Rapise® | Quick Start Guide

Testing Java Applications with Rapise

Date: May 9th, 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 Java using all of the different front-end technologies – AWT, Swing and SWT.

In this guide, you will learn how to record and execute a Rapise script against Java applications.

We will show you how to test the following three different types of Java application:

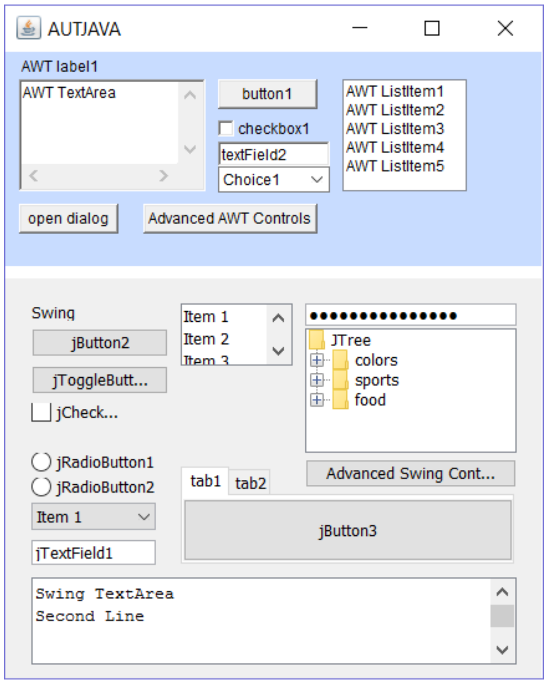
- Java AWT Apps
- Java Swing Apps
- Java SWT Apps

1. Testing the Sample AWT/Swing Application

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

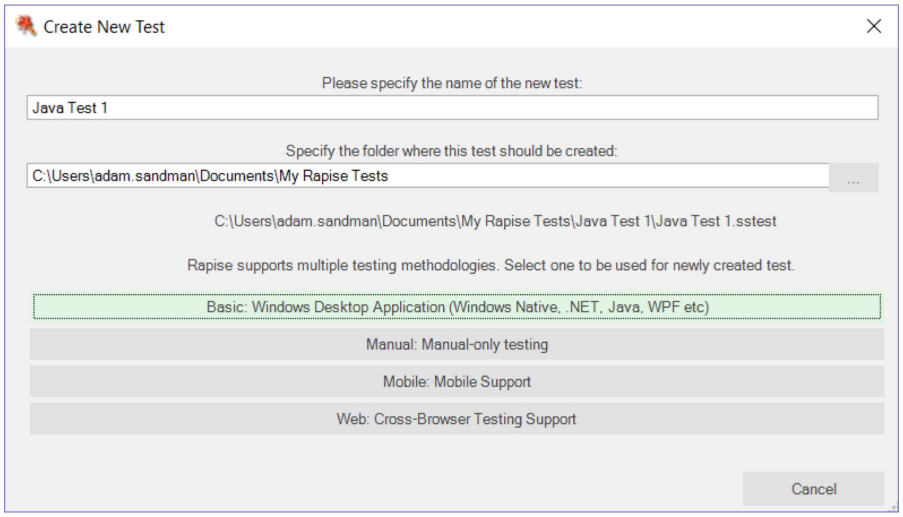
Then go to `C:\Users\Public\Documents\Rapise\Samples\Java\AUTJAVA` and right-click on the x86run.cmd file and choose **Run as Administrator**.

If you have Java configured correctly, you will see:



If the application doesn't start correctly, make sure you have Java SE and the Rapise Java Bridge installed and the JAVA_HOME environment variable correctly set to your Java Runtime (JRE). For more details on this, please refer to **Appendix A – Preparing Rapise for Java SWT/Swing.**

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 test.

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

Manual: Manual-only testing

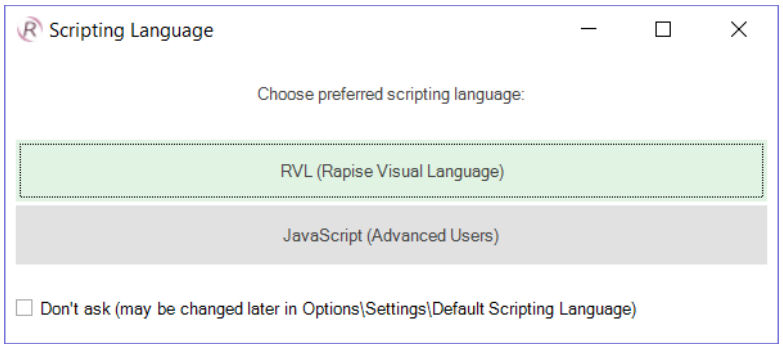
Mobile: Mobile Support

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:



Scripting Language

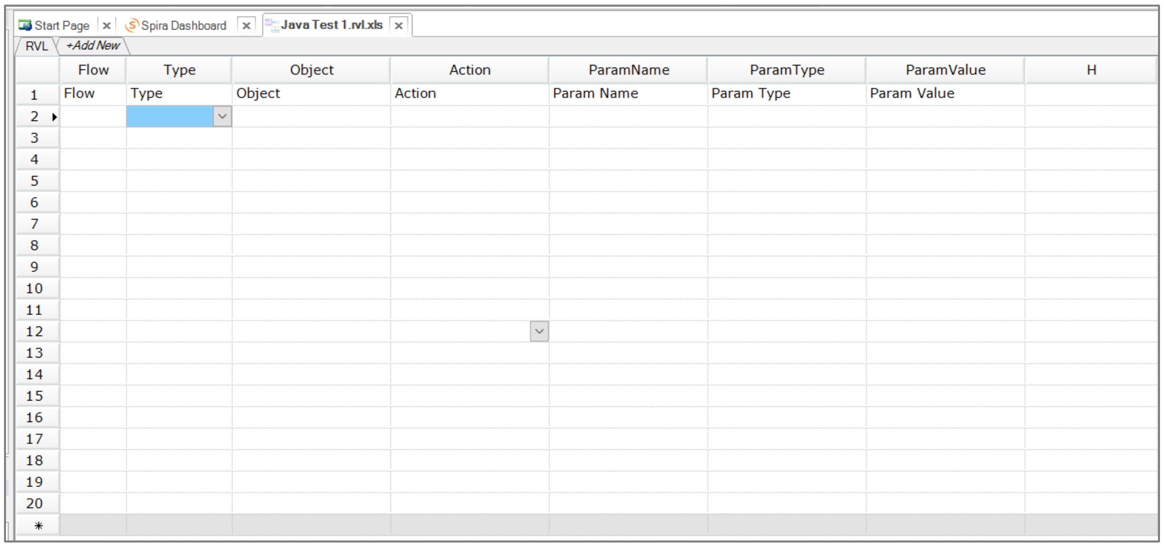
Choose preferred scripting language:

RVL (Rapise Visual Language)

JavaScript (Advanced Users)

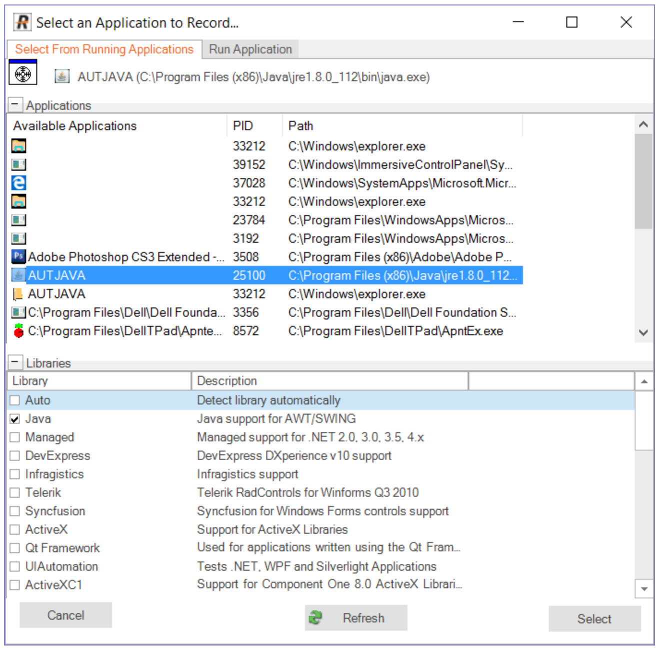
Don't ask (may be changed later in Options\Settings\Default Scripting Language)

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:

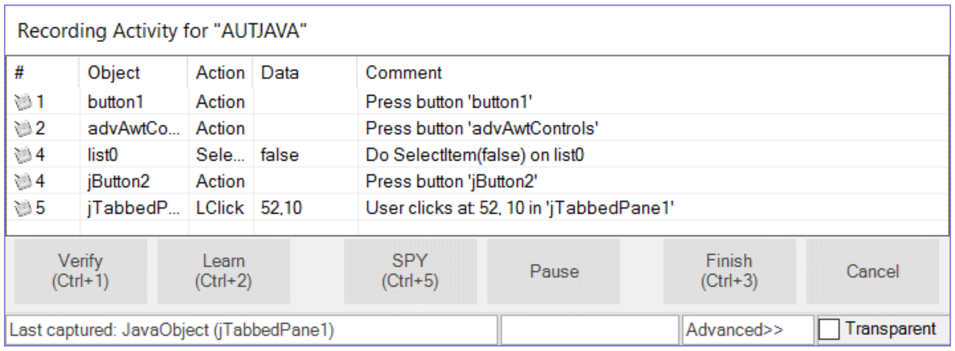

Select an Application to Record...
Select From Running Applications
Run Application
AUTJAVA (C:\Program Files (x86)\Java\jre1.8.0_112\bin\java.exe)
Applications

Available Applications	PID	Path
	33212	C:\Windows\explorer.exe
	39152	C:\Windows\ImmersiveControlPanel\Sy...
	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 CS3 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
C:\Program Files\Dell\Dell Founda...	3356	C:\Program Files\Dell\Dell Foundation S...
C:\Program Files\DellTPad\Apnte...	8572	C:\Program Files\DellTPad\ApntEx.exe

Library	Description
☐ Auto	Detect library automatically
☑ Java	Java support for AWT/SWING
☐ Managed	Managed support for .NET 2.0, 3.0, 3.5, 4.x
☐ DevExpress	DevExpress DXperience v10 support
☐ 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...

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

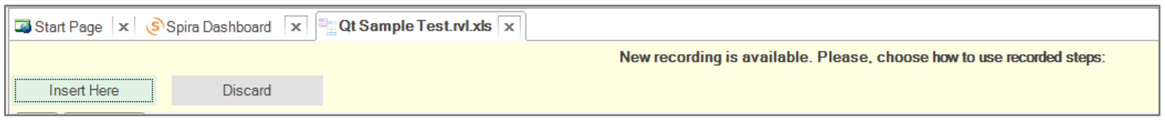
Now in the sample application click on some of the AWT and/or Swing controls. Rapise will record the actions:



#	Object	Action	Data	Comment
1	button1	Action		Press button 'button1'
2	advAwtCo...	Action		Press button 'advAwtControls'
4	list0	Sele...	false	Do SelectItem(false) on list0
4	jButton2	Action		Press button 'jButton2'
5	jTabbedP...	LClick	52,10	User clicks at 52, 10 in 'jTabbedPane1'

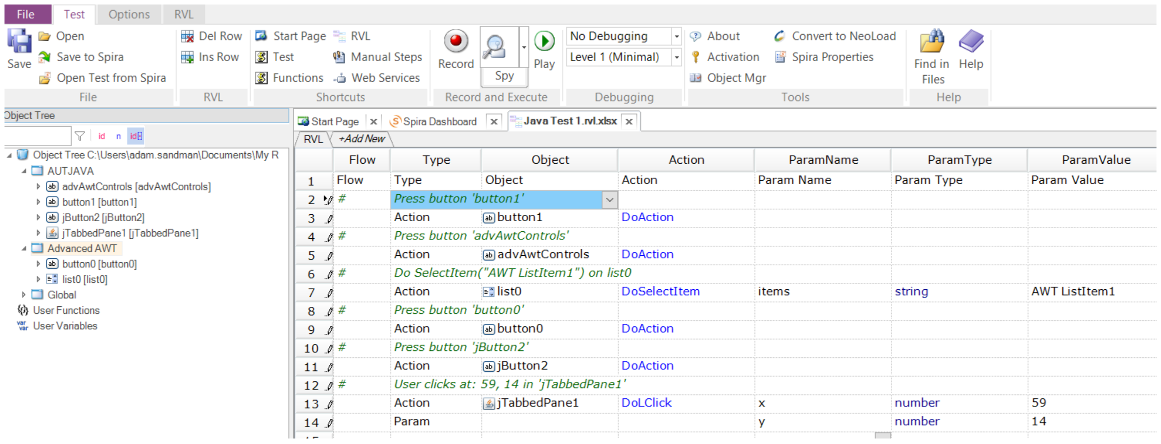
Last captured: JavaObject (jTabbedPane1)

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



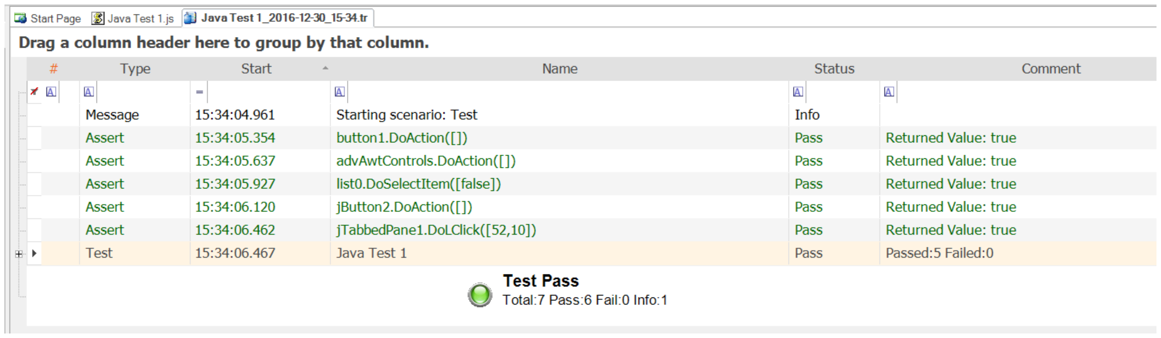
New recording is available. Please, choose how to use recorded steps:

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	#	Press button 'button1'					
3		Action	button1	DoAction			
4	#	Press button 'advAwtControls'					
5		Action	advAwtControls	DoAction			
6	#	Do SelectItem("AWT ListItem1") on list0					
7		Action	list0	DoSelectItem	items	string	AWT ListItem1
8	#	Press button 'button0'					
9		Action	button0	DoAction			
10	#	Press button 'jButton2'					
11		Action	jButton2	DoAction			
12	#	User clicks at: 59, 14 in 'jTabbedPane1'					
13		Action	jTabbedPane1	DoLClick	x	number	59
14		Param			y	number	14

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


Start Page
Java Test 1.js
Java Test 1_2016-12-30_15-34.tr
Drag a column header here to group by that column.

#	Type	Start	Name	Status	Comment
	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

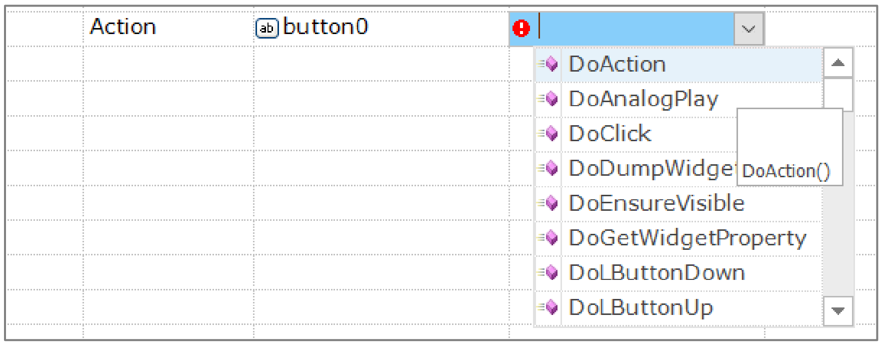
Test Pass
Total:7 Pass:6 Fail:0 Info:1

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 = button0
- Action = DoAction

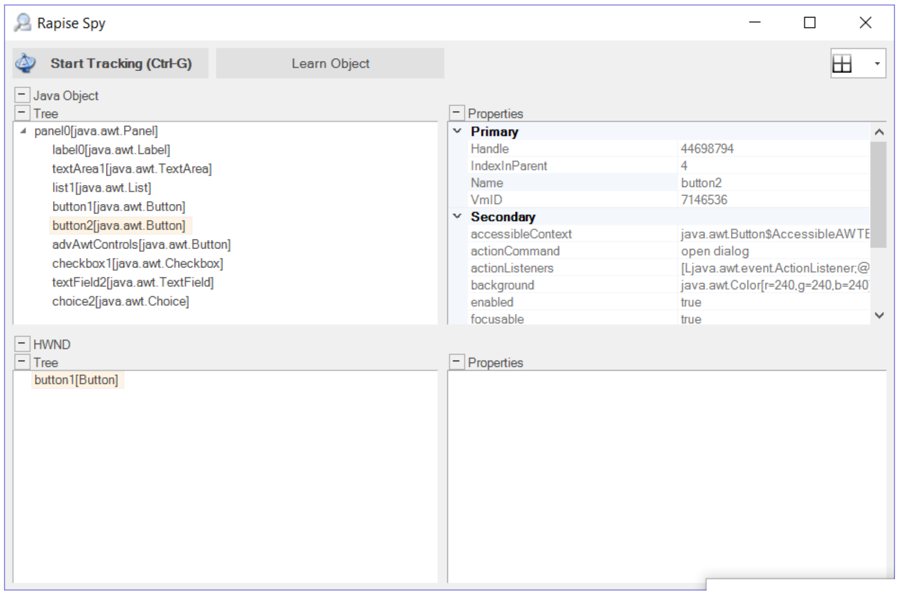
This process is illustrated below:



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:



- 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	44698794
IndexInParent	4
Name	button2
VmID	7146536
Secondary	
accessibleContext	java.awt.Button$AccessibleAWTE
actionCommand	open dialog
actionListeners	[Ljava.awt.event.ActionListener;@
background	java.awt.Color[r=240,g=240,b=240
enabled	true
focusable	true

button1[Button]

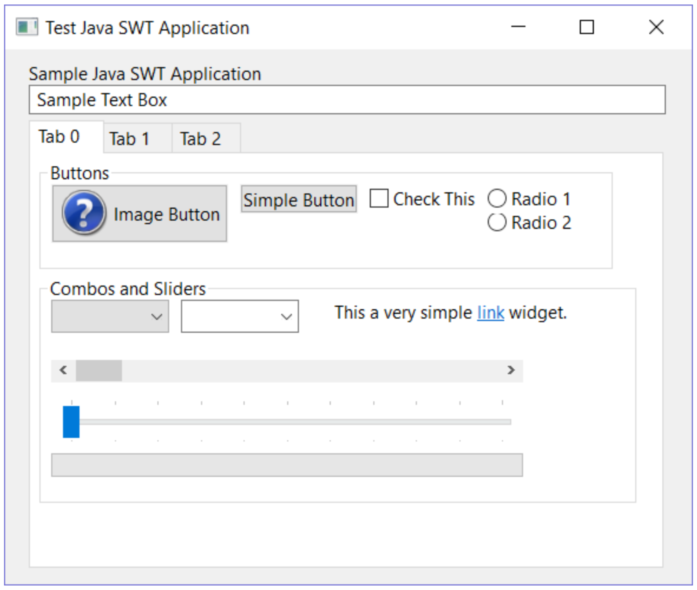
You can then use the Java Spy to track and find objects in the application hierarchy. You can navigate to parent objects by right-clicking on them and choosing **Parent**. Once you have found the desired object, click on the **Learn Object** in the Spy toolbar and Rapise will add the object in the Spy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

2. Testing the Sample SWT Application

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

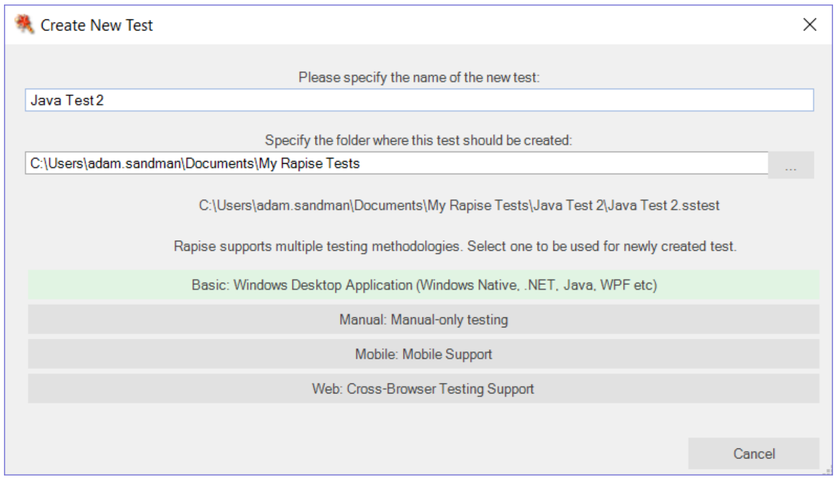
Then go to `C:\Users\Public\Documents\Rapise\Samples\JavaSWT\AUTJavaSWT` and double-click on the `JavaSWTAUT.bat` file to start the sample application:

If you have Java configured correctly, you will see:



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: **Appendix B - Preparing Rapise for Java SWT**

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



Please specify the name of the new test:

Java Test 2

Specify the folder where this test should be created:

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

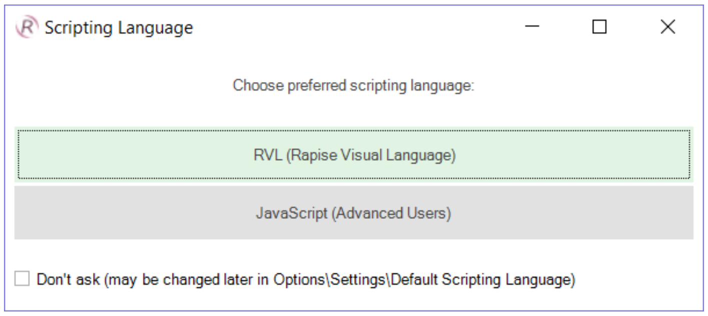
C:\Users\adam.sandman\Documents\My Rapise Tests\Java Test 2\Java Test 2.sstest

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 "**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:



Scripting Language

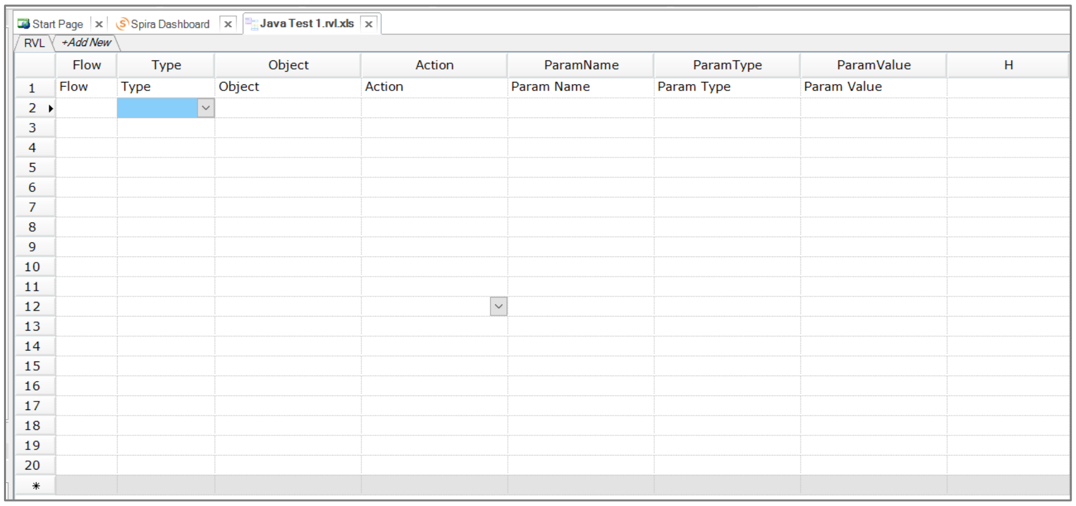
Choose preferred scripting language:

RVL (Rapise Visual Language)

JavaScript (Advanced Users)

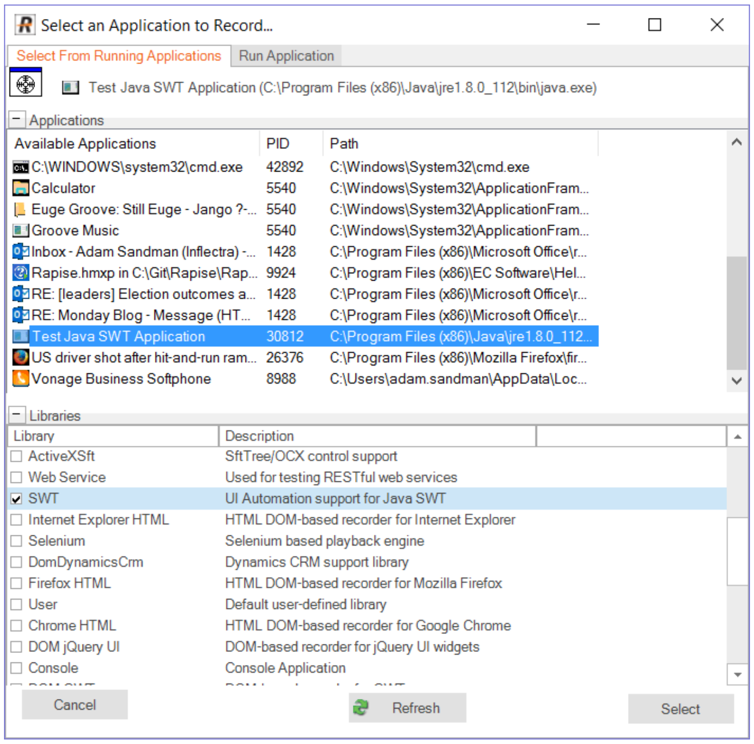
☐ Don't ask (may be changed later in Options\Settings\Default Scripting Language)

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

Test Java SWT Application (C:\Program Files (x86)\Java\jre1.8.0_112\bin\java.exe)

Applications

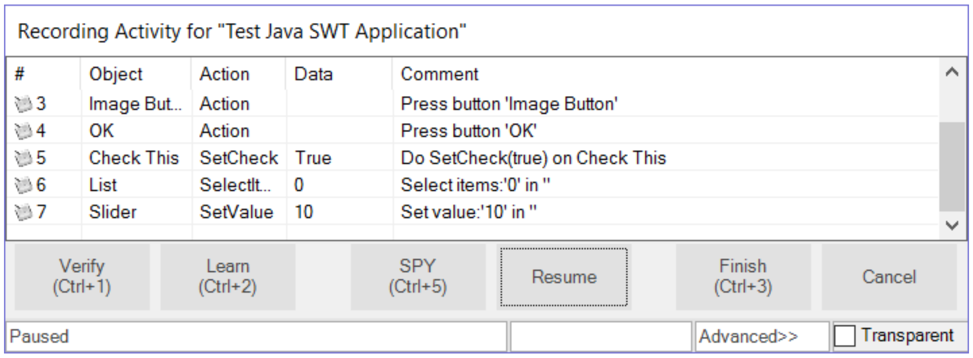
Available Applications	PID	Path
C:\WINDOWS\system32\cmd.exe	42892	C:\Windows\System32\cmd.exe
Calculator	5540	C:\Windows\System32\ApplicationFram...
Euge Groove: Still Euge - Jango ?-...	5540	C:\Windows\System32\ApplicationFram...
Groove Music	5540	C:\Windows\System32\ApplicationFram...
Inbox - Adam Sandman (Inflectra) -...	1428	C:\Program Files (x86)\Microsoft Office\r...
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...
Test Java SWT Application	30812	C:\Program Files (x86)\Java\jre1.8.0_112...
US driver shot after hit-and-run ram...	26376	C:\Program Files (x86)\Mozilla Firefox\fir...
Vonage Business Softphone	8988	C:\Users\adam.sandman\AppData\Loc...

Libraries

Library	Description
☐ 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
☐ Selenium	Selenium based playback engine
☐ DomDynamicsCrm	Dynamics CRM support library
☐ Firefox HTML	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	Console Application

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

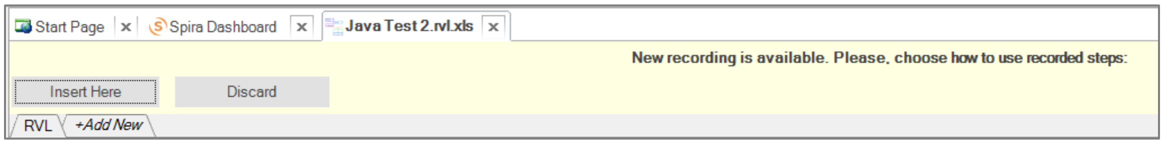
Now in the sample application click on some of the SWT controls. Rapise will record the actions:



Recording Activity for "Test Java SWT Application"

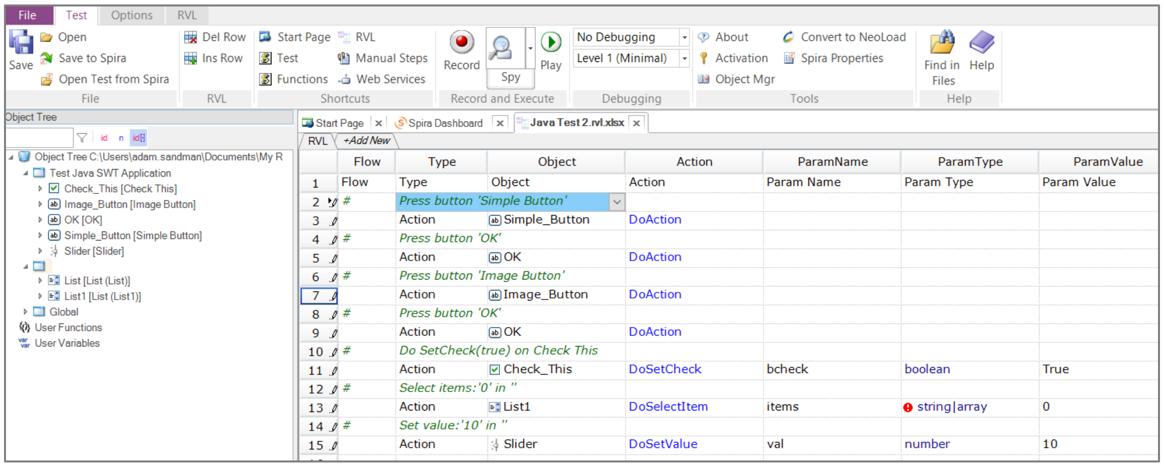
#	Object	Action	Data	Comment
3	Image But...	Action		Press button 'Image Button'
4	OK	Action		Press button 'OK'
5	Check This	SetCheck	True	Do SetCheck(true) on Check This
6	List	SelectIt...	0	Select items:'0' in ''
7	Slider	SetValue	10	Set value:'10' in ''

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



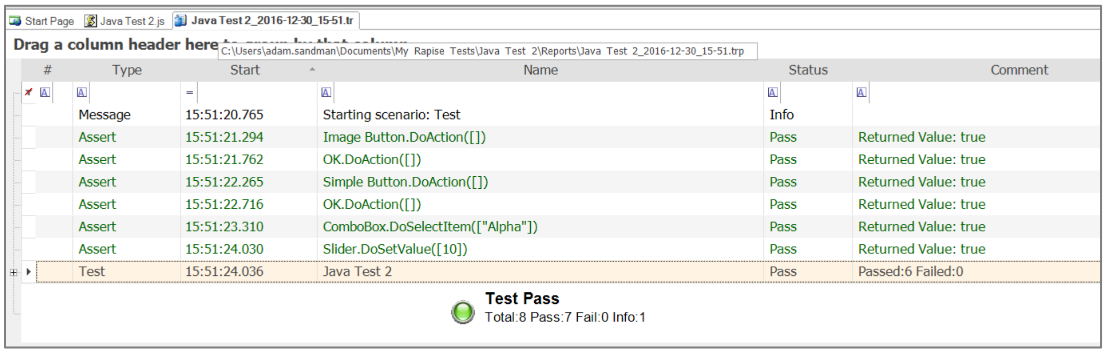
New recording is available. Please, choose how to use recorded steps:

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	#	Press button 'Simple Button'					
3		Action	Simple_Button	DoAction			
4	#	Press button 'OK'					
5		Action	OK	DoAction			
6	#	Press button 'Image Button'					
7		Action	Image_Button	DoAction			
8	#	Press button 'OK'					
9		Action	OK	DoAction			
10	#	Do SetCheck(true) on Check This					
11		Action	Check_This	DoSetCheck	bcheck	boolean	True
12	#	Select items:'0' in ''					
13		Action	List1	DoSelectItem	items	string|array	0
14	#	Set value:'10' in ''					
15		Action	Slider	DoSetValue	val	number	10

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



Drag a column header here

C:\Users\adam.sandman\Documents\My Rapise Tests\Java Test 2\Reports\Java Test 2_2016-12-30_15-51.trp

#	Type	Start	Name	Status	Comment
	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

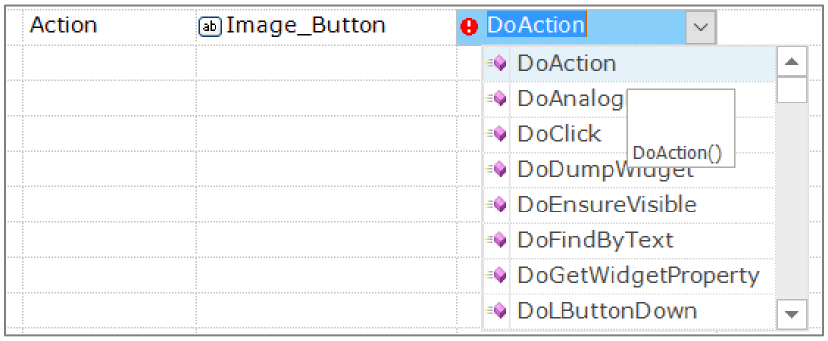
Test Pass
Total:8 Pass:7 Fail:0 Info:1

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 = Image_Button
- Action = DoAction

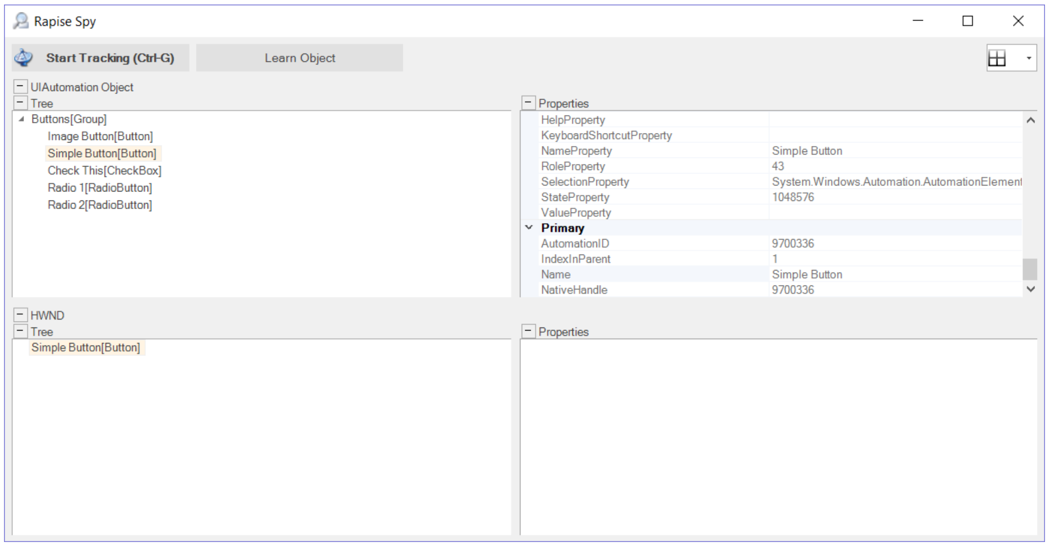
This process is illustrated below:



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 UIAutomation Spy:



Buttons[Group]
Image Button[Button]
Simple Button[Button]
Check This[CheckBox]
Radio 1[RadioButton]
Radio 2[RadioButton]

HelpProperty	
KeyboardShortcutProperty	
NameProperty	Simple Button
RoleProperty	43
SelectionProperty	System.Windows.Automation.AutomationElement
StateProperty	1048576
ValueProperty	
Primary	
AutomationID	9700336
IndexInParent	1
Name	Simple Button
NativeHandle	9700336

Simple Button[Button]

You can then use the UIAutomation 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.

Appendix A – Preparing Rapise for Java AWT/Swing

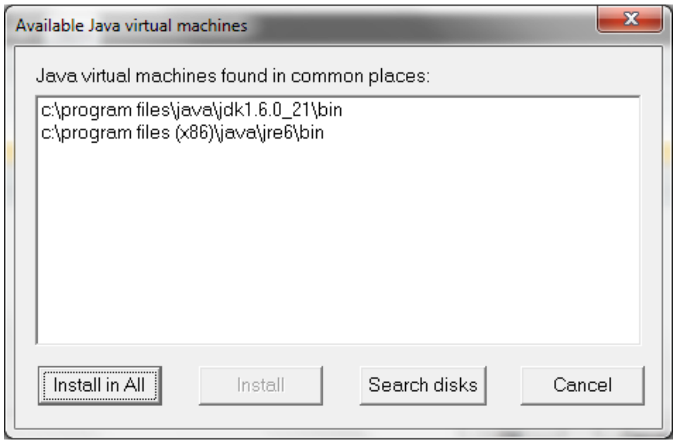
Rapise supports the testing of Java applications using either the Abstract Window Toolkit (AWT) or Swing graphic user interface toolkits. For maximum flexibility, Rapise can connect to your choice of JVM.

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:



Available Java virtual machines

Java virtual machines found in common places:

c:\program files\java\jdk1.6.0_21\bin
c:\program files (x86)\java\jre6\bin

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`):

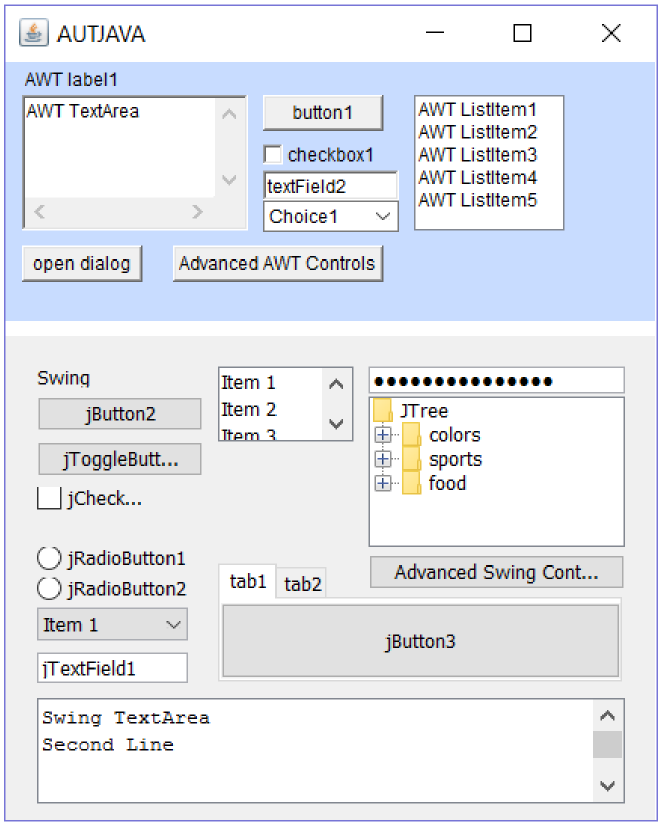
- lib\accessibility.properties
- lib\ext\jaccess.jar
- lib\ext\smartestudio-bridge.jar

If you don't see **all three of these files** then it means the bridge was not installed correctly.

Troubleshooting the Configuration

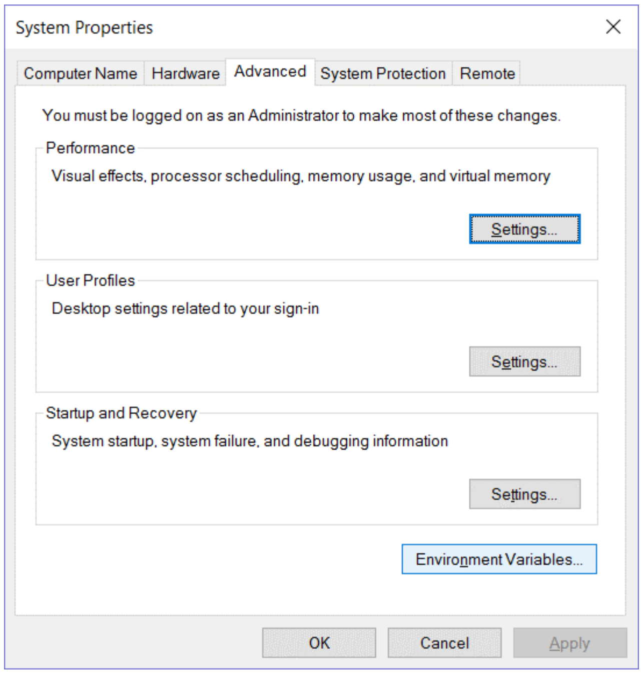
To help you make sure that your environment is correctly setup and also to help you try out Rapise, we have a sample application called AUTJava (AUT = Application Under Test) that can be found in the folder: C:\Users\Public\Documents\Rapise\Samples\Java\AUTJAVA

To run the application, right-click on the x86run.cmd file and choose **Run as Administrator**.



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**:



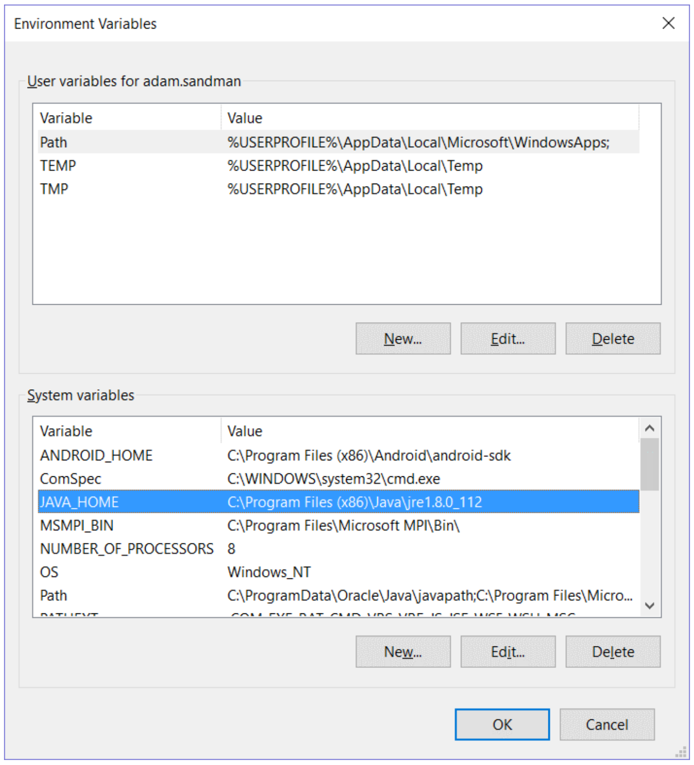
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

Click on the **Environment Variables** button:



Variable	Value
Path	%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

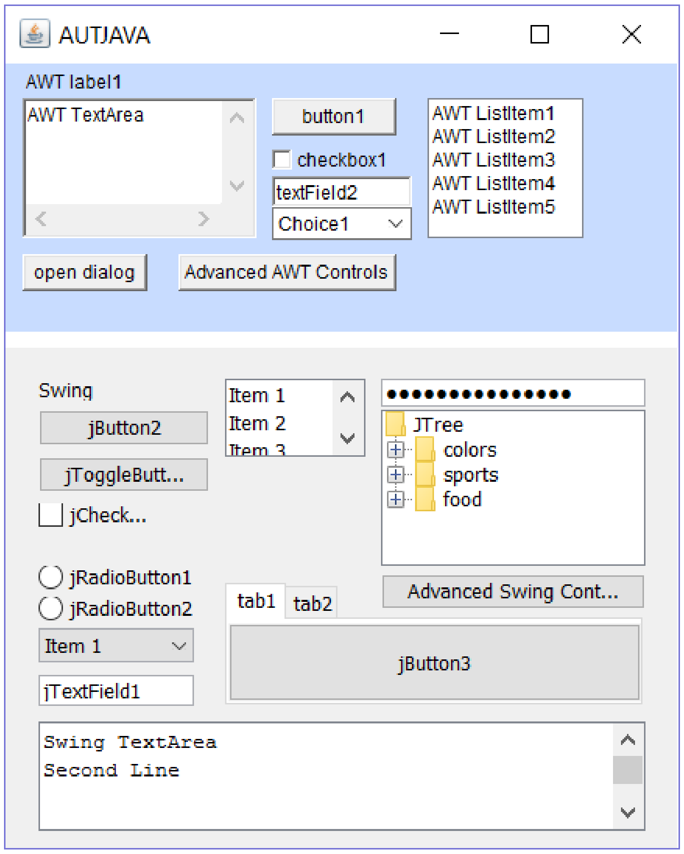
Variable	Value
ANDROID_HOME	C:\Program Files (x86)\Android\android-sdk
ComSpec	C:\WINDOWS\system32\cmd.exe
JAVA_HOME	C:\Program Files (x86)\Java\jre1.8.0_112
MSMPI_BIN	C:\Program Files\Microsoft MPI\Bin\
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\ProgramData\Oracle\Java\javapath;C:\Program Files\Micro...

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

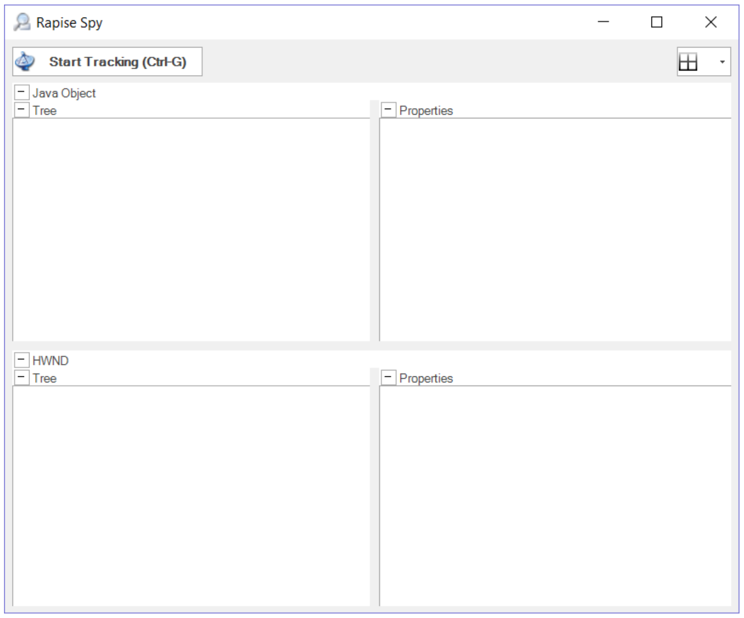
(you will need to match the location of your actual Java VM)

Now you should be able to launch the AUTJava sample application.



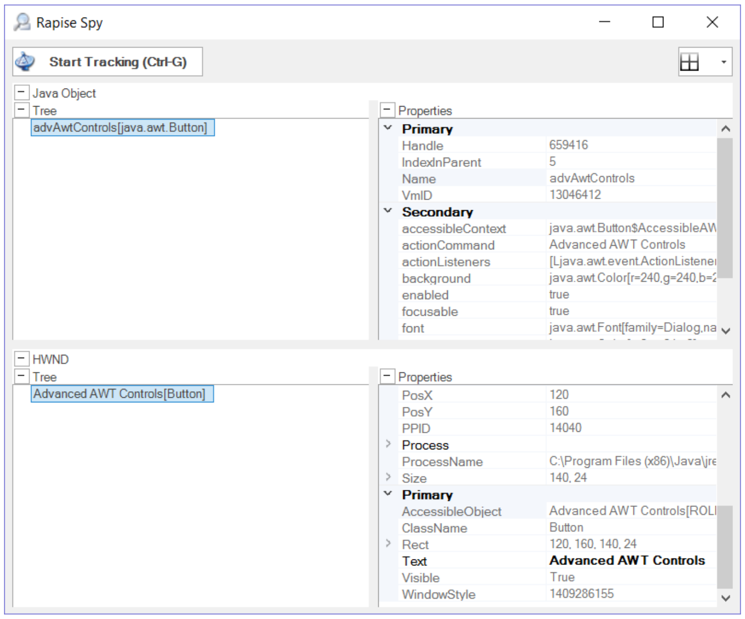
To verify that Rapise is configured correctly, click on the SPY menu in Rapise and choose **Java Spy**.

Then click on the main **Spy** icon and the Java Spy will start up:



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:



advAwtControls[java.awt.Button]

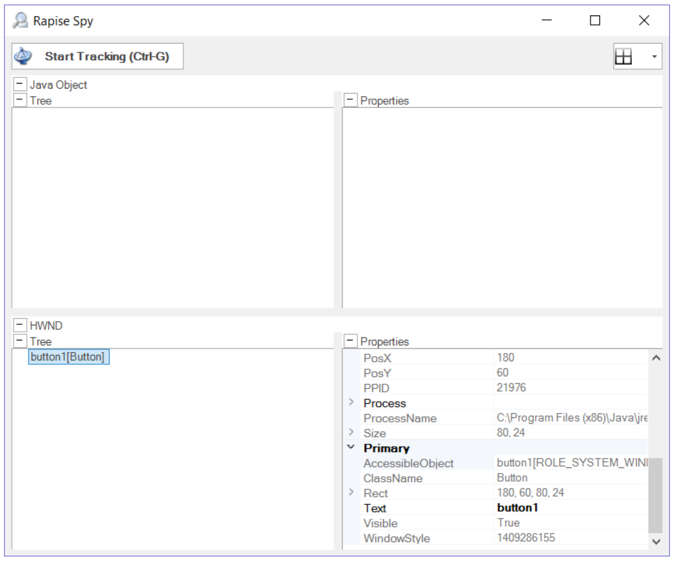
Primary	
Handle	659416
IndexInParent	5
Name	advAwtControls
VmID	13046412
Secondary	
accessibleContext	java.awt.Button$AccessibleAW
actionCommand	Advanced AWT Controls
actionListeners	[Ljava.awt.event.ActionListene
background	java.awt.Color[r=240,g=240,b=2
enabled	true
focusable	true
font	java.awt.Font[family=Dialog,na

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
WindowStyle	1409286155

Which shows that Rapise is able to see the AWT button (in this example) and its properties.

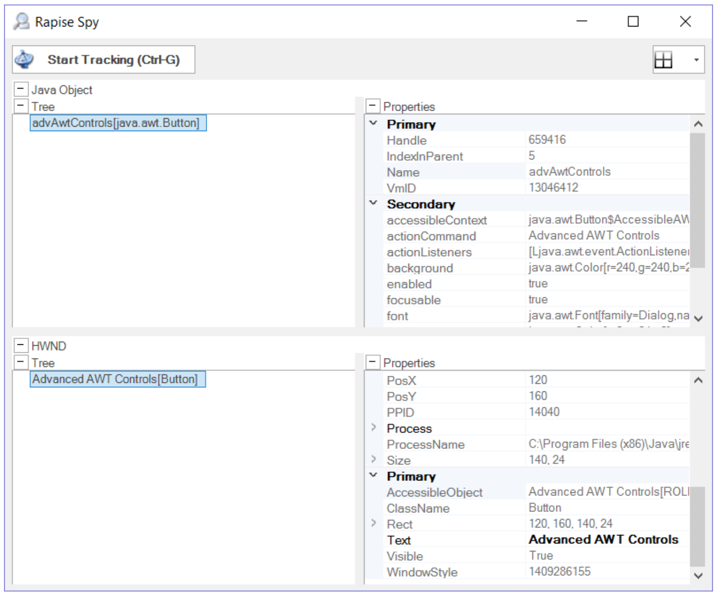
However if you see the following instead:



button1[Button]

PosX	180
PosY	60
PPID	21976
Process	
ProcessName	C:\Program Files (x86)\Java\jre
Size	80, 24
Primary	
AccessibleObject	button1[ROLE_SYSTEM_WINI
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:



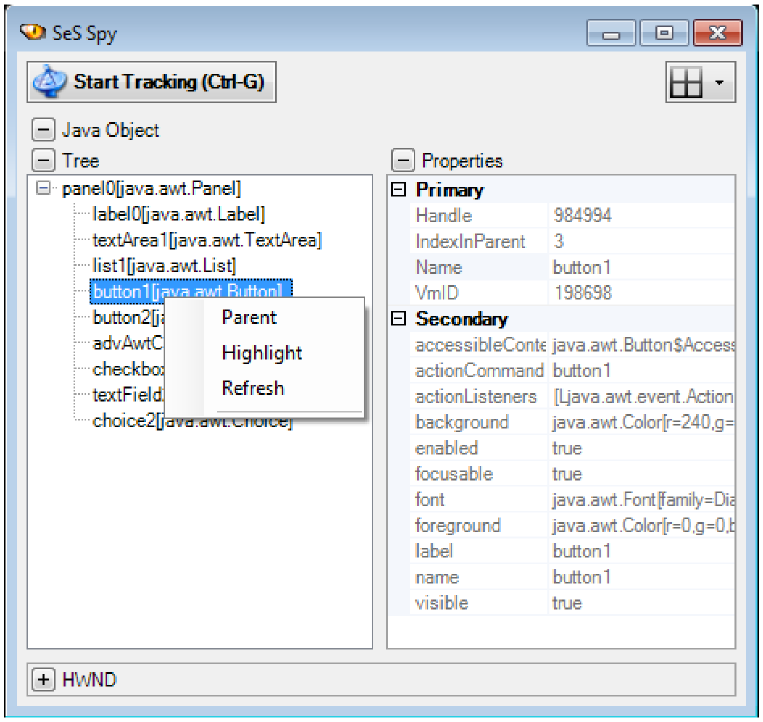
Primary	
Handle	659416
IndexInParent	5
Name	advAwtControls
VmID	13046412
Secondary	
accessibleContext	java.awt.Button$AccessibleAW
actionCommand	Advanced AWT Controls
actionListeners	[Ljava.awt.event.ActionListener
background	java.awt.Color[r=240,g=240,b=2
enabled	true
focusable	true
font	java.awt.Font[family=Dialog,na

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
WindowStyle	1409286155

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

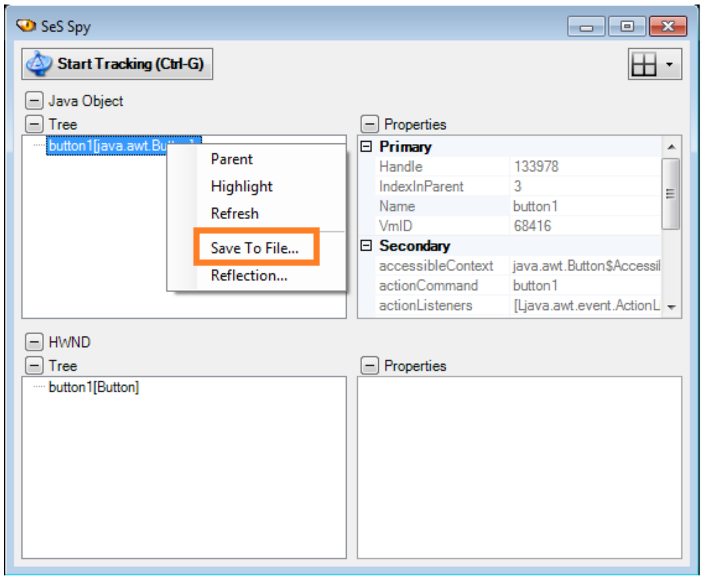
With Spy you can walk along the tree of Java objects in your application.

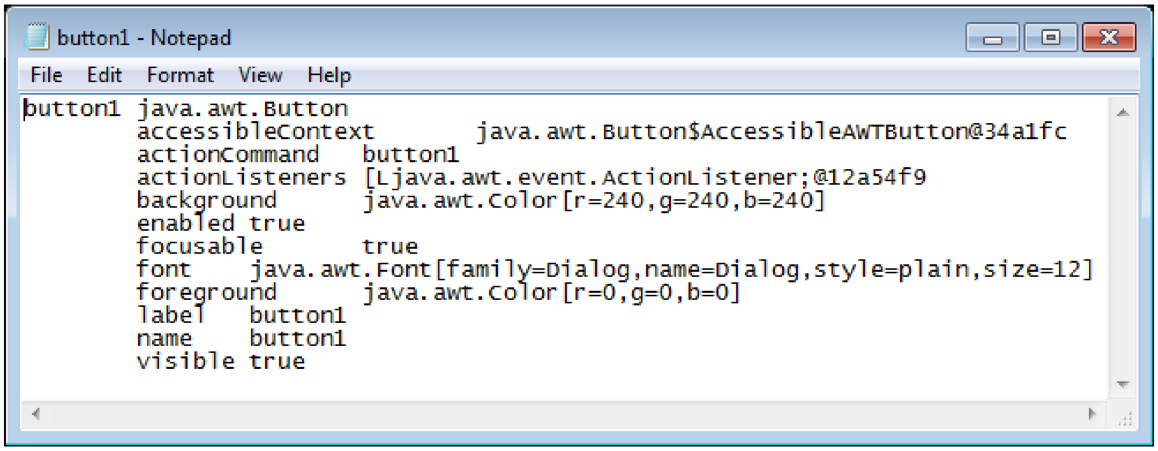


Primary	
Handle	984994
IndexInParent	3
Name	button1
VmID	198698
Secondary	
accessibleConte	java.awt.Button$Access
actionCommand	button1
actionListeners	[Ljava.awt.event.Action
background	java.awt.Color[r=240,g=
enabled	true
focusable	true
font	java.awt.Font[family=Dia
foreground	java.awt.Color[r=0,g=0,b
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.

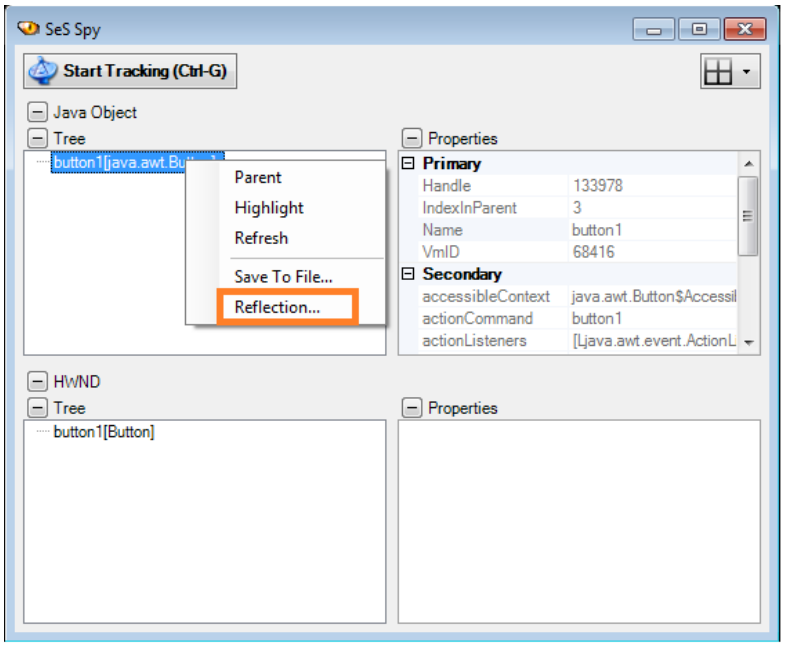




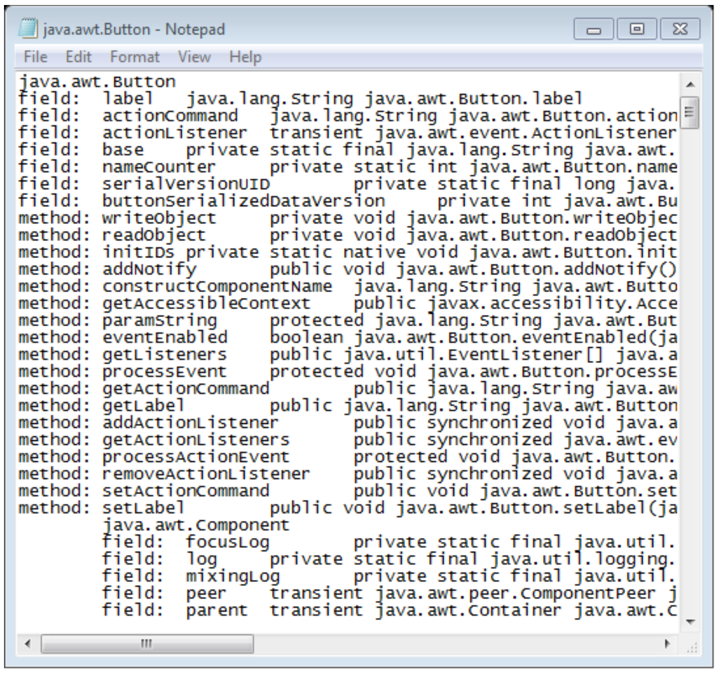
```
button1 java.awt.Button
        accessibleContext       java.awt.Button$AccessibleAWTButton@34a1fc
        actionCommand   button1
        actionListeners [Ljava.awt.event.ActionListener;@12a54f9
        background      java.awt.Color[r=240,g=240,b=240]
        enabled true
        focusable       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
```

Reflection Information

You can save reflection information for a java class used to implement a GUI control.



Primary	
Handle	133978
IndexInParent	3
Name	button1
VmID	68416
Secondary	
accessibleContext	java.awt.Button$Accessil
actionCommand	button1
actionListeners	[Ljava.awt.event.ActionL



```
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:  serialVersionUID        private static final long java.
field:  buttonSerializedDataVersion     private int java.awt.Bu
method: writeObject     private void java.awt.Button.writeObjec
method: readObject      private void java.awt.Button.readObject
method: initIDs private static native void java.awt.Button.init
method: addNotify       public void java.awt.Button.addNotify()
method: constructComponentName  java.lang.String java.awt.Butto
method: getAccessibleContext    public javax.accessibility.Acce
method: paramString     protected java.lang.String java.awt.But
method: eventEnabled    boolean java.awt.Button.eventEnabled(ja
method: getListeners    public java.util.EventListener[] java.a
method: processEvent    protected void java.awt.Button.processE
method: getActionCommand        public java.lang.String java.aw
method: getLabel        public java.lang.String java.awt.Button
method: addActionListener       public synchronized void java.a
method: getActionListeners      public synchronized java.awt.ev
method: processActionEvent      protected void java.awt.Button.
method: removeActionListener    public synchronized void java.a
method: setActionCommand        public void java.awt.Button.set
method: setLabel        public void java.awt.Button.setLabel(ja
        java.awt.Component
        field:  focusLog        private static final java.util.
        field:  log     private static final java.util.logging.
        field:  mixingLog       private static final java.util.
        field:  peer    transient java.awt.peer.ComponentPeer j
        field:  parent  transient java.awt.Container java.awt.C
```

Appendix B – Preparing Rapise for Java SWT

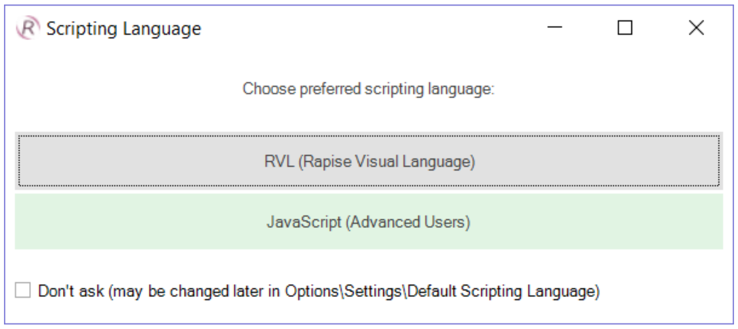
The **Java SWT** GUI library is an alternative to Swing developed by the **Eclipse** foundation and it provides Java applications the ability to access the **native GUI libraries** of the operating system using JNI (Java Native Interface) in a manner that is similar to those programs written using operating 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).

Appendix C – Using JavaScript Scripting Mode

In the main section of the guide we used the Rapise Visual Language (RVL) to write the tests. For advanced users that are familiar with standard programming languages, you can use the JavaScript language editor instead.

To do that, choose the **JavaScript** option during test creation instead of RVL.


Scripting Language

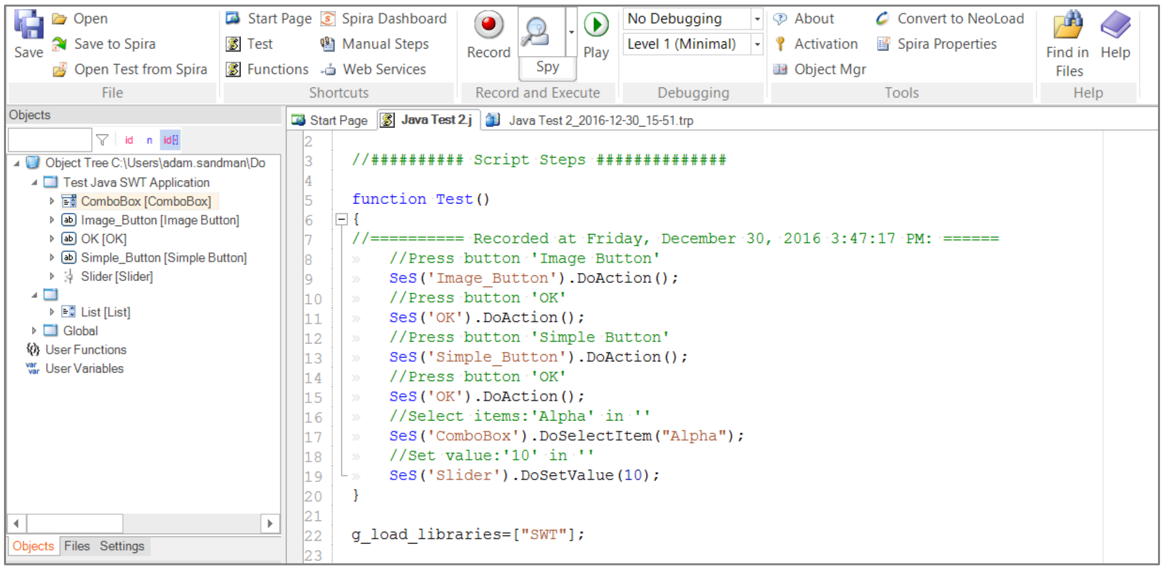
Choose preferred scripting language:

RVL (Rapise Visual Language)

JavaScript (Advanced Users)

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

The difference will be that when you finish recording the script it will as follows:



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

function Test()
{
//========== Recorded at Friday, December 30, 2016 3:47:17 PM: ======
    //Press button 'Image Button'
    SeS('Image_Button').DoAction();
    //Press button 'OK'
    SeS('OK').DoAction();
    //Press button 'Simple Button'
    SeS('Simple_Button').DoAction();
    //Press button 'OK'
    SeS('OK').DoAction();
    //Select items:'Alpha' in ''
    SeS('ComboBox').DoSelectItem("Alpha");
    //Set value:'10' in ''
    SeS('Slider').DoSetValue(10);
}

g_load_libraries=["SWT"];
```

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.

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