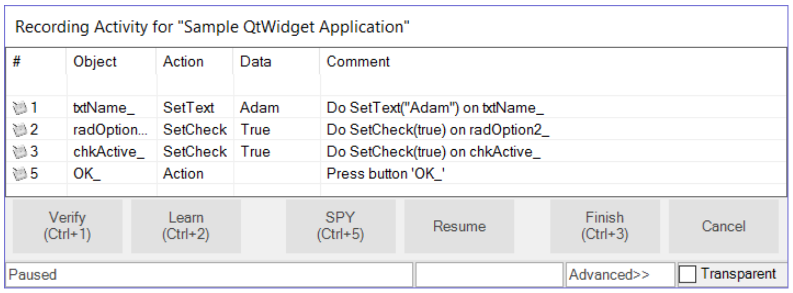
Available Applications	PID	Path
C:\WINDOWS\system32\cmd.exe	18972	C:\Windows\System32\cmd.exe
Calculator	5540	C:\Windows\System32\ApplicationFram...
Euge Groove: Still Euge - Jango ?-...	5540	C:\Windows\System32\ApplicationFram...
Groove Music	5540	C:\Windows\System32\ApplicationFram...
Inbox - Adam Sandman (Inflectra) -...	1428	C:\Program Files (x86)\Microsoft Office\r...
QtFramework	33212	C:\Windows\explorer.exe
Rapise.hmxp in C:\Git\Rapise\Rap...	9924	C:\Program Files (x86)\EC Software\Hel...
RE: [leaders] Election outcomes a...	1428	C:\Program Files (x86)\Microsoft Office\r...
RE: Monday Blog - Message (HT...	1428	C:\Program Files (x86)\Microsoft Office\r...
Sample QtWidget Application	23496	C:\Users\Public\Documents\Rapise\Sa...
Test Java SWT Application	3120	C:\Program Files (x86)\Java\jre1.8.0_112...

Libraries

Library	Description
☐ Infragistics	Infragistics support
☐ Telerik	Telerik RadControls for Winforms Q3 2010
☐ Syncfusion	Syncfusion for Windows Forms controls support
☐ ActiveX	Support for ActiveX Libraries
☑ Qt Framework	Used for applications written using the Qt Fram...
☐ UIAutomation	Tests .NET, WPF and Silverlight Applications
☐ ActiveXC1	Support for Component One 8.0 ActiveX Librari...
☐ ActiveXSft	SftTree/OCX control support
☐ Web Service	Used for testing RESTful web services
☐ SWT	UI Automation support for Java SWT
☐ Internet Explorer HTML	HTML DOM-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**.

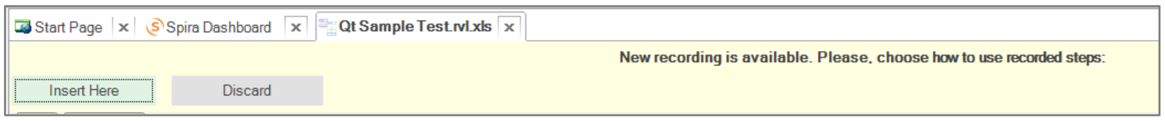
Now in the sample application click on some of the Qt controls. Rapise will record the actions:



Recording Activity for "Sample QtWidget Application"

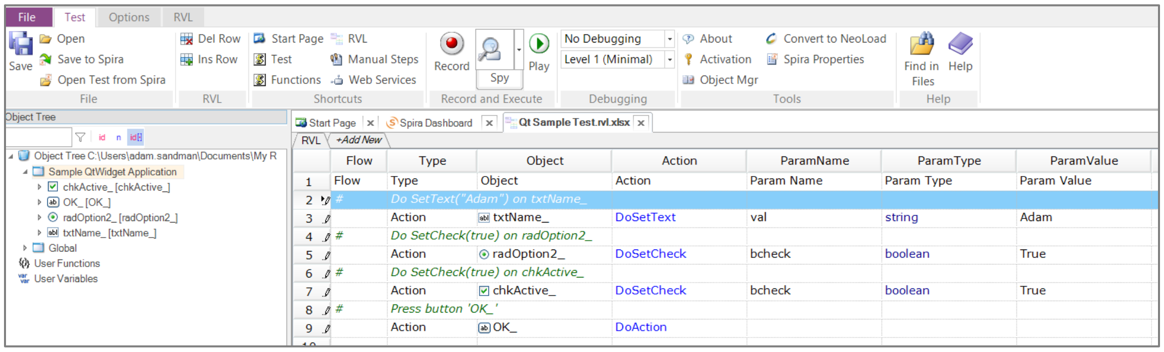
#	Object	Action	Data	Comment
1	txtName_	SetText	Adam	Do SetText("Adam") on txtName_
2	radOption...	SetCheck	True	Do SetCheck(true) on radOption2_
3	chkActive_	SetCheck	True	Do SetCheck(true) on chkActive_
5	OK_	Action		Press button 'OK_'

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



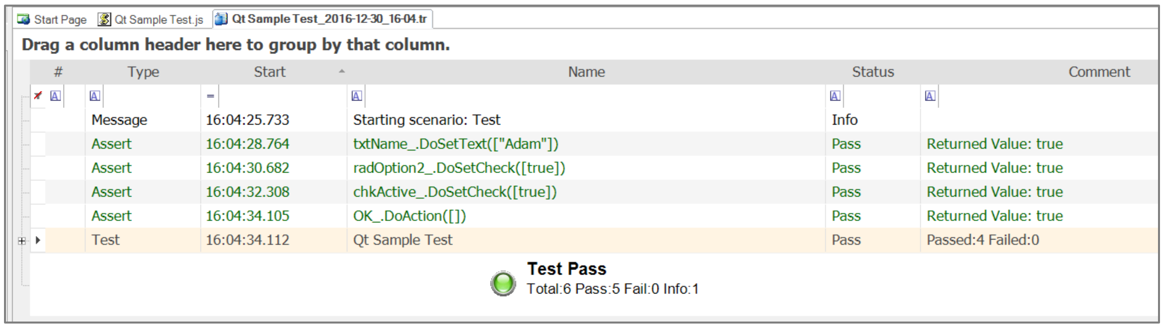
New recording is available. Please, choose how to use recorded steps:

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



	Flow	Type	Object	Action	ParamName	ParamType	ParamValue
1	Flow	Type	Object	Action	Param Name	Param Type	Param Value
2	#	Do SetText("Adam") on txtName_					
3		Action	txtName_	DoSetText	val	string	Adam
4	#	Do SetCheck(true) on radOption2_					
5		Action	radOption2_	DoSetCheck	bcheck	boolean	True
6	#	Do SetCheck(true) on chkActive_					
7		Action	chkActive_	DoSetCheck	bcheck	boolean	True
8	#	Press button 'OK_'					
9		Action	OK_	DoAction			

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



Drag a column header here to group by that column.

#	Type	Start	Name	Status	Comment
	Message	16:04:25.733	Starting scenario: Test	Info	
	Assert	16:04:28.764	txtName_.DoSetText(["Adam"])	Pass	Returned Value: true
	Assert	16:04:30.682	radOption2_.DoSetCheck([true])	Pass	Returned Value: true
	Assert	16:04:32.308	chkActive_.DoSetCheck([true])	Pass	Returned Value: true
	Assert	16:04:34.105	OK_.DoAction([])	Pass	Returned Value: true
	Test	16:04:34.112	Qt Sample Test	Pass	Passed:4 Failed:0

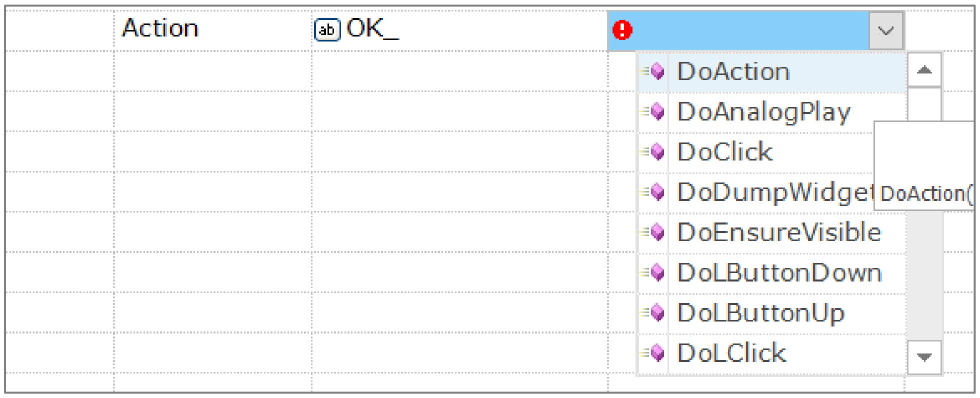
Test Pass
Total:6 Pass:5 Fail:0 Info:1

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 = OK_
- Action = DoAction

This process is illustrated below:

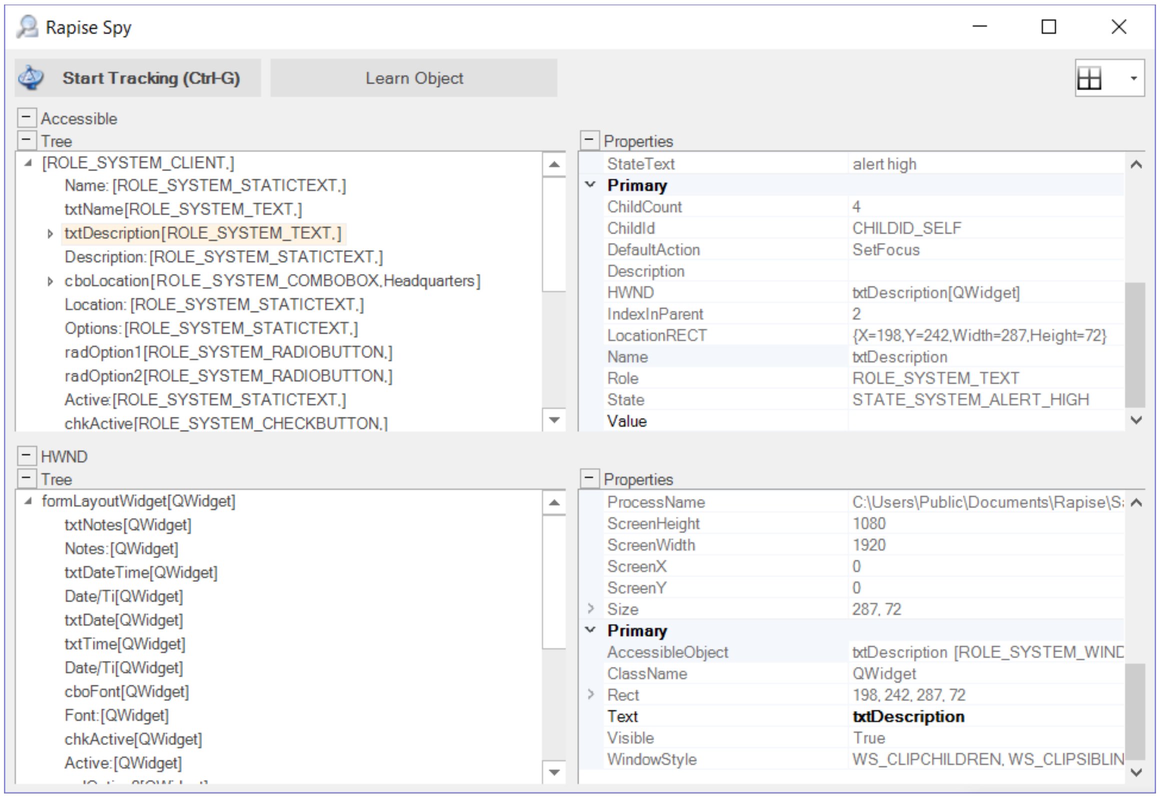


1.1. Using the Object Spy

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 Accessible Spy:



Accessible

Tree

```
◢ [ROLE_SYSTEM_CLIENT,]
    Name: [ROLE_SYSTEM_STATICTEXT,]
    txtName[ROLE_SYSTEM_TEXT,]
  ▸ txtDescription[ROLE_SYSTEM_TEXT,]
    Description: [ROLE_SYSTEM_STATICTEXT,]
  ▸ cboLocation[ROLE_SYSTEM_COMBOBOX,Headquarters]
    Location: [ROLE_SYSTEM_STATICTEXT,]
    Options: [ROLE_SYSTEM_STATICTEXT,]
    radOption1[ROLE_SYSTEM_RADIOBUTTON,]
    radOption2[ROLE_SYSTEM_RADIOBUTTON,]
    Active:[ROLE_SYSTEM_STATICTEXT,]
    chkActive[ROLE_SYSTEM_CHECKBUTTON,]
```

Properties

StateText	alert high
Primary	
ChildCount	4
ChildId	CHILDID_SELF
DefaultAction	SetFocus
Description	
HWND	txtDescription[QWidget]
IndexInParent	2
LocationRECT	{X=198,Y=242,Width=287,Height=72}
Name	txtDescription
Role	ROLE_SYSTEM_TEXT
State	STATE_SYSTEM_ALERT_HIGH
Value	

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]
    Active:[QWidget]
```

Properties

ProcessName	C:\Users\Public\Documents\Rapise\S:
ScreenHeight	1080
ScreenWidth	1920
ScreenX	0
ScreenY	0
Size	287, 72
Primary	
AccessibleObject	txtDescription [ROLE_SYSTEM_WINE
ClassName	QWidget
Rect	198, 242, 287, 72
Text	**txtDescription**
Visible	True
WindowStyle	WS_CLIPCHILDREN, WS_CLIPSIBLIN

You can then use the Accessible Spy 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.

Once that object has been learned, you can select it as one of the objects in the test grid:

2. Testing Your Qt Applications

Now that you have tested our sample Qt Application with Rapise, the next stage is test one of your 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).

2.1. Loading the 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
   ```

Legal Notices

Please send comments and questions to:

Technical Publications

Inflectra Corporation

8121 Georgia Ave, Suite 504

Silver Spring, MD 20910-4957

U.S.A.

support@inflectra.com