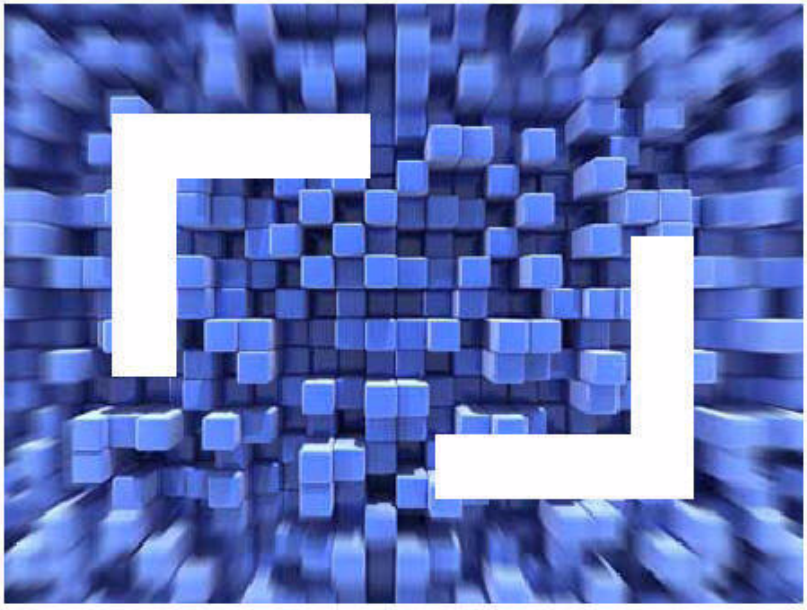


SilkTest® 2011
Testing Flex Applications

Borland®
(A MICRO FOCUS COMPANY)

MICRO® FOCUS
Leading the Evolution™

Borland Software Corporation
4 Hutton Centre Dr., Suite 900
Santa Ana, CA 92707

July 2011

Contents

Introduction

Introduction

This chapter provides an overview of Adobe Flex support for SilkTest Classic and Silk4J and the sample applications available with SilkTest.

What you will learn

This book contains the following chapters:

Section	Page
Enabling Your Flex Application for Testing	7
Tutorial for Flex Application Testing	13

Overview of Adobe Flex Support

SilkTest provides built-in support for testing Adobe Flex applications using the 4Test scripting language. You can also test Adobe Flex applications with the Silk4J Eclipse plug-in using the Java programming language. Silk4J is an optional program. During installation, you must specify that you want to install Silk4J in order to use it.

For details about supported versions and known Flex issues, refer to the *Release Notes*.

Sample Applications

SilkTest provides several sample Adobe Flex test applications. You can use these sample applications to record tests with SilkTest and Silk4J.

You must access the Flex sample applications from http://demo.borland.com/flex/SilkTest2011/3.5/Flex3TestApp_withAutomation/Flex3TestApp.html.

The Quick Tour chapter uses the Component Explorer sample application to walk you through testing a Flex application. However, if you prefer, you can use

your own Flex application to perform these steps. If you use your own application, follow the steps in the "Enabling Your Flex Application for Testing" chapter before you begin the Quick Tour.

1

Enabling Your Flex Application for Testing

Introduction

This chapter details how to use the Adobe Flex Automation API to prepare your Flex application for automation testing using SilkTest. Flex developers are the target audience for this document.

For information about which version of Flex to use and which browsers and operating environments are supported for testing, refer to the *SilkTest Release Notes*.

Enabling Your Application for Testing

To enable your Flex application for testing, you must include the following components in your application:

- "Adobe Flex Automation Package"
- "SilkTest Automation Package"

You can load these packages at run time or prior to run time by precompiling your application. When you load the automation packages at run time, your application is not modified. However, this method is difficult to use in applications that are tested in a web browser. For details about these limitations, see "Limitations" on page 8. In contrast, precompiling your application modifies your application and increases the file size, which means that you must create two builds, one for testing and one for release. However, this method works for all applications.

For more details about the differences between the run time and precompiled approaches, refer to the Adobe guideline at http://download.macromedia.com/pub/documentation/en/flex/2/at_api.pdf.

Note If you are using a Flex sample application, you do not need to perform these steps. The sample applications have already been enabled for testing.

Loading Automation Packages at Run Time

You can load automation support at run time using the SilkTest Flex Automation Launcher. This application is compiled with the automation libraries and loads your application with the SWFLoader class. This automatically enables your application for testing without compiling automation libraries into your SWF file. The SilkTest Flex Automation Launcher is available in HTML and SWF file formats.

Limitations

- The Flex Automation Launcher Application automatically becomes the root application. If your application must be the root application, you cannot load automation support with the SilkTest Flex Automation Launcher. For other options, see "Compiling Automation Packages Prior to Run Time" on page 9.
- Testing applications that load external libraries – Applications that load other SWF file libraries require a special setting for automated testing. A library that is loaded at run time (including run-time shared libraries (RSLs)) must be loaded into the ApplicationDomain of the loading application. If the SWF file used in the application is loaded in a different application domain, automated testing record and playback will not function properly. The following example shows a library that is loaded into the same ApplicationDomain:

```
import flash.display.*;
import flash.net.URLRequest;
import flash.system.ApplicationDomain;
import flash.system.LoaderContext;

var ldr:Loader = new Loader();

var urlReq:URLRequest = new
URLRequest("RuntimeClasses.swf");
var context:LoaderContext = new LoaderContext();
context.applicationDomain =
ApplicationDomain.currentDomain;
loader.load(request, context);
```

To use the Flex Automation Launcher for run-time loading

1. Copy the content of the **Silk\SilkTest\ng\AutomationSDK\Flex\<VERSION>\FlexAutomationLauncher** directory into the directory of the Flex application that you are testing.
2. Open **FlexAutomationLauncher.html** in Windows Explorer and add the following parameter as a suffix to the file path:

   `?automationurl=YourApplication.swf`

   where `YourApplication.swf` is the name of the .swf file for your Flex application.
3. Add `file:///` as a prefix to the file path.

   For example, if your file URL includes a parameter, such as: ?automationurl=explorer.swf, type: **file:///C:/Program%20Files/Silk/SilkTest/ng/samples/Flex/3.2/FlexControlExplorer32/FlexAutomationLauncher.html?automationurl=explorer.swf**.

For details about creating events and custom controls to support automated testing, refer to "Testing Flex Custom Controls" in the SilkTest online help.

Compiling Automation Packages Prior to Run Time

You can precompile applications that you plan to test. The functional testing classes are embedded in the application at compile time, and the application has no external dependencies for automated testing at run time.

When you embed functional testing classes in your application SWF file at compile time, the size of the SWF file increases. If the size of the SWF file is not important, use the same SWF file for functional testing and deployment. If the size of the SWF file is important, generate two SWF files, one with functional testing classes embedded and one without. Use the SWF file that does not include the embedded testing classes for deployment.

When you precompile the Flex application for testing, in the **`include-libraries compiler`** option, reference the following files:

- automation.swc
- automation_agent.swc
- FlexTechDomain.swc
- automation_charts.swc (include if your application uses charts and Flex 2.0)
- automation_dmv.swc (include if your application uses charts and Flex > 3.x)
- automation_flasflexkit.swc (include if your application uses embedded flash content)
- automation_spark.swc (include if your application uses the new Flex 4.x controls)
- automation_air.swc (include if your application is an AIR application)
- automation_airspark.swc (include if your application is an AIR application and uses new Flex 4.x controls)

When you create the final release version of your Flex application, you recompile the application without the references to these SWC files. For more information about using the automation SWC files, see the *Adobe Flex Release Notes*.

If you do not deploy your application to a server, but instead request it by using the file protocol or run it from within Adobe Flex Builder, you must include each SWF file in the local-trusted sandbox. This requires additional configuration information. Add the additional configuration information by modifying the compiler's configuration file or using a command-line option.

Modifying the Compiler's Configuration File to Add Configuration Information

1 Include the **automation.swc**, **automation_agent.swc**, and **FlexTechDomain.swc** libraries in the compiler's configuration file by adding the following code to the configuration file:

```
<include-libraries>
  ...
  <library>/libs/automation.swc</library>
  <library>/libs/automation_agent.swc</library>
  <library>pathinfo/FlexTechDomain.swc</library>
</include-libraries>
```

Note If your application uses charts, you must also add the **automation_charts.swc** file to the **`include-libraries`** compiler option.

2 Specify the location of the automation.swc, automation_agent.swc, and FlexTechDomain.swc libraries using the include-libraries compiler option with the command-line compiler.

The configuration files are located at:

Product	File location
Adobe Flex 2 SDK	<***`flex_installation_directory`***>/**frameworks/flex-config.xml**
Adobe Flex Data Services	<***`flex_installation_directory`***>/**flex/WEB-INF/flex/flex-config.xml**

Table 1: Configuration file locations

The following example adds the automation.swc and automation_agent.swc files to the application:

```
mxmlc -include-libraries+=../frameworks/libs/
automation.swc;../frameworks/libs/
automation_agent.swc;pathinfo/FlexTechDomain.swc
MyApp.mxml
```

Note Explicitly setting the **`include-libraries`** option on the command line overwrites, rather than appends, the existing libraries. If you add the **automation.swc** and **automation_agent.swc** files using the **`include-libraries`** option on the command line, ensure that you use the **`+=`** operator. This appends rather than overwrites the existing libraries that are included.

Note The SilkTest Flex Automation SDK is based on the Automation API for Flex. The SilkTest Automation SDK supports the same components in the same manner that the Automation API for Flex supports them. For instance, when an application is compiled with automation code and successive .swf files are loaded, a memory leak occurs and the application runs out of memory eventually. The Flex Component Explorer sample application is affected by this issue. The workaround is to not compile the application .swf files that Explorer loads with automation libraries. For example, compile only the Explorer main application with automation libraries. Another alternative is to use the module loader instead of swfloader. For more information about using the Flex Automation API, see the *Adobe Flex Release Notes*.

For details about creating events and custom controls to support automated testing, refer to "Testing Flex Custom Controls" in the SilkTest online help.

2

Tutorial for Flex Application Testing

Introduction

This chapter guides you through the steps of testing an Adobe Flex application. Quality Assurance testers are the target audience for this chapter.

For information about which version of Flex to use and which browsers and operating environments are supported for testing, refer to the *Release Notes*.

What you will learn

This chapter contains the following sections:

Prerequisites for Testing Adobe Flex Applications

Before you launch an Adobe Flex application that runs as a local application for the first time, you must configure security settings for your local Flash Player. You must modify the Adobe specific security settings to enable the local application access to the file system.

To configure security settings for your local Flash player

1. Open the Flex Security Settings Page by clicking Flash Player Security Manager on http://demo.borland.com/flex/SilkTest2011/index.html..
2. Click **Always allow**.

3 From the **Edit Locations** drop-down menu, click **Add Location**.

4 Click **Browse for folder** and navigate to the folder where your local application is installed.

5 Click **Confirm** and then close the browser.

SilkTest Tutorial

This tutorial explains each of the steps involved in testing an Adobe Flex application with SilkTest. These procedures use the Component Explorer sample application. However, if you have another Flex application that you prefer to use, make sure that the steps in the previous chapter have been completed and then follow these steps.

The steps include:

- "Launching the Component Explorer"
- "Creating a New Project"
- "Configuring a Flex Application"
- "Recording a Testcase"
- "Replaying a Testcase"
- "Silk4J Quick Tour"

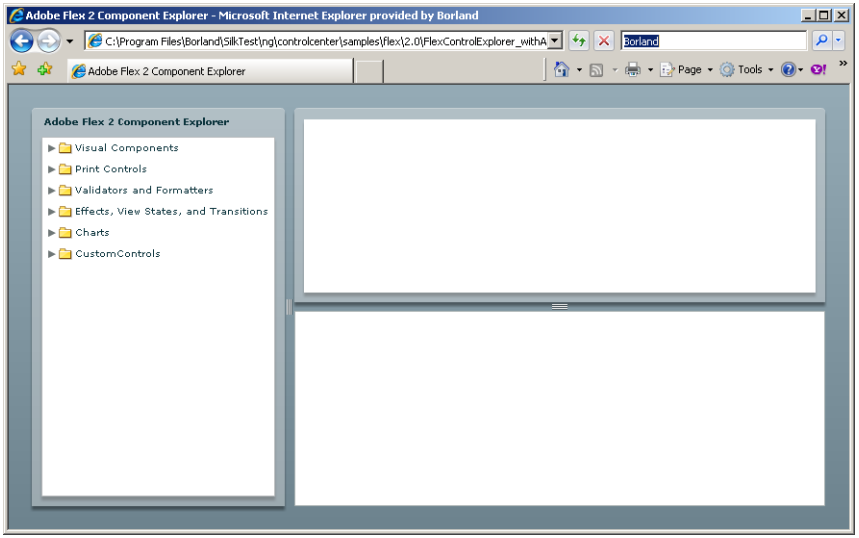
Launching the Component Explorer

Compiled with the Adobe Automation SDK and the SilkTest specific automation implementation, the Component Explorer is preconfigured for testing.

To launch the Component Explorer

- In Internet Explorer, open http://demo.borland.com/flex/SilkTest2011/3.5/Flex3TestApp_withAutomation/Flex3TestApp.html.

The application launches in Internet Explorer and looks similar to the following image:



Creating a New Project

To test the sample SilkTest Flex application, begin by creating a new project.

To create a project

1. In SilkTest, choose **File/New Project**, or click **Open Project/New Project** on the Basic workflow bar.
2. On the New Project dialog, under Rich Internet Applications, click **Adobe Flex**.
3. Click **OK**.
4. On the Create Project dialog, type the **Projectname** and **Description**.
5. Click **OK** to save your project in the default location, ***<SilkTest_Install_Directory>*\Projects**.

   If you do not want to save your project in the default location, click Browse and specify the folder in which you want to save your project.
6. SilkTest creates your project and displays nodes on the Files and Global tabs for the files and resources associated with this project.

Configuring a Flex Application

Configure the application to set up the environment that SilkTest will create each time you record or replay a testcase.

When you configure an application, SilkTest automatically creates a base state for the application. An application's base state is the known, stable state that you expect the application to be in before each test begins execution, and the state the application can be returned to after each test has ended execution.

Launch the Component Explorer before you perform this step.

1. In the SilkTest Basic Workflow bar, click **Configure Applications** on the Basic Workflow bar.

   The New Test Frame dialog box opens.

2. Double-click **Web Site Test Configuration**.

   The New Web Site Configuration page opens.

3. From the **Browser Type** list, select **Internet Explorer**.

   You can use Firefox to replay tests but not to record them.

4. Perform one of the following steps:

   - **Use existing browser** – Click this option button to use a browser window that is already open to configure the test. For example, if the Web page that you want to test is already displayed in the browser window, you might want to use this option.
   - **Start new browser** – Click this option button to start a new browser instance to configure the test. Then, in the Browse to URL text box specify the Web page to open.

5. Click **Finish**.

   The Choose name and folder of the new frame file page opens. SilkTest configures the recovery system and names the corresponding file frame.inc by default.

6. Navigate to the location in which you want to save the frame file.

7. In the **File name** text box, type the name for the frame file that contains the default base state and recovery system. Then, click **Save**.

   SilkTest automatically creates a base state for the application. When you configure an application, SilkTest adds an include file based on the technology or browser type that you enable to the Use files location in the Runtime Options dialog. For instance, if you configure an Adobe Flex application, a file named flex.inc is added.

SilkTest opens the include file. Record the testcase whenever you are ready.

Recording a Testcase

A testcase:

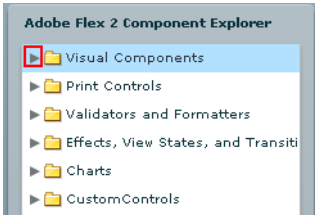
- Drives the application from the initial state to the state you want to test.
- Verifies that the actual state matches the expected (correct) state.
- Cleans up the application, in preparation for the next testcase, by undoing the steps performed in the first stage.

To record a testcase

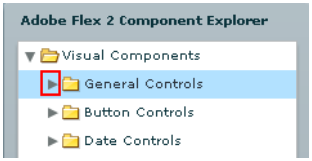
1. Click **Record Testcase** on the Basic Workflow bar.
2. In the Record Testcase dialog, type the name of your testcase in the **Testcase name** text box.

   Testcase names are not case sensitive; they can be any length and consist of any combination of alphabetic characters, numerals, and underscore characters.
3. From the **Application State** list box, select **DefaultBaseState** to have the built-in recovery system restore the default base state before the testcase begins executing.

   The testcase is recorded in the script file as: testcase testcase_name ().
4. Click **Start Recording**. SilkTest closes the Record Testcase dialog and displays the Flex Component Explorer application.
5. When the Record Status window opens, record the following scenario using the Flex Component Explorer application.
   a. Click the ▶ arrow next to the Visual Components tree element to expand the list.



b Click the ▶ arrow next to the General Controls tree element to expand the list.



c Click the SimpleAlert tree element.

d Point to the SimpleAlert tree and press Ctrl+Alt to add a verification to the script. You can add a verification for any of the information that appears.

The Verify Properties dialog box opens.

Check the Visible check box and then click **OK**.

A verification action is added to the script for the tree.

e In the Alert Control Example section, click the **Click Me** button and then click **OK** in the Hello World message box.

f Click the ▼ arrow next to the General Controls tree element to hide the list.

g Click the ▼ arrow next to the Visual Components tree element to hide the list.

6 In the Recording status window, click **Stop Recording**.

SilkTest opens the Record Testcase dialog, which contains the 4Test code that has been recorded for you.

7 Click **Paste to Editor**.

The Update Files dialog opens.

8 Choose **Paste testcase and update window declaration(s)** and then click **OK**.

9 Choose **File/Save**.

10 Specify the file name and location.

11 When SilkTest prompts you to add the file to the project, click **Yes**.

Replaying a Testcase

When you run a testcase, SilkTest interacts with the application by executing all the actions you specified in the testcase and testing whether all the features of the application performed as expected.

To run a testcase

1. Make sure that the testcase you want to run is in the active window.
2. Click **Run Testcase** on the Basic Workflow bar.
3. SilkTest displays the Run Testcase dialog, which lists all the testcases contained in the current script.
4. Select the testcase that you created.
5. To wait one second after each script line is executed, check the **Animated Run Mode (Slow-Motion)** check box.

   Typically, you will only use this check box if you want to watch the testcase run. For instance, if you want to demonstrate a testcase to someone else, you might want to check this check box.
6. Click **Run**. SilkTest runs the testcase and generates a results file. The results file describes whether the test passed or failed, and provides summary information.

Silk4J Quick Tour

Silk4J provides a Java run time library that includes test classes for all the classes that Silk4J supports for testing. This run time library is compatible with JUnit, which means you can leverage the JUnit infrastructure and run and create tests using JUnit. You can also use all available Java libraries in your testcases.

This quick tour explains how to use the Adobe Flex sample scripts provided with Silk4J. Use the sample script files in the sample application to view typical script configurations and testcase execution.

This quick tour includes:

- "Importing Silk4J Sample Scripts"
- "Running a Sample Testcase"

For a Getting Started tutorial for Silk4J, refer to the Silk4J User Guide.

Importing Silk4J Sample Scripts

Silk4J can use the SilkTest sample Flex applications and provides several Flex scripts that work with Silk4J. Import the Flex scripts to view how Silk4J works with Flex.

1. In the Eclipse workspace, choose **File > Import**. The Import wizard opens.

2 In the menu tree, click the plus sign (+) to expand the **General** folder and select **Existing Projects into Workspace**.

3 Click **Next**. The Import Projects dialog opens.

4 In the **Select root directory** text box, click **Browse**. The Browse For Folder dialog opens.

5 Navigate to the Documents and Settings\All Users\Shared Documents\SilkTest\samples\Silk4J folder.

   For Windows Vista and Windows 7.0 operating systems, the file location is Users\Public\Documents\SilkTest\samples\Silk4J\ rather than Documents and Settings\All Users\Shared Documents\SilkTest\samples.

6 Select the root directory for the sample and then click **OK**.

7 In the Projects section, select the project that you want to import and then click **Finish**. The sample project shows in the Package Explorer view.

Running a Sample Testcase

Use the sample testcases that Silk4J provides to view script contents and test execution results.

Before you run a testcase, start the SilkTest Open Agent.

1 Navigate to the Flex sample project.

2 Choose one of the following:

   a Right-click the package name to run all tests in the project.

   For example, right-click **com.borland.flex.store**.

   b Right-click the class name to run all tests for only that class.

   For example, right-click **FlexStoreTest.java** in the **com.borland.flex.store** package.

3 Choose **Run As > JUnit Test**. The test results display in the JUnit view as the test runs. If all tests pass, the status bar is green. If one or more tests fail, the status bar is red. You can click a failed test to display the stack trace in the Failure Trace area.

For detailed procedures about creating Silk4J scripts, refer to the Silk4J User Guide.