Rapise® | Quick Start Guide

Testing Windows® Applications with Rapise

Date: May 4th, 2017

Contents

Introduction

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**
- **Third Party Component Libraries**
  - Infragistics WinForm Controls
  - Telerik RadControls
  - DevExpress Controls
  - ComponentOne ActiveX Controls
  - SyncFusion Windows Form Controls
  - FarPoint Spreadsheet Control

This tutorial walks you through how to use Rapise to test a sample Windows desktop application as well as provide information to help you test your applications.

1. 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.

2. Select the "**UIAutomation 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 "**UIAutomation**" - using the record application dialog box. 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.

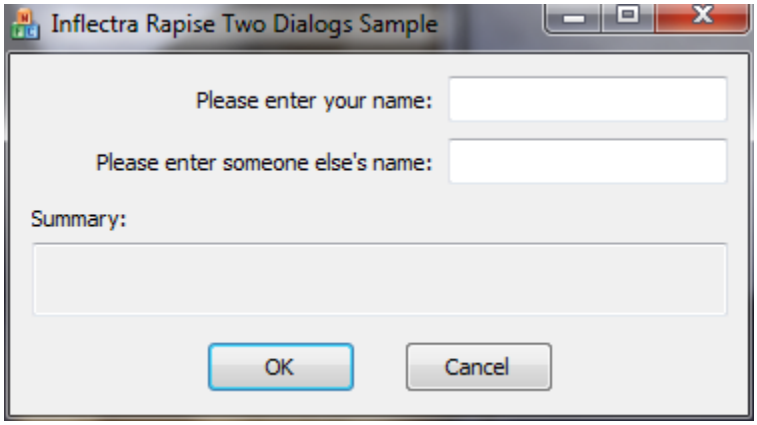
6. If neither of options below satisfy then it may be worth to try low level recording (https://www.inflectra.com/Support/KnowledgeBase/KB114.aspx).

2. Sample Applications

In the Samples index of the Rapise help manual you can find 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

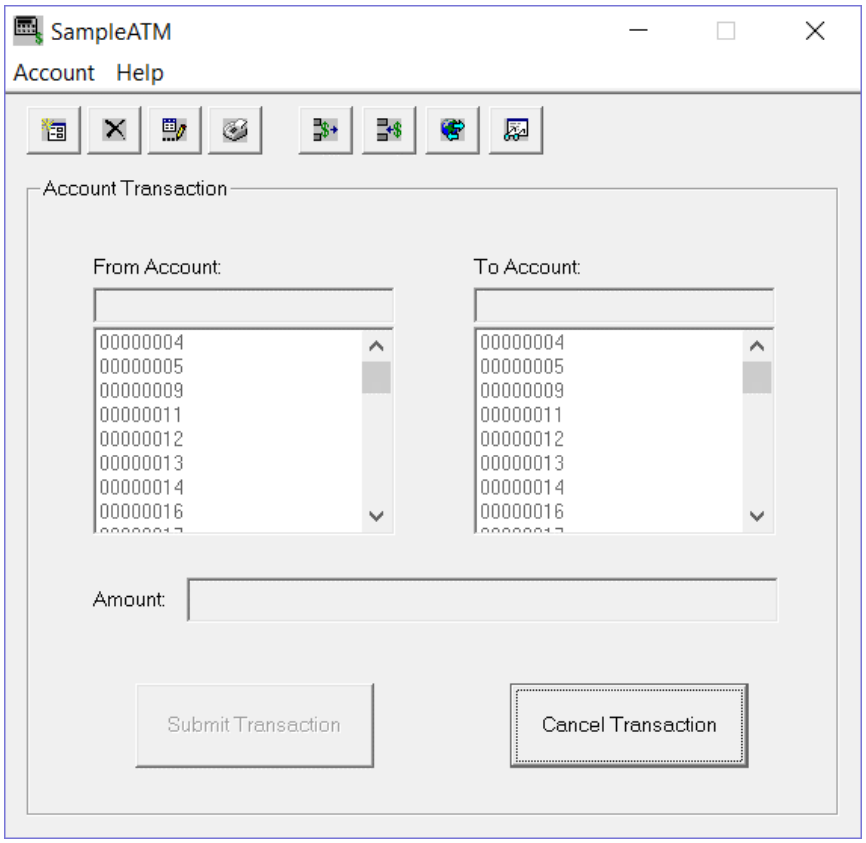
This is a simple Win32 MFC application that uses the **Generic** library and lets you try some basic testing scenarios.



This sample is the one described in more detail in **Section 3**, where we walk you through the process of creating a new test from scratch and record/playback some simple actions against it.

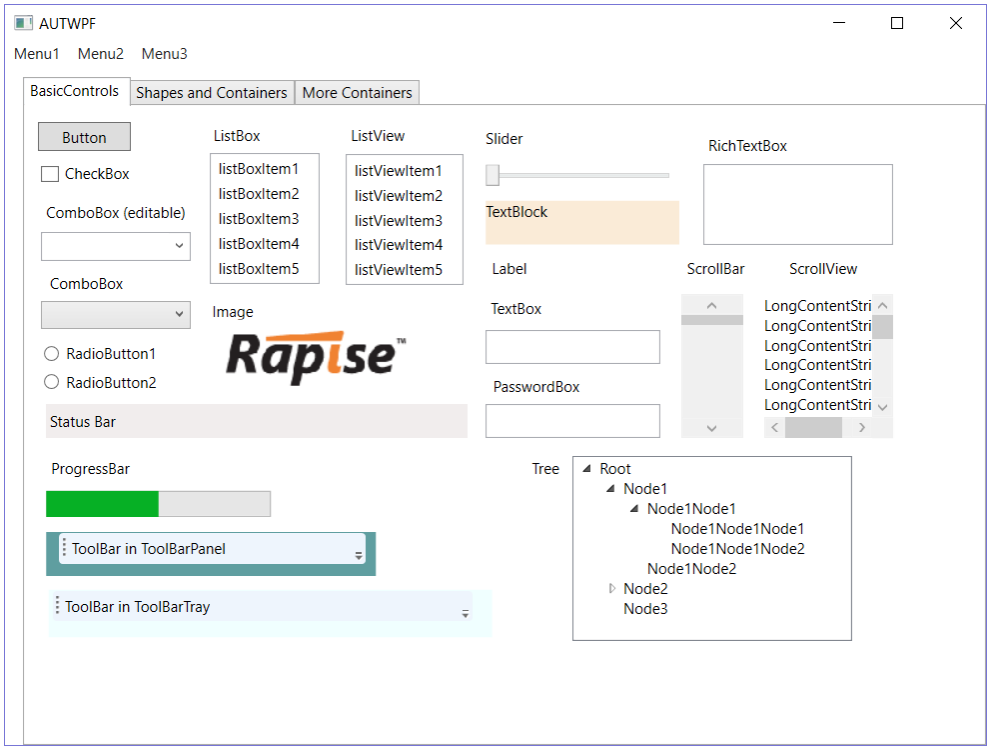
b) Sample ATM

This is a more robust Win32 MFC application that is based on a banking ATM. You should use the **Generic** library with this application:



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:



In the next two sections, we will demonstrate how to test the TwoDialogs sample application. Section 3 will demonstrate testing the application using the JavaScript scripting option and Section 4 will demonstrate testing the application using the Rapise Visual Language (RVL) scriptless option.

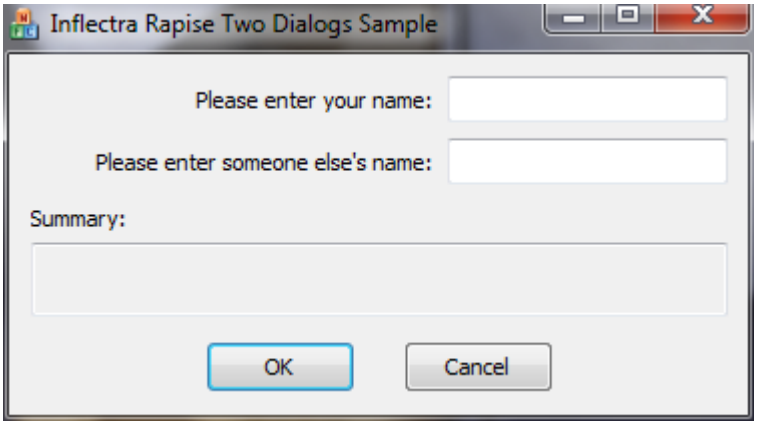
3. Testing the Two Dialogs Sample – Using JavaScript

This section outlines the usage of Rapise for testing a simple Windows Desktop Application Under Test (AUT) using the JavaScript scripting option. If you want to learn how to record the test using the Rapise Visual Language (RVL) instead, please go to Section 4 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:



Please run the application a few times and observe its behavior. 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 read-only 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.

A good testing strategy for this simple application would 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).

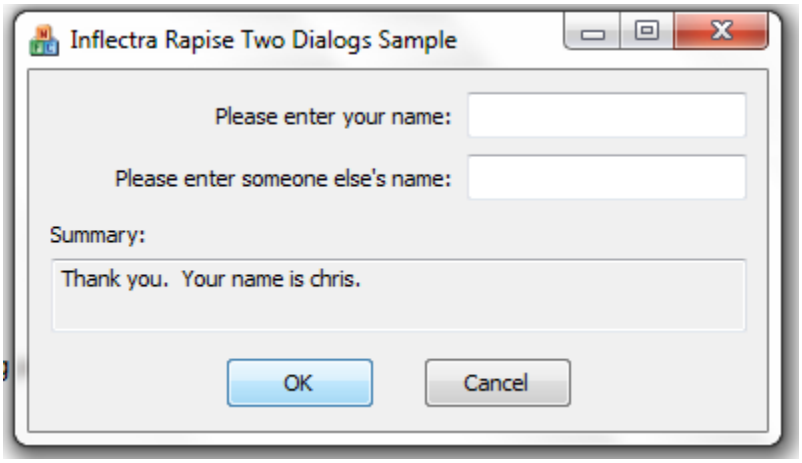
If at this point you do not understand what the application is supposed to do, or the application is not behaving as described here, please contact support and clarify the details before proceeding.

3.1. Recording and Playback of Test

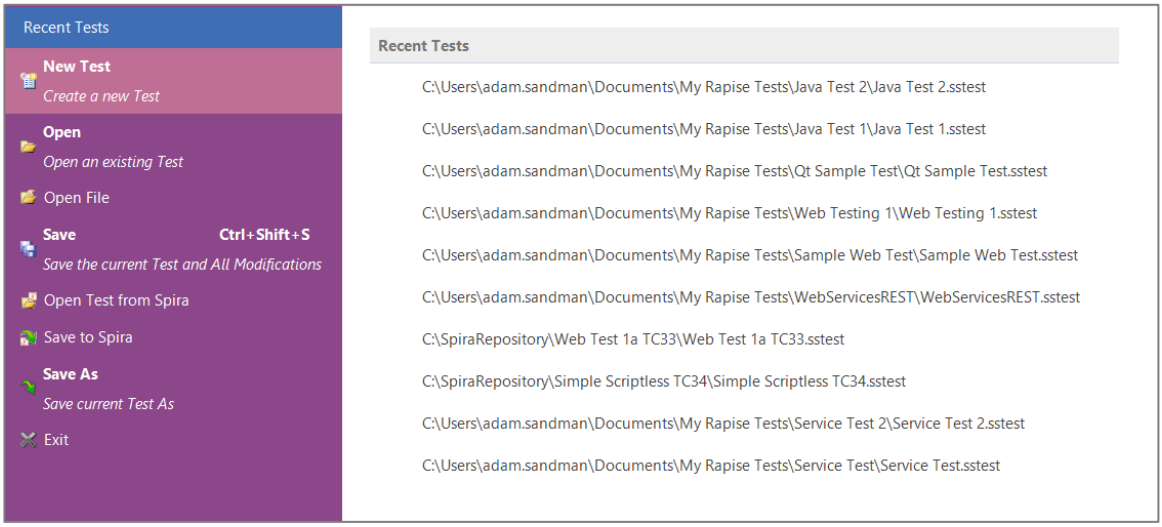
Now, let's use Rapise to implement the first of these tests.

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

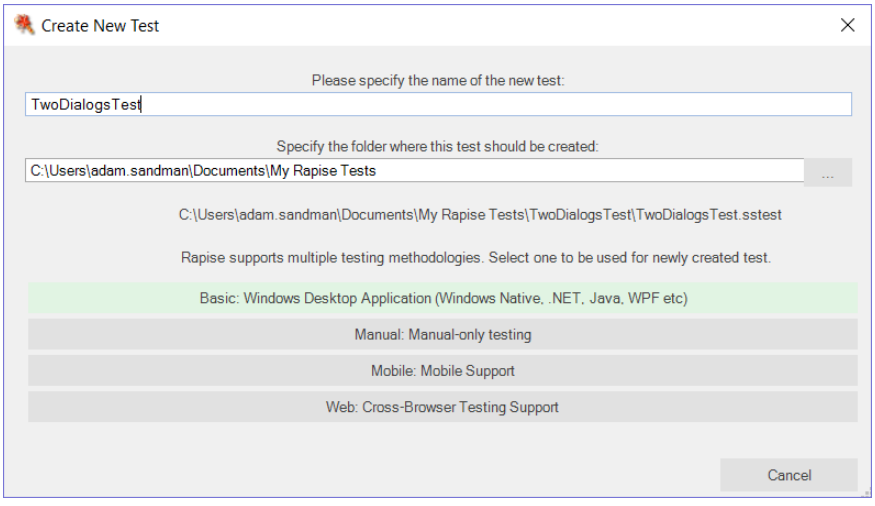
Once you execute the TwoDialogs.exe application it will be displayed on the screen:



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

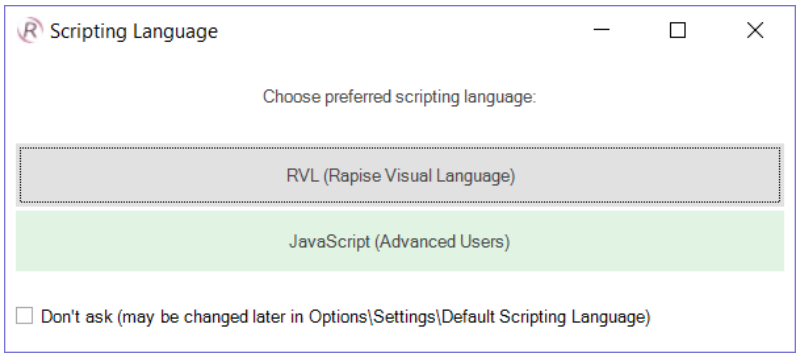


Step 3. Navigate to the desired path using the "..." button on the "Create New Test" dialog.

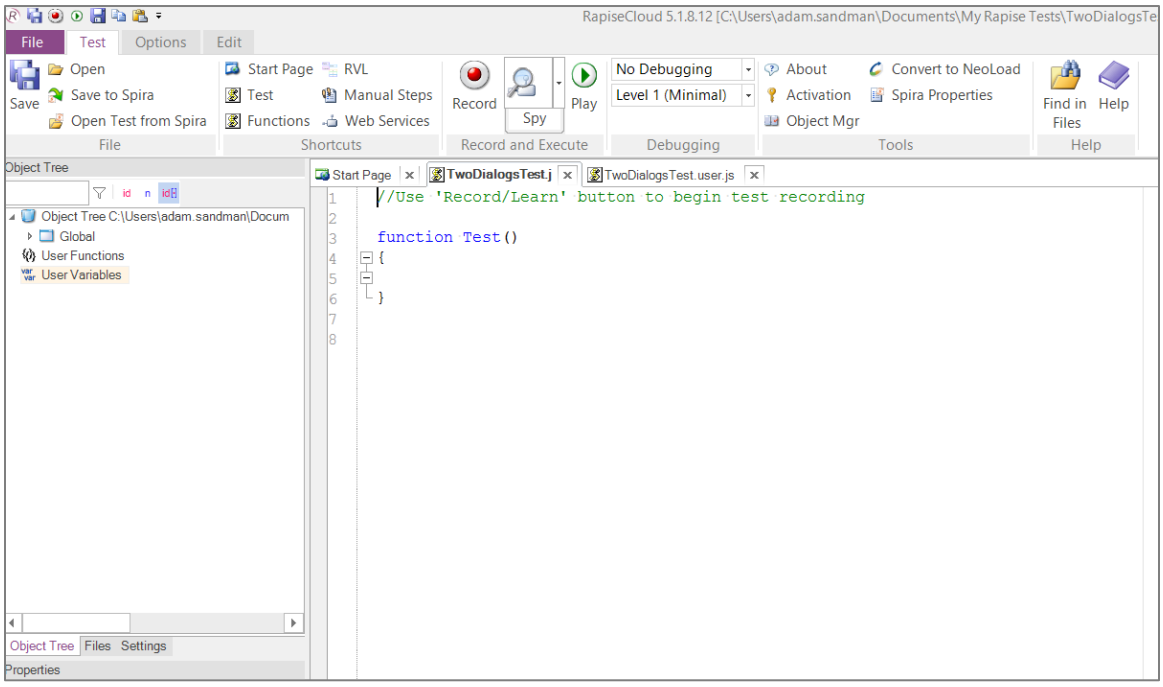


- Enter the name of the new test script we're going to write (e.g. "**TwoDialogsTest**").
- Click on the "**Basic: Windows Desktop Application**" methodology. This should always be used for testing Windows desktop applications.

The following dialog will be displayed:

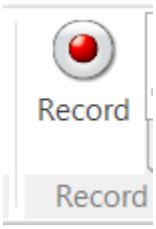


- Click on the **JavaScript (Advanced Users)** button.
- You will now see the following:

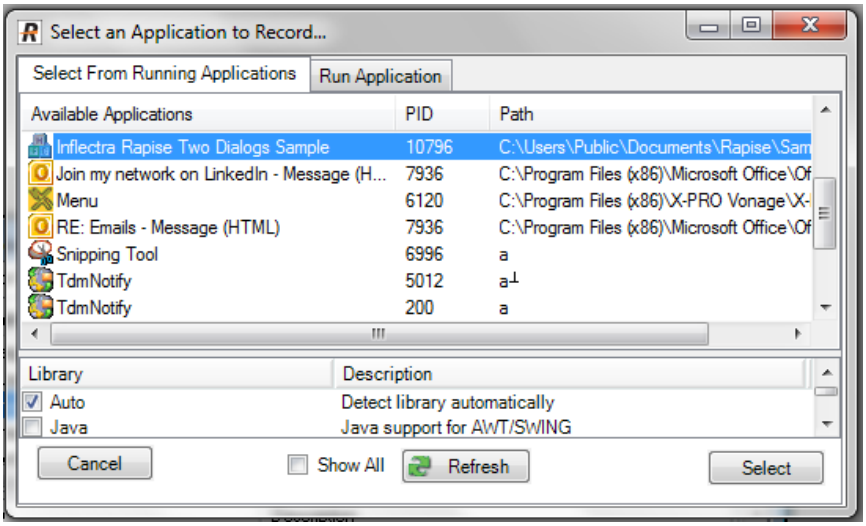


Step 4. Recording the test sequence.

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.

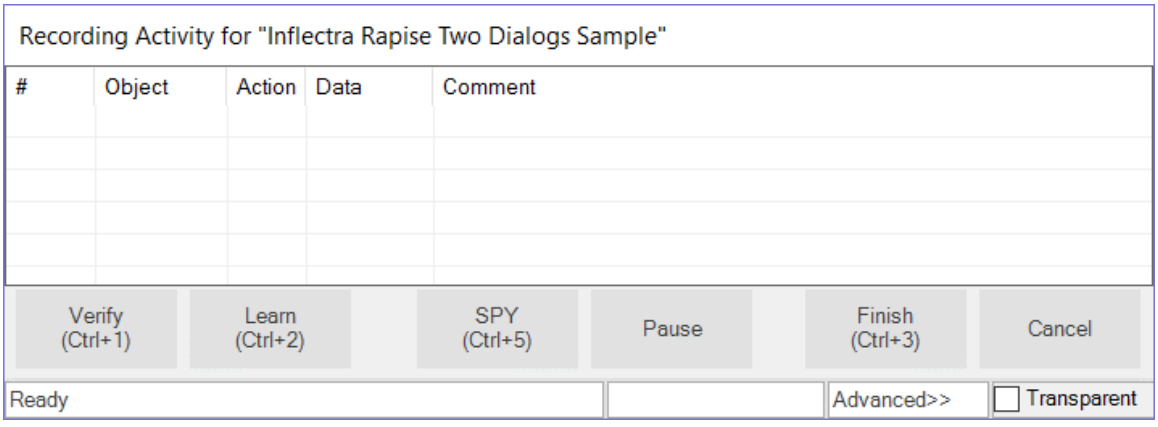


- Select the "Inflectra Rapise Two Dialogs Sample" entry.
- Leave the library selection as "Auto."
- Press the "Select" button at the bottom right.

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:



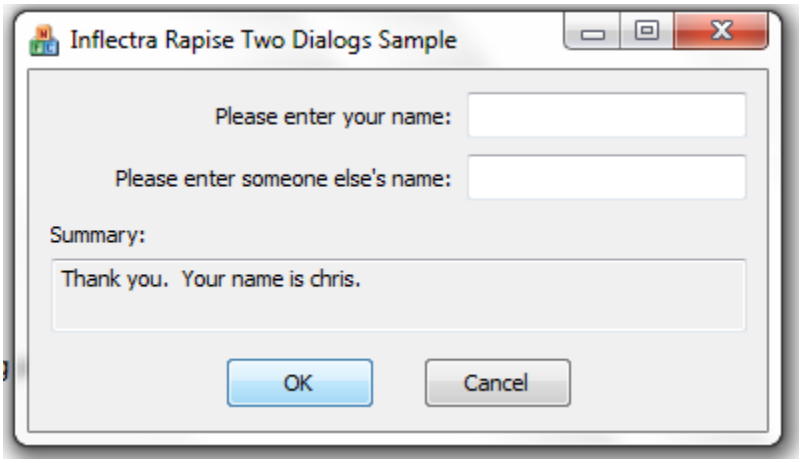
The AUT will be brought to the foreground and Rapise will be minimized.
You will achieve best results in recording if you observe the following guidelines:

(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.

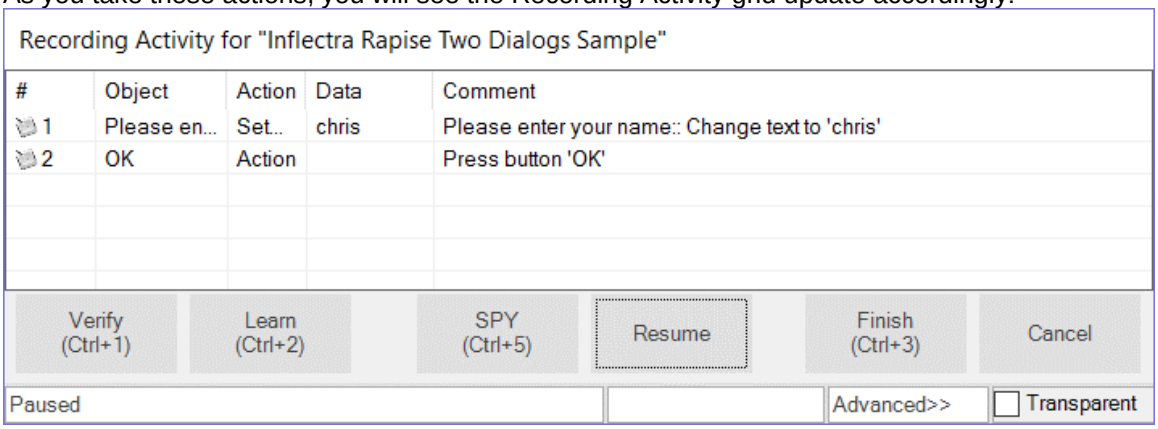
Step 6. Perform the Test Activities.

- 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.



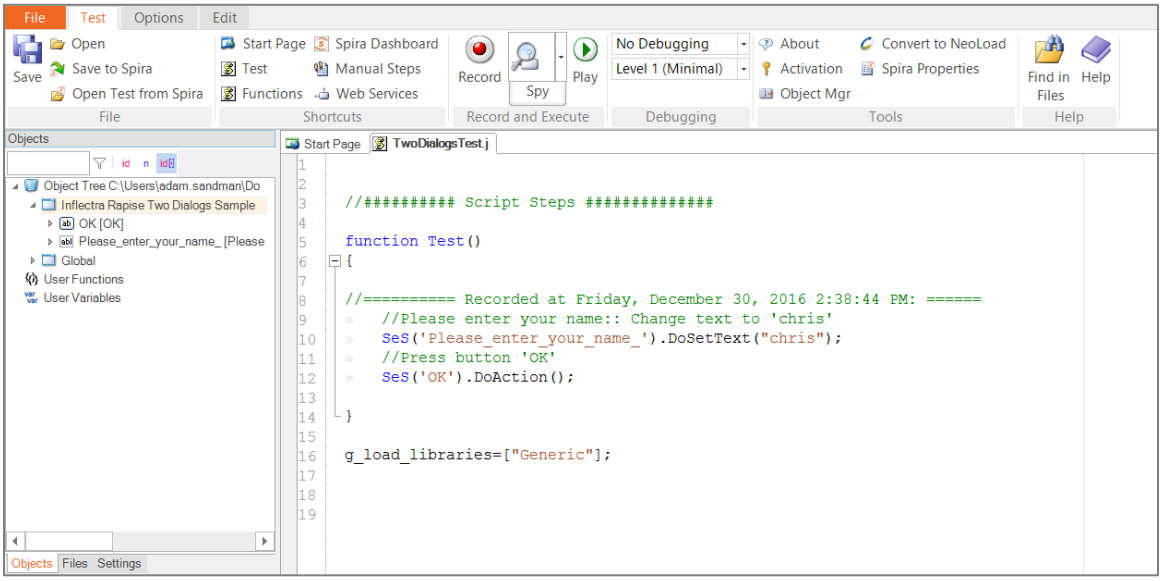
As you take these actions, you will see the Recording Activity grid update accordingly:



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

Note: Do not terminate the **TwoDialogs** application.

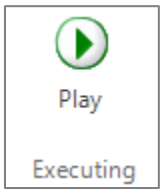
When you do this, the "Recording Activity" dialog will be closed and the AUT will lose focus. Rapise will change the view to display the newly recorded script. It will look something like the following:



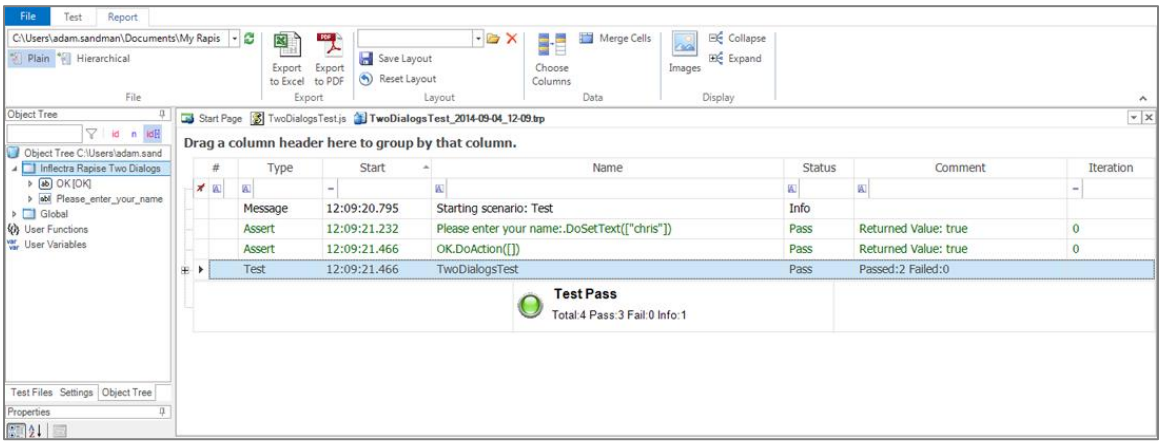
Notice that the two steps of the script are automatically documented and that they correspond precisely and in the same order as the way they appeared in the Recording Activity dialog during recording.

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:



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

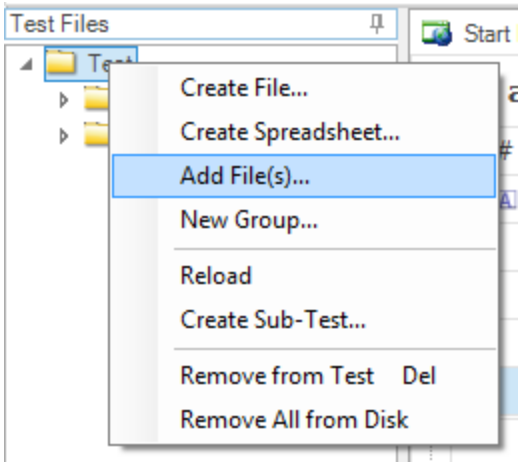
To date, we have operated on the assumption that the `TwoDialogs` sample program (application) is running. If this situation remained, the test script would require that the AUT be running before the script started. That would require that the person running the test remembered where it resided. To overcome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

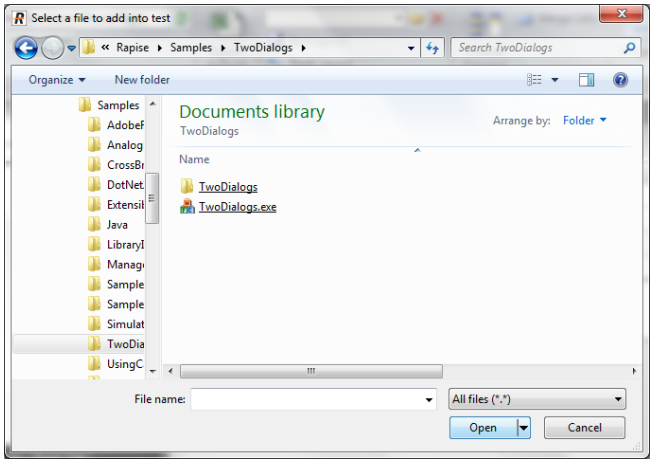
There are at least 3 ways of adding application launch code to your test.

Way 1: Drag The File from the Test Files view

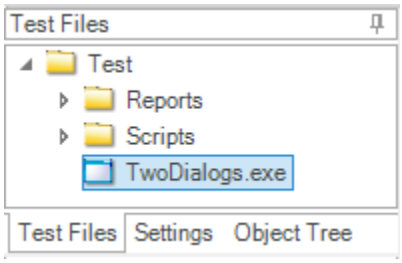
First, switch to Test Files view. Right-click on "Test" folder and choose "Add File(s)..." menu item:



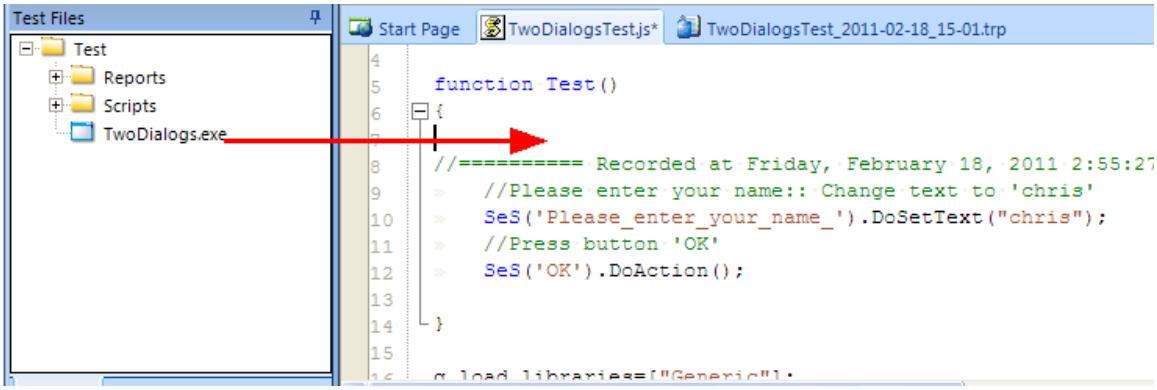
And select the location of the `TwoDialogs.exe` (normally, it is `C:\Program Files\Inflectra\Rapise\Samples\TwoDialogs\TwoDialogs.exe`),



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



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()
{
    Global.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

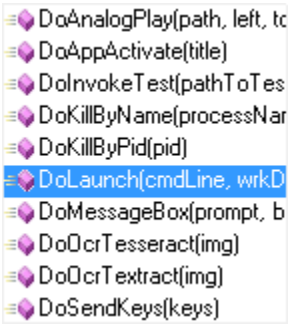
The `Global` object contains methods that are available to all scripts.
Select the `TwoDialogs.js` file in the Test Files view of the Rapise main page.
Double-click the file name to open it in the main (editing) window of Rapise. You will see the generated script from the recording session from earlier 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:



Select the DoLaunch(cmdLine, wrkD) member and hit the Enter key.
Now your script contains the line:

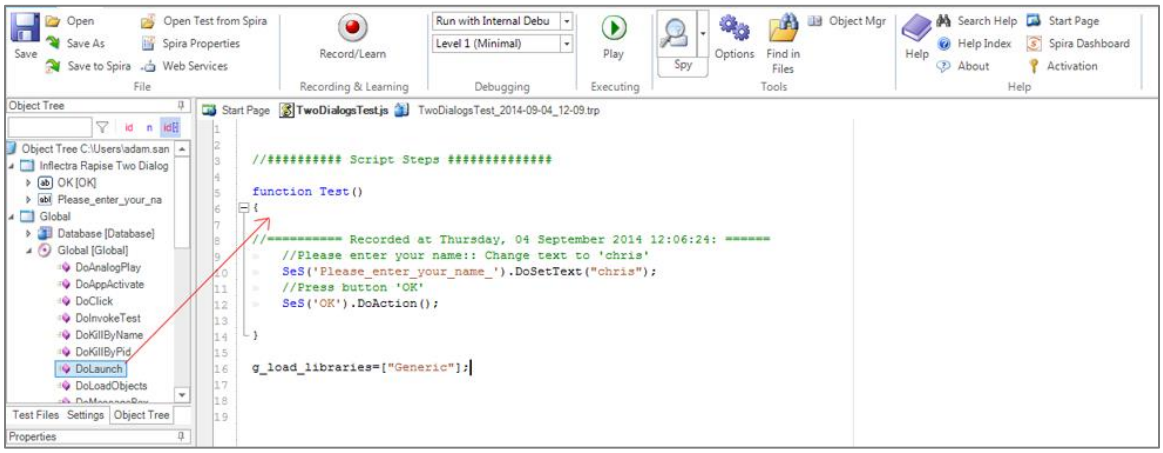
```
Global.DoLaunch('')
```

You need to correct the references to the command line:

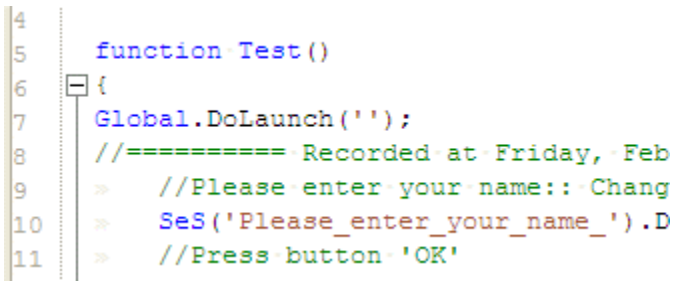
```
Global.DoLaunch('"C:\\Program
Files\\Inflectra\\Rapise\\Samples\\TwoDialogs\\TwoDialogs.exe"');
```

Way 3: Drag the Action from the Objects Tree

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



The template call is inserted automatically:



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

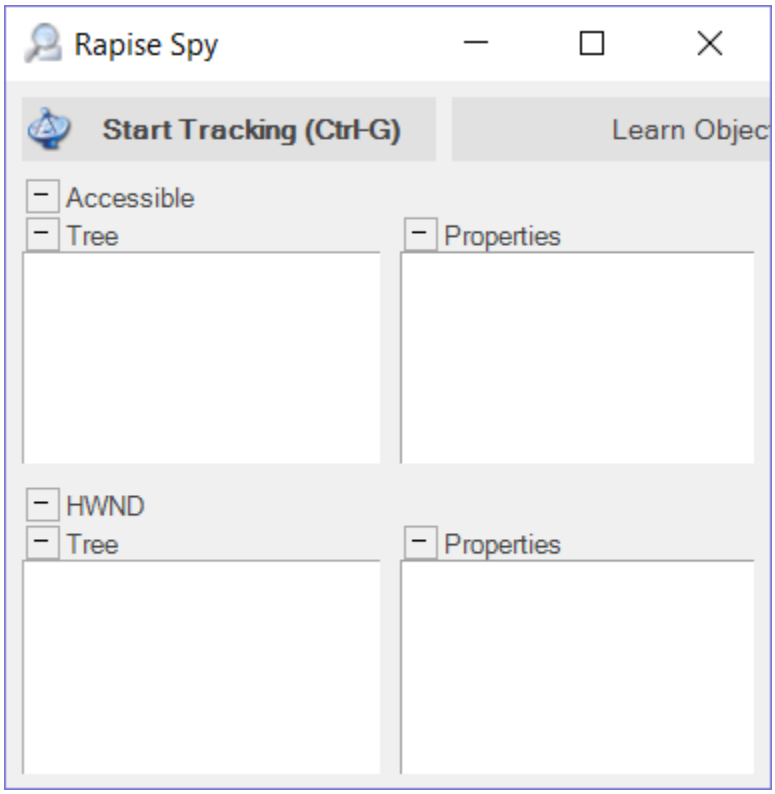
```
Global.DoLaunch('"C:\\Program
Files\\Inflectra\\Rapise\\Samples\\TwoDialogs\\TwoDialogs.exe"');
```

3.2. Advanced Testing using the Object Spy

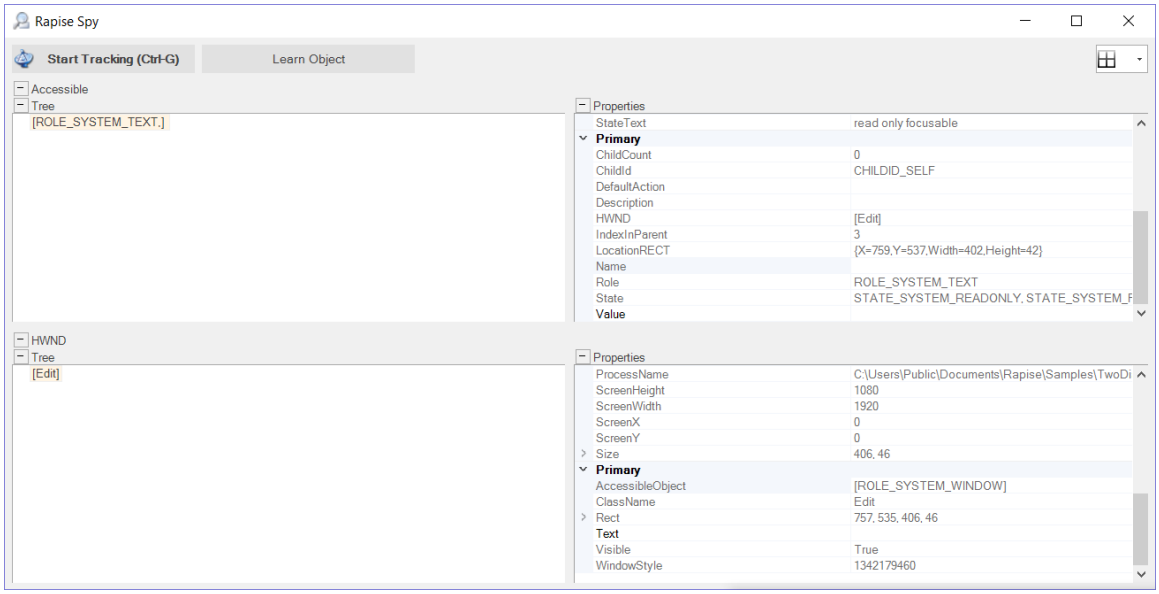
Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

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

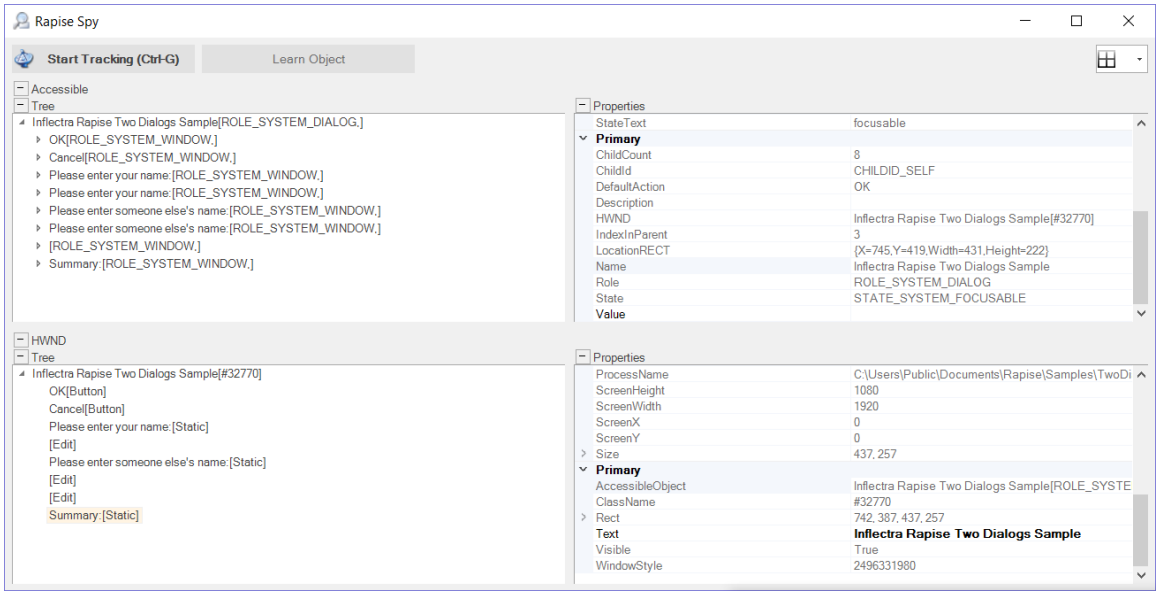
When you are in the middle of recording, click on the **Spy** button and Rapise will display the Accessible Spy:



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:



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:



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.

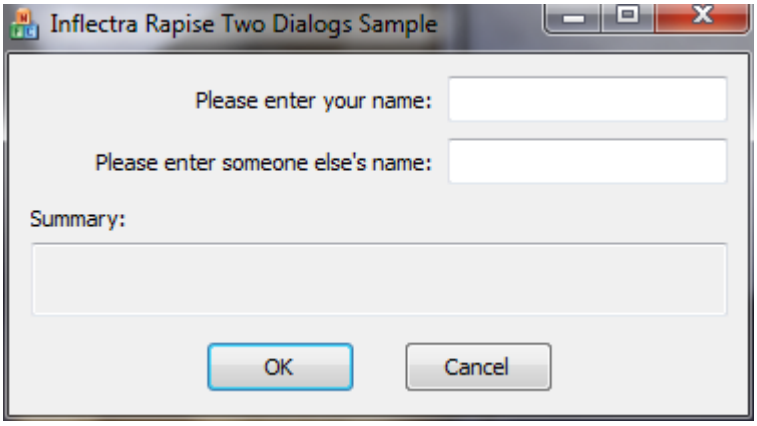
4. Testing the Two Dialogs Sample – Using RVL

This section outlines the usage of Rapise for testing a simple Windows Desktop Application Under Test (AUT) using the Rapise Visual Language (RVL) scripting option. If you want to learn how to record the test using JavaScript as your test script language, please read Section 3 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:



Please run the application a few times and observe its behavior. 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 read-only 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.

A good testing strategy for this simple application would be to:

4. Put data in the first text box but not the second, and verify that the summary information is correct.
5. Press the OK button with no data in either text box, and verify that a message box is displayed.
6. Verify that if the success "Thank You" message is displayed the edit box input fields are cleared (but not the summary information).

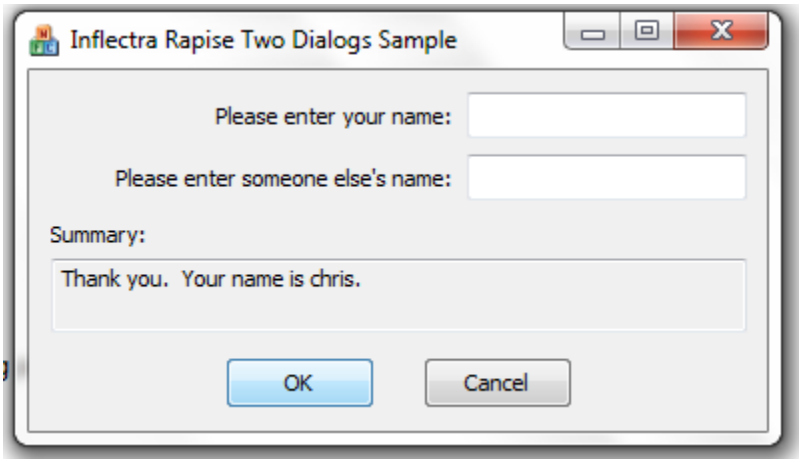
If at this point you do not understand what the application is supposed to do, or the application is not behaving as described here, please contact support and clarify the details before proceeding.

4.1. Recording and Playback of Test

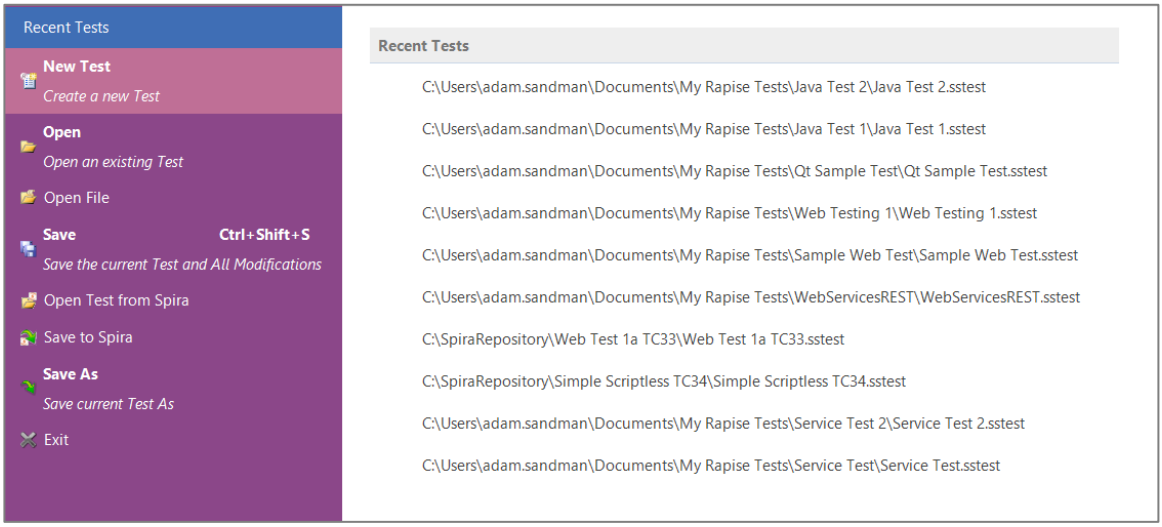
Now, let's use Rapise to implement the first of these tests.

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

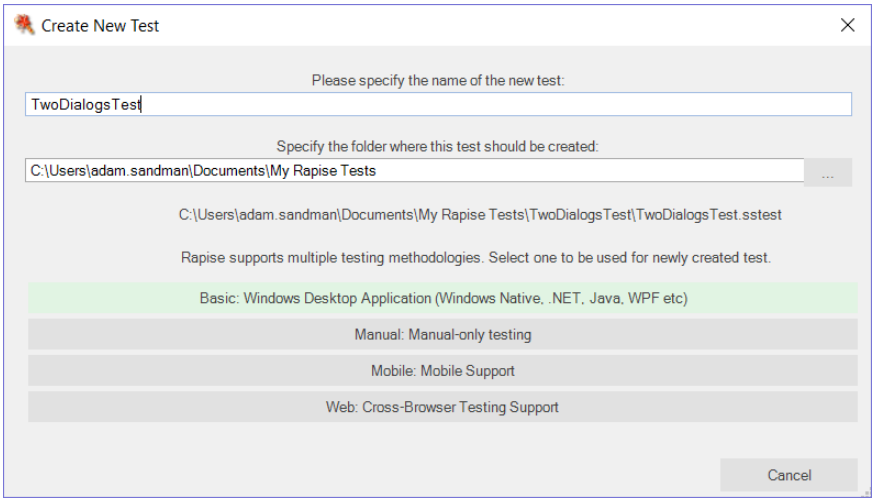
Once you execute the TwoDialogs.exe application it will be displayed on the screen:



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

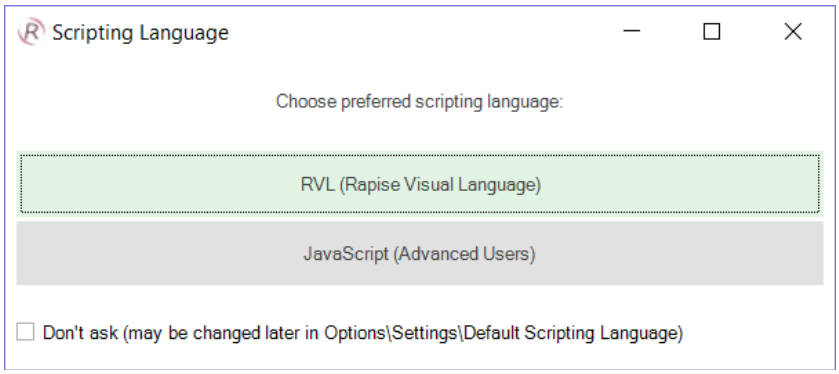


Step 3. Navigate to the desired path using the "..." button on the "Create New Test" dialog.

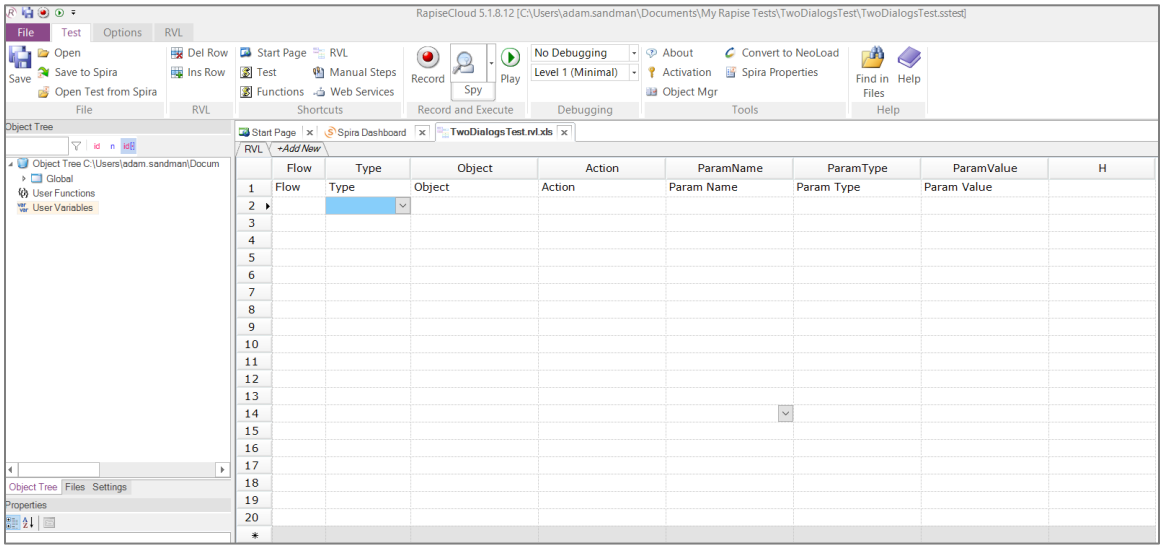


- Enter the name of the new test script we're going to write (e.g. "**TwoDialogsTest**").
- Click on the "**Basic: Windows Desktop Application**" methodology. This should always be used for testing Windows desktop applications.

The following dialog will be displayed:

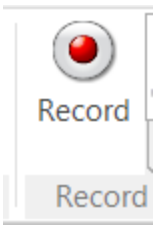


- Click on the **RVL (Rapise Visual Language)** button.
- You will now see the following:

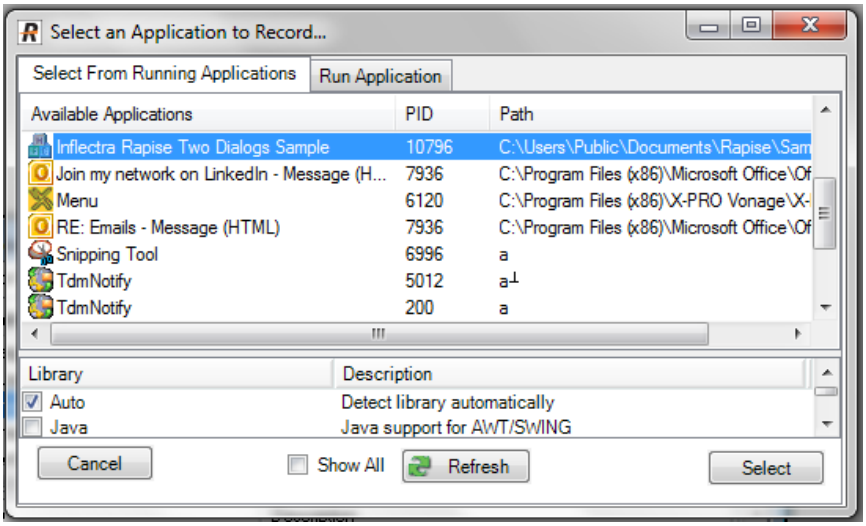


Step 4. Recording the test sequence.

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.

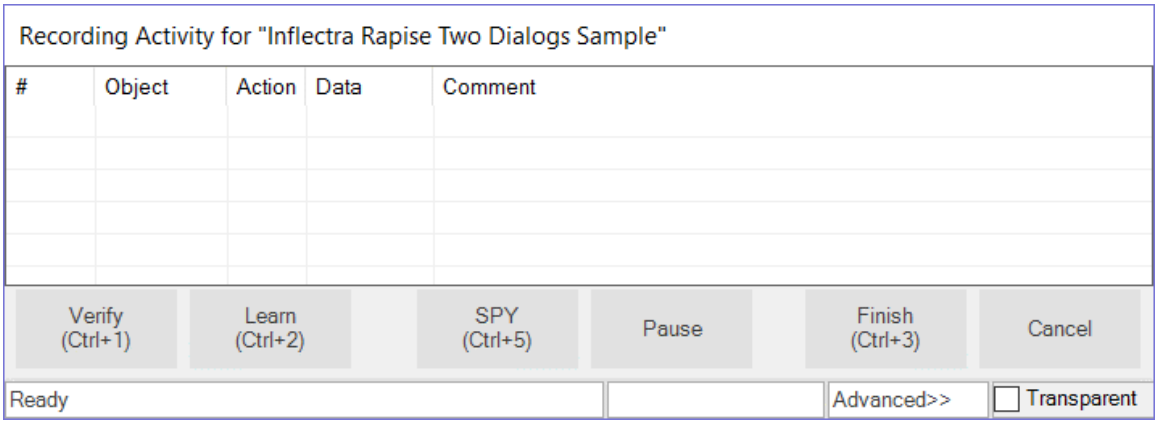


- Select the "Inflectra Rapise Two Dialogs Sample" entry.
- Leave the library selection as "Auto."
- Press the "Select" button at the bottom right.

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:



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:

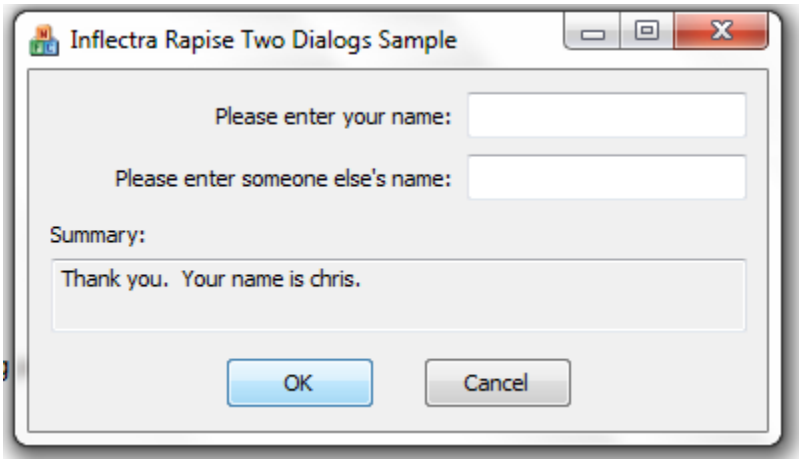
(3) Work slowly while recording. Perform one action and wait for the results to be recorded in the Recording Activity dialog as a new grid line-item before going to the next item.

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

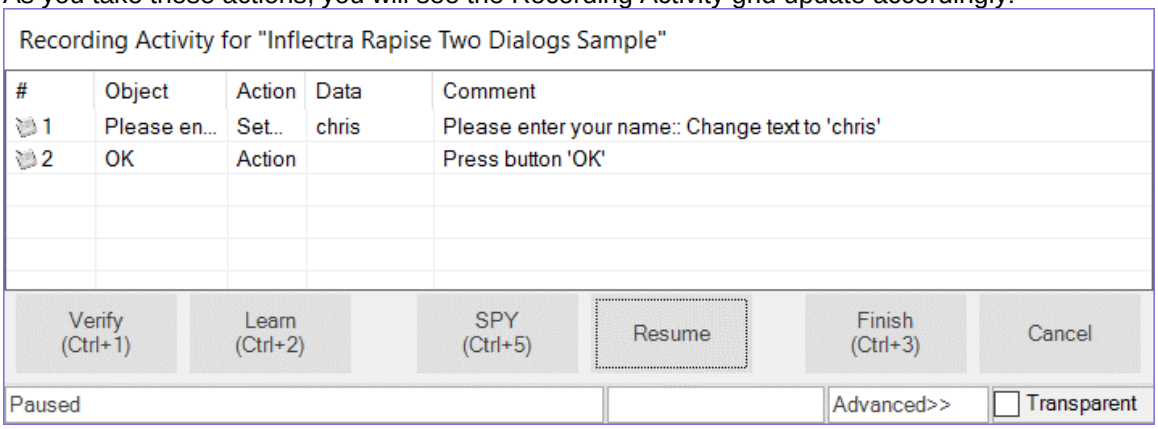
Step 6. Perform the Test Activities.

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

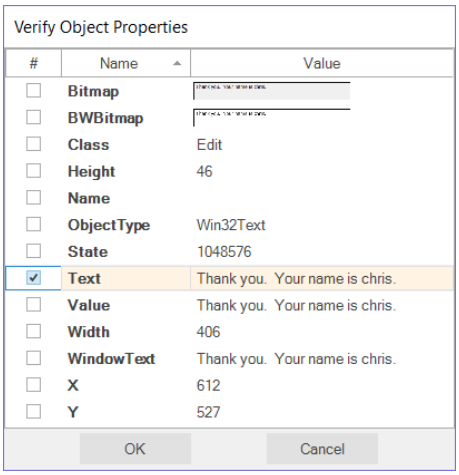
Watch the Recording activity dialog as you operate the AUT interface. As you press a button or fill a field, notice that the grid in the Recording activity has entries added to it.



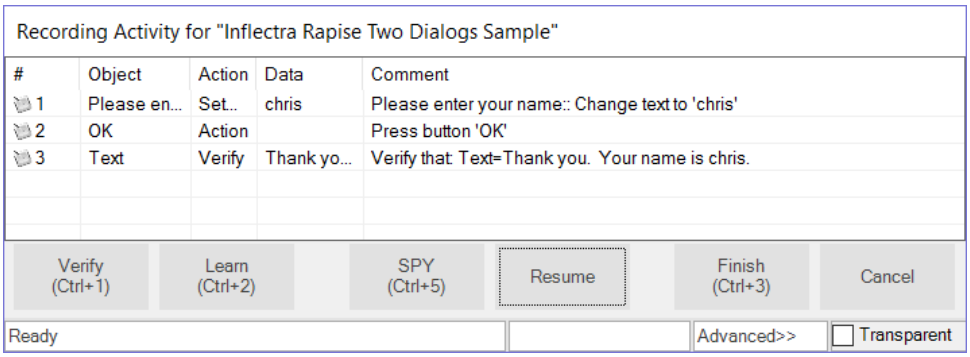
As you take these actions, you will see the Recording Activity grid update accordingly:



Hover the mouse over the “Thank you. Your name is chris” result box and click CTRL+1 on the keyboard to Verify the output. This will display the verification box:



Check the box marked “**Text**” and click OK. Rapise will now record the verification test:



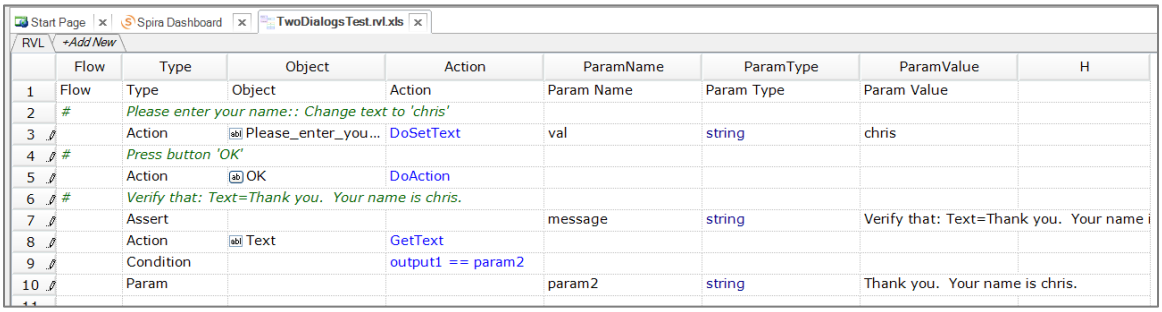
When you have finished recording the activity for the AUT, press the "Finish" button or CTRL+3.

*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:

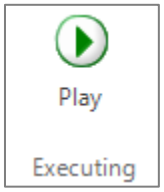
Click **‘Insert** Here’, then Rapise will change the view to display the newly recorded test steps. It will look something like the following:



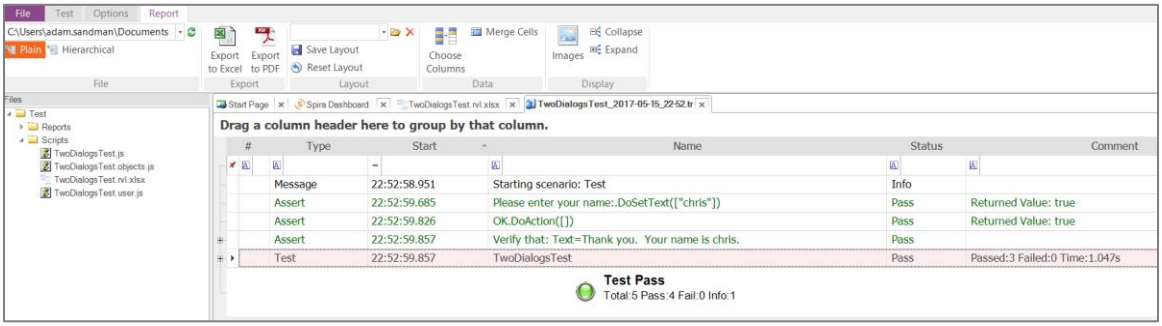
Notice that the two steps of the test are automatically documented as Action commands in the test grid, and that they correspond precisely and in the same order as the way they appeared in the Recording Activity dialog during recording. In addition, the Verify checkpoint has been recorded as an Assert…Condition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

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

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



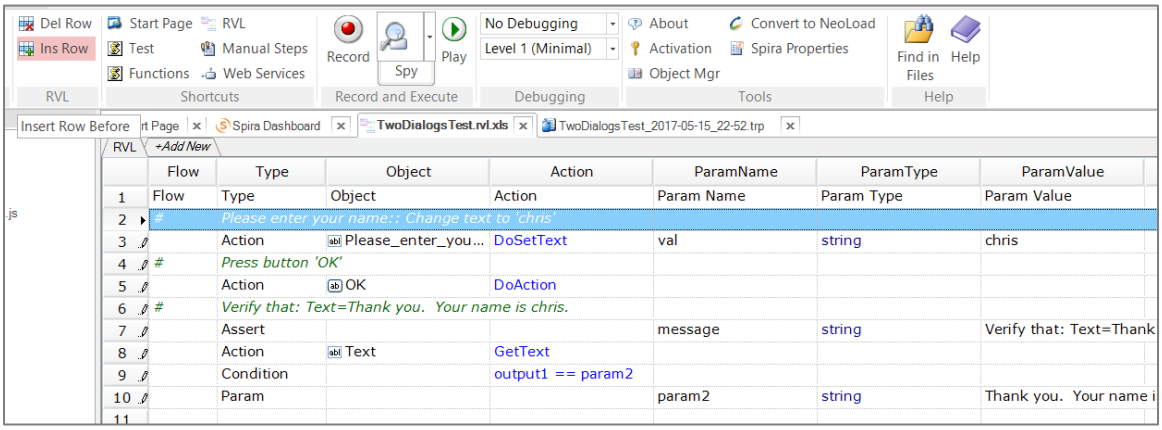
As the script runs, the Rapise window will be minimized to the taskbar and you will see the results of the script's activities on the TwoDialogs application window.
At the end of the script execution, the Rapise window will be restored and the view will be of the report for the test:



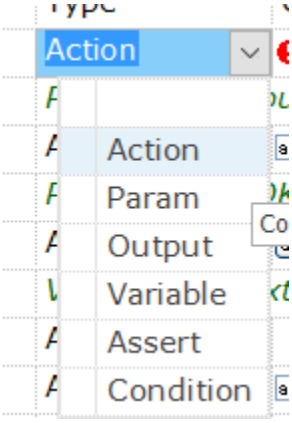
Step 8: A refinement on the launching of TwoDialogs.exe.

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.

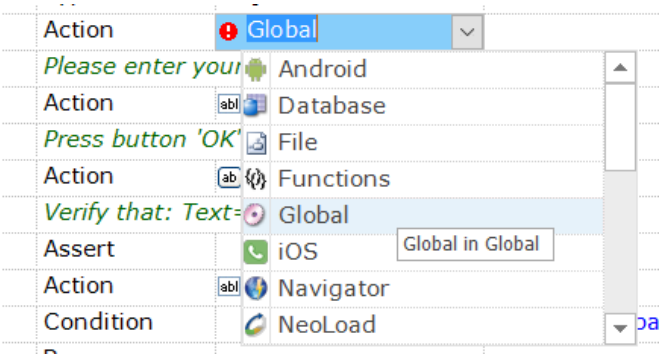
Rapise comes with a series of useful Global utility objects that can do things such as start applications, kill processes, access the file system, etc. To launch the TwoDialogs application at the start of the test, go to the first row in the test grid and click **"Ins Row"** in the RVL ribbon:



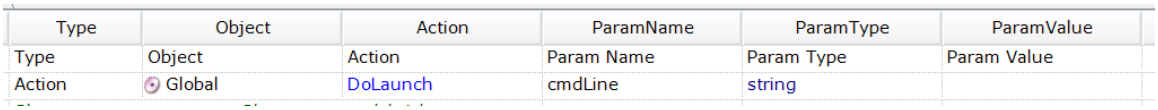
Now in the new row that was created, choose **Action** as the type of row:



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

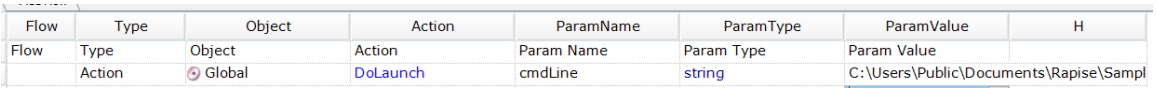


Then in the next cell, choose **DoLaunch** and press ENTER on the keyboard:



Type	Object	Action	ParamName	ParamType	ParamValue
Type	Object	Action	Param Name	Param Type	Param Value
Action	Global	DoLaunch	cmdLine	string	

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	Type	Object	Action	ParamName	ParamType	ParamValue	H
Flow	Type	Object	Action	Param Name	Param Type	Param Value	
	Action	Global	DoLaunch	cmdLine	string	C:\Users\Public\Documents\Rapise\Sampl	

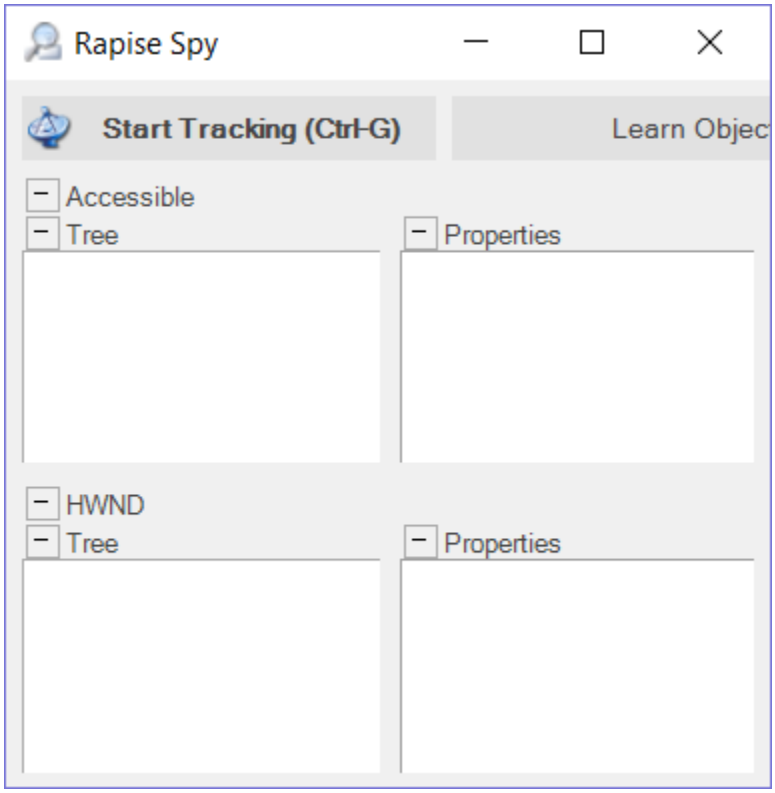
Now if you click **Play**, Rapise will launch the application and then complete the recorded test steps.

4.2. Advanced Testing using the Object Spy

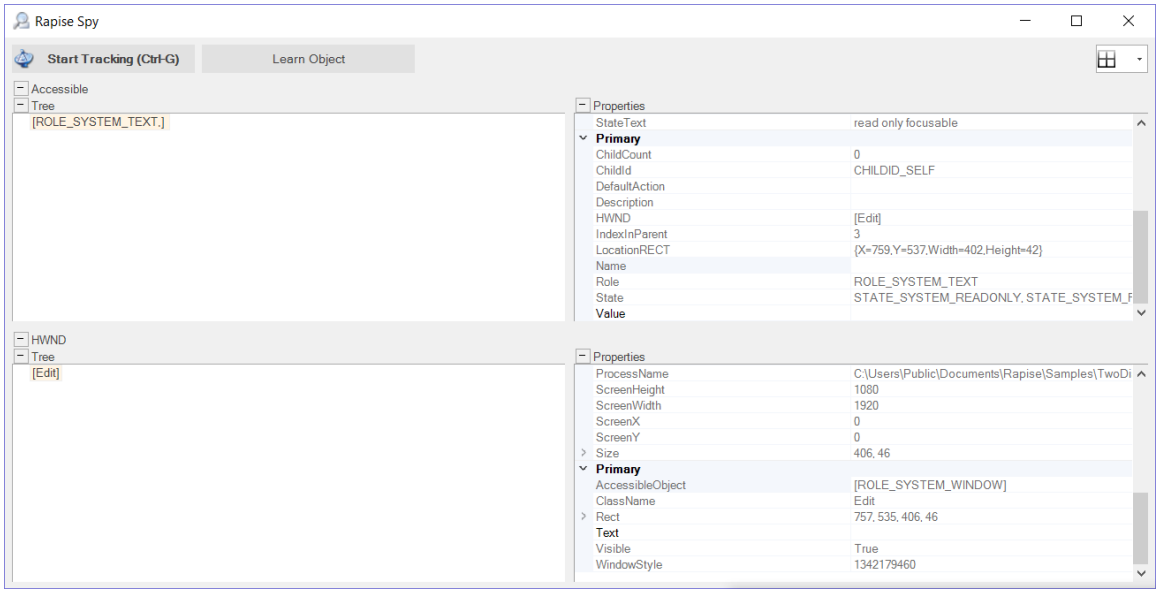
Sometimes you need to learn objects that are not visible or are obscured by other objects. To help with this, Rapise has the Object Spy tool.

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

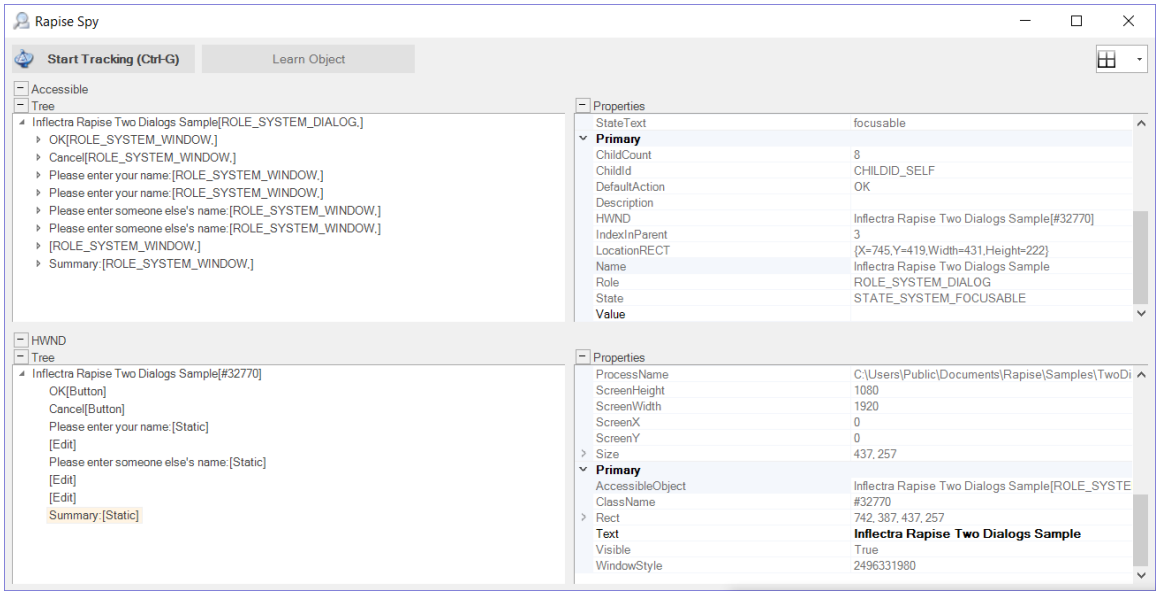
When you are in the middle of recording, click on the **Spy** button and Rapise will display the Accessible Spy:



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:



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:



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.

Legal Notices

This publication is provided as is without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

This publication could include technical inaccuracies or typographical errors. Changes are periodically added to the information contained herein; these changes will be incorporated in new editions of the publication. Inflectra Corporation may make improvements and/or changes in the product(s) and/or program(s) and/or service(s) described in this publication at any time.

The sections in this guide that discuss internet web security are provided as suggestions and guidelines. Internet security is constantly evolving field, and our suggestions are no substitute for an up-to-date understanding of the vulnerabilities inherent in deploying internet or web applications, and Inflectra cannot be held liable for any losses due to breaches of security, compromise of data or other cyber-attacks that may result from following our recommendations.

SpiraTest®, SpiraPlan®, SpiraTeam®, Rapise® and Inflectra® are either trademarks or registered trademarks of Inflectra Corporation in the United States of America and other countries. Microsoft®, Windows®, Explorer® and Microsoft Project® are registered trademarks of Microsoft Corporation. All other trademarks and product names are property of their respective holders.

Please send comments and questions to:

Technical Publications

Inflectra Corporation

8121 Georgia Ave, Suite 504

Silver Spring, MD 20910-4957

U.S.A.

support@inflectra.com