Borland
A MICRO FOCUS COMPANY

Silk Test 13.5

Silk4NET User Guide

Micro Focus
575 Anton Blvd., Suite 510
Costa Mesa, CA 92626

2012-09-19

Contents

Silk4NET 6
- Silk Test Product Suite 6
- Product Notification Service 6
- What's New In Silk4NET 7
  - Recording in Silk4NET 7
  - TrueLog with Silk4NET 7
  - Options User Interface in Silk4NET 7
  - Locator Spy in Silk4NET 7
  - Enhanced Silk4NET Help 7
  - Windows Internet Explorer Support 8
  - Mozilla Firefox Support 8
  - Google Chrome Support 8
  - Support for 64-bit Windows Internet Explorer Versions 8
  - Adobe Flex 4.6 Support 8
  - Microsoft .NET Framework 4.5 Support 8
  - Conveniently Switch Between Browsers 8
- Contacting Micro Focus 8
  - Information Needed by Micro Focus SupportLine 9

Getting Started with Silk4NET 10

Base State 11
- Modifying the Base State 11

Application Configuration 12
- Modifying an Application Configuration 12
- Application Configuration Errors 13

Working with Silk4NET Projects 14
- Creating a Silk4NET Project 14

Working with Silk4NET Tests 16
- Adding a Silk4NET Test to a Project 16
- Recording a Silk4NET Test 17
- Manually Creating a Silk4NET Test 18
- Adding a Locator to a Test Method Using the Locator Spy 19
- Running Silk4NET Tests 19
- Analyzing Test Results 20

Visual Execution Logs with TrueLog 21
- Enabling TrueLog 21
- Why is TrueLog Not Displaying Non-ASCII Characters Correctly? 22

Using Silk4NET with Team Foundation Server 23
- Executing Silk4NET Tests in TFS 23
- Locating TrueLog Files for Silk4NET Tests Executed with TFS 23

Setting Script Options 25
- Setting TrueLog Options 25
- Setting Recording Preferences 25
- Setting Browser Recording Options 26
- Setting Custom Attributes 26
- Setting Classes to Ignore 27
- Setting WPF Classes to Expose During Recording and Playback 28
- Setting Synchronization Options 28
- Setting Replay Options 29

Silk4NET Sample Tests 30

Dynamic Object Recognition 31
XPath Basic Concepts 31
Supported XPath Subset 32
XPath Samples 33
Troubleshooting Performance Issues for XPath 34
Locator Spy 35
Supported Attribute Types 35
Attributes for Adobe Flex Applications 35
Attributes for Java AWT/Swing Applications 35
Attributes for Java SWT Applications 36
Attributes for MSUIA Applications 36
Locator Attributes for Identifying Rumba Controls 37
Attributes for SAP Applications 37
Locator Attributes for Identifying Silverlight Controls 37
Attributes for Web Applications 38
Attributes for Windows API-based Client/Server Applications 39
Attributes for Windows Forms Applications 39
Attributes for Windows Presentation Foundation (WPF) Applications 39
Dynamic Locator Attributes 41
Enhancing Tests 42
Calling Windows DLLs 42
Calling a Windows DLL from Within a Script 42
DLL Function Declaration Syntax 42
Passing Arguments to DLL Functions 43
Passing String Arguments to DLL Functions 43
Aliasing a DLL Name 44
Conventions for Calling DLL Functions 44
Windows Accessibility 45
Using Accessibility 45
Enabling Accessibility 45
Dynamic Invoke 45
Text Recognition Support 46
Custom Controls 47
Supporting a Custom Control 48
Custom Controls Options 49
xBrowser Frequently Asked Questions 50
How do I Verify the Font Type Used for the Text of an Element? 50
What is the Difference Between textContents, innerText, and innerHtml? 50
I Configured innerText as a Custom Class Attribute, but it Is Not Used in Locators 51
What Should I Take Care Of When Creating Cross-Browser Scripts? 51
How Can I See Which Browser I Am Currently Using? 51
Which Locators are Best Suited for Stable Cross-Browser Testing? 51
Logging Output of My Application Contains Wrong Timestamps 52
My Test Script Hangs After Navigating to a New Page 52
Recorded an Incorrect Locator 52
Rectangles Around Elements in Windows Internet Explorer are Misplaced 52
Link.Select Does Not Set the Focus for a Newly Opened Window in Internet Explorer 52
DomClick(x, y) Is Not Working Like Click(x, y) 52
FileInputField.DomClick() Will Not Open the Dialog 52
The Move Mouse Setting Is Turned On but All Moves Are Not Recorded. Why Not? 53
I Need Some Functionality that Is Not Exposed by the xBrowser API. What Can I Do? 53
Why Are the Class and the Style Attributes Not Used in the Locator? 53
Dialog is Not Recognized During Replay 53
Why Do I Get an Invalidated-Handle Error When I Call the WaitForProperty Method? 53

Why Are Clicks Recorded Differently in Windows Internet Explorer 10? 54

Silk4NET

Silk4NET is the Silk Test plug-in for Microsoft Visual Studio. Silk4NET enables you to efficiently create and manage functional, regression, and localization tests directly in Visual Studio. With Silk4NET, you can perform the following tasks within Visual studio:

- Develop tests using Visual Basic .NET.
- Develop tests using C#.
- Run tests as a part of a test plan in the Microsoft test environment.
- Run tests as a part of as a part of your build process.
- View test results.

Silk4NET supports the testing of a broad set of application technologies, including AJAX and Web 2.0, RCP, WPF, Windows Forms, and Win32. Designed for realizing automation benefits even when applied to complex tests, Silk4NET brings true test automation capability directly to the developer's preferred environment and lets you easily cope with changes made in the test application.

Additionally, the powerful testing framework of Silk4NET enables high reusability of tests across multiple test projects, which further increases the achievable Return On Investment (ROI). With less time spent on building and maintaining testing suites, your QA staff can expand test coverage and optimize application quality.

Silk Test Product Suite

The Silk Test product suite includes the following components:

- Silk Test Workbench – Silk Test Workbench is the new, native quality testing environment that offers .NET scripting for power users and innovative storyboard-based visual tests to make testing more accessible.
- Silk4NET – The Silk4NET Visual Studio plug-in enables you to create Visual Basic or C# test scripts directly in Visual Studio.
- Silk4J – The Silk4J Eclipse plug-in enables you to create Java-based test scripts directly in your Eclipse environment.
- Silk Test Recorder – Silk Test Recorder enables you to record and replay tests using a GUI and then export those tests to Silk Test Classic, Silk4J, or Silk4NET.
- Silk Test Classic – Silk Test Classic is the traditional, 4Test Silk Test product.
- Silk Test Agents – The Silk Test Agent is the software process that translates the commands in your tests into GUI-specific commands. In other words, the Agent drives and monitors the application you are testing. One Agent can run locally on the host machine. In a networked environment, any number of Agents can run on remote machines.

The product suite that you install determines which components are available. To install all components, choose the complete install option. To install all components with the exception of Silk Test Classic, choose the standard install option.

Product Notification Service

The product notification service is an application that runs in your system tray and allows you to find out if updates are available for Silk Test. It also provides a link for you to click to navigate to the updates.

Running the Service

In the system tray, click the update notification icon and the Product Notification Service application opens.

Installed Version	Provides the version number of the currently installed Silk Test application.
Update Version	Provides a link and the version number of the next minor update, if one is available.
New Version	Provides a link and the version number of the next full release, if one is available.
Settings	Click the **Settings** button to open the **Settings** window. Select if and how often you want the notification service to check for updates.

What's New In Silk4NET

This section lists the significant enhancements and changes that were made for Silk4NET.

Recording in Silk4NET

You can now record Silk4NET tests in Visual Studio without starting the Silk Test Recorder. When you create a new Silk4NET project or when you add a new Silk4NET test to an existing Silk4NET project in Visual Studio, you can select to record the test. The recorded tests are now automatically parsed into a new script file when you finish recording.

TrueLog with Silk4NET

When you are working with Silk4NET, you can now use TrueLog to create visual execution logs during the execution of Silk4NET tests. The TrueLog file is created in the working directory of the process that executed the Silk4NET tests. When the Silk4NET test execution is complete, the new **Playback Complete** dialog box opens, and you can choose to review the TrueLog for the completed test.

Options User Interface in Silk4NET

You can now set the options for Silk4NET projects in Visual Studio. For example, you can set the base state for the application under test, the application configuration, and other options. All options for a Silk4NET are stored in a project-specific configuration file.

Locator Spy in Silk4NET

You can now use the **Locator Spy** in Silk4NET to capture the locators of the controls in your application. You can then paste the captured locators into your test scripts. Additionally, you can manually edit the attributes of the locators in your test scripts and validate the changes in the **Locator Spy**. By using the **Locator Spy**, you ensure that the locators in your test methods are valid.

Enhanced Silk4NET Help

The *Silk4NET Help* now provides additional information to better assist you in testing the functionality of your applications. The following enhancements have been added to the *Silk4NET Help*:

- A language reference has been added to the Help. This reference describes the classes and methods in Visual Basic .NET and Visual C# that you can use to test your applications.
- The Help now includes an index and a search functionality, so that you can easily find the content that you require.
- The contents of the help have been adapted to reflect the workflow in Silk4NET.

Windows Internet Explorer Support

Silk Test now includes recording and playback support for applications running in:

- Windows Internet Explorer 10

Mozilla Firefox Support

Silk Test now includes playback support for applications running in:

- Mozilla Firefox 12
- Mozilla Firefox 13
- Mozilla Firefox 14

Google Chrome Support

Silk Test now includes playback support for applications running in:

- Google Chrome 19
- Google Chrome 20
- Google Chrome 21

Support for 64-bit Windows Internet Explorer Versions

Silk Test now supports 64-bit versions of Windows Internet Explorer.

Adobe Flex 4.6 Support

Silk Test now supports Adobe Flex 4.6 applications.

Microsoft .NET Framework 4.5 Support

Silk Test now supports applications developed in or running on Microsoft .NET Framework 4.5.

Conveniently Switch Between Browsers

For tests that use the Web, a new setting has been added to the application configuration allowing you to easily select which of your installed browsers you would like to use to replay a test.

Contacting Micro Focus

Micro Focus is committed to providing world-class technical support and consulting services. Micro Focus provides worldwide support, delivering timely, reliable service to ensure every customer's business success.

All customers who are under a maintenance and support contract, as well as prospective customers who are evaluating products are eligible for customer support. Our highly trained staff respond to your requests as quickly and professionally as possible.

Visit *http://supportline.microfocus.com/assistedservices.asp* to communicate directly with Micro Focus SupportLine to resolve your issues or email supportline@microfocus.com.

Visit Micro Focus SupportLine at *http://supportline.microfocus.com* for up-to-date support news and access to other support information. First time users may be required to register to the site.

Information Needed by Micro Focus SupportLine

When contacting Micro Focus SupportLine, please include the following information if possible. The more information you can give, the better Micro Focus SupportLine can help you.

- The name and version number of all products that you think might be causing an issue.
- Your computer make and model.
- System information such as operating system name and version, processors, and memory details.
- Any detailed description of the issue, including steps to reproduce the issue.
- Exact wording of any error messages involved.
- Your serial number.

To find out these numbers, look in the subject line and body of your Electronic Product Delivery Notice email that you received from Micro Focus.

Getting Started with Silk4NET

Perform the following actions to use Silk4NET:

1. Create a Silk4NET project.
2. Add Silk4NET tests to your project. A project can include any combination of recorded tests and manually scripted tests.
3. Execute the tests.
4. Analyze the test results.

Base State

An application's base state is the known, stable state that you expect the application to be in before each test case begins execution, and the state the application can be returned to after each test case has ended execution. This state may be the state of an application when it is first started.

When you create a class for a Web application or standard configuration, Silk4NET automatically creates a base state.

Base states are important because they ensure the integrity of your tests. By guaranteeing that each test case can start from a stable base state, you can be assured that an error in one test case does not cause subsequent test cases to fail.

Silk4NET automatically ensures that your application is at its base state during the following stages:

- Before a test runs
- During the execution of a test
- After a test completes successfully

Note: Silk4NET stores the base state and any Silk4NET options in the `config.silk4net` configuration file. Silk4NET creates such a file for each Silk4NET project.

Modifying the Base State

You can change the executable location, working directory, locator, or URL of the base state if necessary. For example, if you want to launch tests from a production Web site that were previously tested on a testing Web site, change the base state URL and the tests will run in the new environment.

1. Click **Silk4NET** and choose **Edit Application Configurations**. The **Edit Application Configurations** dialog box opens and lists the existing application configurations.
2. Click **Edit Base State**. The **Edit Base State** dialog box opens.
3. In the **Executable** text box, type the executable name and file path of the application that you want to test.
   For example, you might type `C:\Program Files\Internet Explorer\IEXPLORE.EXE` to specify Internet Explorer.
4. To use a command line pattern in combination with the executable file, in the **Command Line Arguments** text box, type the command line pattern.
5. If the application that you want to test depends on a supplemental directory, specify a directory in the **Working Directory** text box.

   For example, if you use a batch file to start a Java application, the batch file may reference a JAR file that relies on a relative path. In this case, specify a working directory to reconcile the relative path.
6. In the **Locator** text box, type the XPath locator string that identifies the main window of the application.
7. If you are testing a Web site, in the **Url** text box, type the Web address for the Web page to launch when a test begins.
8. Click **OK**.

Application Configuration

An application configuration defines how Silk4NET connects to the application that you want to test. Silk4NET automatically creates an application configuration when you create the base state. However, at times, you might need to modify, remove, or add an additional application configuration. For example, if you are testing an application that modifies a database and you use a database viewer tool to verify the database contents, you must add an additional application configuration for the database viewer tool.

An application configuration includes the:

- Executable pattern

  All processes that match this pattern are enabled for testing. For example, the executable pattern for Notepad is `\*notepad.exe`. All processes whose executable is named `notepad.exe` and that are located in any arbitrary directory are enabled
- Command line pattern

  The command line pattern is an additional pattern that is used to constrain the process that is enabled for testing by matching parts of the command line arguments (the part after the executable name). An application configuration that contains a command line pattern enables only processes for testing that match both the executable pattern and the command line pattern. If no command-line pattern is defined, all processes with the specified executable pattern are enabled. Using the command line is especially useful for Java applications because most Java programs run by using `javaw.exe`. This means that when you create an application configuration for a typical Java application, the executable pattern, `* \javaw.exe` is used, which matches any Java process. Use the command line pattern in such cases to ensure that only the application that you want is enabled for testing. For example, if the command line of the application ends with **com.example.MyMainClass** you might want to use ***com.example.MyMainClass** as the command line pattern.

Modifying an Application Configuration

An application configuration defines how Silk4NET connects to the application that you want to test. Silk4NET automatically creates an application configuration when you create the base state. However, at times, you might need to modify, remove, or add an additional application configuration. For example, if you are testing an application that modifies a database and you use a database viewer tool to verify the database contents, you must add an additional application configuration for the database viewer tool.

1. Click **Silk4NET** and choose **Edit Application Configurations**. The **Edit Application Configurations** dialog box opens and lists the existing application configurations.
2. To add an additional application configuration, click **Add**.

   The **New Application Configuration** wizard opens.

   a) Select the type of the application that you want to test and click **Next**.
   b) Click the configuration that corresponds with the application that you want to test.

      If the application that you want to test does not appear in the list, uncheck the **Hide processes without caption** check box. This option, checked by default, is used to filter only those applications that have captions.
   c) If you have selected to test a Web application, select whether you want to use an existing browser instance or start a new one.
   d) Click **Finish**. The executable pattern of the application is added to the list of application configurations in the **Edit Application Configurations** dialog box.
3. To remove an application configuration, click **Remove** next to the appropriate application configuration.

4. To edit an application configuration, click **Edit Base State**. The **Edit Base State** dialog box opens.
5. Click **OK**.

Application Configuration Errors

When the program cannot attach to an application, the following error message opens:
Failed to attach to application <Application Name>. For additional information, refer to the Help.

In this case, one or more of the issues listed in the following table may have caused the failure:

Issue	Reason	Solution
Time out	• The system is too slow. • The size of the memory of the system is too small.	Use a faster system or try to reduce the memory usage on your current system.
User Account Control (UAC) fails	You have no administrator rights on the system.	Log in with a user account that has administrator rights.
Command-line pattern	The command-line pattern is too specific. This issue occurs especially for Java. The replay may not work as intended.	Remove ambiguous commands from the pattern.

Working with Silk4NET Projects

This section describes how you can use Silk4NET projects.

A Silk4NET project contains all the resources needed to test the functionality of your applications by using Silk4NET.

Creating a Silk4NET Project

1. Click **Silk4NET** > **New Project** or **File** > **New** > **Project** . The **New Project** dialog box displays.
2. Under **Installed Templates**, click **Visual Basic** or **Visual C#**, and then double-click **Silk4NET Project**. The **Create a Silk4NET Test** dialog box opens.
3. Select how you want to create your Silk4NET test by clicking one of the following option buttons:

Record a Silk4NET test	Record actions and verifications against your application under test and generate a new test containing the recorded automation statements.
Create an empty Silk4NET test	Create an empty test that can be filled with automation statements later on.

4. Click **OK**. If you have selected to create an empty Silk4NET test, a new solution containing the Silk4NET project is created. Additionally, a Silk4NET test is created in the project with the following language-specific file name:
   - `UnitTest1.vb`
   - `UnitTest1.cs`
5. If you have selected to record a new Silk4NET test, the **New Application Configuration** wizard opens. Select the type of application that you want to test.
   - If you want to test a standard application, click **Standard Test Configuration**.
   - If you want to test a Web application, click **Web Site Test Configuration**.
6. Click **Next**.
7. If you have selected to test a standard application, the **New Standard Configuration** dialog box opens. Click the configuration that corresponds with the application that you want to test.

   If the application that you want to test does not appear in the list, uncheck the **Hide processes without caption** check box. This option, checked by default, is used to filter only those applications that have captions.
8. If you have selected to test a Web application, the **New Web Site Configuration** dialog box opens.Select the browser type that you want to test from the **Browser Type** list box.
   a) In the **Browser Instance** section, click one of the option buttons:

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

9. Click **Finish**. If you have selected an existing instance of Google Chrome, on which you want to replay a test method, Silk Test Recorder checks whether the automation support is included. If the automation support is not included, Silk Test Recorder informs you that Google Chrome has to be restarted. The application and the **Recording** dialog box open.

Note: You can also use the context menu in the **Solution Explorer** to add Silk4NET projects to an existing solution.

Working with Silk4NET Tests

Describes how you can use Silk4NET tests.

You can create new Silk4NET tests by recording user actions made against your AUT or by manually scripting your test classes and methods in Visual Basic or Visual C#.

Adding a Silk4NET Test to a Project

You can only add Silk4NET tests to an existing Silk4NET or Test project. If no Silk4NET or Test project exists, create a Silk4NET or Test project before you try to create a Silk4NET test.

1. Click **Silk4NET** > **New Test** or **Project** > **Add New Item** .

   Note: If your solution contains more than one Silk4NET projects, select the project to which you want to add the new test from the list in the **Project Selector**.

   The **Add New Item** dialog box opens.
2. Under **Installed Templates**, click one of the following:

   - If your project is a Visual Basic project, click **Common Items** > **Silk4NET Test**.
   - If your project is a Visual C# project, click **Visual C# Items** > **Silk4NET Test**.

   The **Create a Silk4NET Test** dialog box opens.
3. Select how you want to create your Silk4NET test by clicking one of the following option buttons:

Record a Silk4NET test	Record actions and verifications against your application under test and generate a new test containing the recorded automation statements.
Create an empty Silk4NET test	Create an empty test that can be filled with automation statements later on.

4. Click **OK**. If you have selected to create an empty Silk4NET test, a new solution containing the Silk4NET project is created. Additionally, a Silk4NET test is created in the project with the following language-specific file name:

   - `UnitTest1.vb`
   - `UnitTest1.cs`
5. If you have selected to record a new Silk4NET test, the **New Application Configuration** wizard opens. Select the type of application that you want to test.

   - If you want to test a standard application, click **Standard Test Configuration**.
   - If you want to test a Web application, click **Web Site Test Configuration**.
6. Click **Next**.
7. If you have selected to test a standard application, the **New Standard Configuration** dialog box opens. Click the configuration that corresponds with the application that you want to test.

   If the application that you want to test does not appear in the list, uncheck the **Hide processes without caption** check box. This option, checked by default, is used to filter only those applications that have captions.
8. If you have selected to test a Web application, the **New Web Site Configuration** dialog box opens.Select the browser type that you want to test from the **Browser Type** list box.

   a) In the **Browser Instance** section, click one of the option buttons:

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

9. Click **Finish**. If you have selected an existing instance of Google Chrome, on which you want to replay a test method, Silk Test Recorder checks whether the automation support is included. If the automation support is not included, Silk Test Recorder informs you that Google Chrome has to be restarted. The application and the **Recording** dialog box open.

If you have selected to record the test, the recorded test is added to your project. If you have selected to add an empty test, an empty Silk4NET test is added to your project.

Note: You can also use the context menu in the **Solution Explorer** to add Silk4NET tests to your Silk4NET or Test project.

Recording a Silk4NET Test

1. Click **Silk4NET** > **New Test** or **Project** > **Add New Item** .

   Note: If your solution contains more than one Silk4NET projects, select the project to which you want to add the new test from the list in the **Project Selector**.

   The **Add New Item** dialog box opens.
2. Under **Installed Templates**, click one of the following:

   - If your project is a Visual Basic project, click **Common Items** > **Silk4NET Test**.
   - If your project is a Visual C# project, click **Visual C# Items** > **Silk4NET Test**.

   The **Create a Silk4NET Test** dialog box opens.
3. Click **Record a Silk4NET test**.

Record a Silk4NET test	Record actions and verifications against your application under test and generate a new test containing the recorded automation statements.

   The **New Application Configuration** wizard opens.
4. Select the type of application that you want to test.

   - If you want to test a standard application, click **Standard Test Configuration**.
   - If you want to test a Web application, click **Web Site Test Configuration**.
5. Click **Next**.
6. If you have selected to test a standard application, the **New Standard Configuration** dialog box opens. Click the configuration that corresponds with the application that you want to test.

   If the application that you want to test does not appear in the list, uncheck the **Hide processes without caption** check box. This option, checked by default, is used to filter only those applications that have captions.
7. If you have selected to test a Web application, the **New Web Site Configuration** dialog box opens.Select the browser type that you want to test from the **Browser Type** list box.

   a) In the **Browser Instance** section, click one of the option buttons:

Use existing browser	Click this option button to use a browser that is already open. For example, if the Web page that you want to test is already displayed in a browser, you might want to use this option.

Start new browser	Click this option button to start a new browser instance when you configure the test. Then, in the **Browse to URL** text box, specify the Web page to open.

8. Click **Finish**. If you have selected an existing instance of Google Chrome, on which you want to replay a test method, Silk Test Recorder checks whether the automation support is included. If the automation support is not included, Silk Test Recorder informs you that Google Chrome has to be restarted. The application and the **Recording** dialog box open.
9. Perform the interactions, which you want to record, with your application under test.

   For additional information, refer to the *Silk Test Recorder Help*.
10. When you are finished with recording, click **Stop Recording** in the **Recording** dialog box.

The recorded interactions are added as a file to your project. The default file name of the generated file is `UnitTest<Index>.cs` or `UnitTest<Index>.vb`, depending on the default programming language of your project. For example, if you are recording the first test for a Visual Basic project, the name of the generated file is `UnitTest1.vb`

Note: You can also create a new project and record the new test into the new project.

Manually Creating a Silk4NET Test

1. Add a Silk4NET test to your project.
2. *Optional:* To add support for controls of a specific application technology, you must include an import statement at the beginning of the test that references the application technology namespace, as shown in the following examples:

```
'Visual Basic .NET
Imports SilkTest.Ntf.Wpf
Imports SilkTest.Ntf.XBrowser
Imports SilkTest.Ntf.Win32
```

```
//C#
using SilkTest.Ntf.Wpf;
using SilkTest.Ntf.XBrowser;
using SilkTest.Ntf.Win32;
```

3. Configure the base state of the test application. For example:

```
'Visual Basic .NET
Dim baseState = New BaseState("C:\\Windows\\system32\\notepad.exe",
"/Window[@caption='Untitled - Notepad']")
baseState.WorkingDirectory = "%USERPROFILE%"
baseState.Execute()
```

```
//C#
BaseState baseState = new BaseState("C:\\Windows\\system32\\notepad.exe",
 "/Window[@caption='Untitled - Notepad']");
baseState.WorkingDirectory = "%USERPROFILE%"; baseState.Execute();
```

   Note: The base state makes sure that the application that you want to test is running and in the foreground. This ensures that tests will always start with the same application state, which makes them more reliable. In order to use the base state, it is necessary to specify what the main window looks like and how to launch the application that you want to test if it is not running. Creating a base state is optional. However, it is recommended as a best practice.
4. Add test classes and methods that test the desired functionality of the test application.

Adding a Locator to a Test Method Using the Locator Spy

Manually capture a locator using the **Locator Spy** and copy the locator to the test method. For instance, you can identify the caption or the XPath locator string for GUI objects using the **Locator Spy**. Then, copy the relevant locator strings and attributes into the test methods in your scripts.

1. Open the test class that you want to modify.
2. Click **Silk4NET** > **Record Locators**. The **Locator Spy** opens.
3. Position the mouse over the object that you want to record. The related locator string shows in the **Selected Locator** text box.
4. Press **Ctrl+Alt** to capture the object.

   Note: Press **Ctrl+Shift** to capture the object if you specified the alternative record break key sequence on the **General Recording Options** page of the **Script Options** dialog box.
5. *Optional:* Click **Show additional locator attributes** to display any related attributes in the **Locator Attribute** table.
6. *Optional:* You can replace a recorded locator attribute with another locator attribute from the **Locator Attribute** table.

   For example, your recorded locator might look like the following:

   ```
   /Window[@caption='MyApp']//Control[@id='table1']
   ```

   If you have a caption `Files` listed in the **Locator Attribute** table, you can manually change the locator to the following:

   ```
   /Window[@caption='MyApp']//Control[@caption='Files']
   ```

   The new locator displays in the **Selected Locator** text box.
7. To copy the locator, click **Copy Locator to Clipboard**.

   In the **Selected Locator** text box, you can also mark the portion of the locator string that you want to copy, and then you can right-click the marked text and click **Copy**.
8. In the script, position your cursor to the location to which you want to paste the recorded locator.

   For example, position your cursor in the appropriate parameter of a `Find` method in the script.

   The test method, into which you want to paste the locator, must use a method that can take a locator as a parameter. Using the **Locator Spy** ensures that the locator is valid.
9. Copy the locator to the test case or to the Clipboard.
10. Click **Close**.

Running Silk4NET Tests

1. In Visual Studio, click **File** > **Open** > **Project/Solution**. The **Open Project** dialog box displays.
2. Browse to the project file of the project which includes the test that you want to run.

   The project file is a `.sln` file, for example `Silk4NETProject1.sln`.
3. Double-click on the project file. The test files that are included in the project file open in the main window of Visual Studio.
4. In the toolbar, click **Run All Tests in Solution**. All tests in the open project are executed. When the execution is finished, the **Playback Complete** dialog box opens.
5. Click **Explore Results** to examine the results of the tests with TrueLog or click **OK** to close the dialog box.

Analyzing Test Results

1. Run a Silk4NET test. When the execution is finished, the **Playback Complete** dialog box opens.
2. Click **Explore Results** to examine the results of the tests with TrueLog. Silk TrueLog Explorer opens.
3. Click through the results in Silk TrueLog Explorer.
   Silk TrueLog Explorer captures a screenshot whenever a test fails.

Visual Execution Logs with TrueLog

TrueLog is a powerful technology that simplifies root cause analysis of test case failures through visual verification. The results of test runs can be examined in TrueLog Explorer. When an error occurs during a test run, TrueLog enables you to easily locate the line in your script that generated the error so that the issue can be resolved.

Note: TrueLog is supported only for one local or remote agent in a script. When you use multiple agents, for example when testing an application on one machine, and that application writes data to a database on another machine, a TrueLog is written only for the first agent that was used in the script. When you are using a remote agent, the TrueLog file is also written on the remote machine.

For additional information about TrueLog Explorer, refer to the *Silk TrueLog Explorer User Guide*, located in **Start** > **Programs** > **Silk** > **Silk Test** > **Documentation**.

You can enable TrueLog in Silk4NET to create visual execution logs during the execution of Silk4NET tests. The TrueLog file is created in the working directory of the process that executed the Silk4NET tests.

The default setting is that screenshots are only created when an error occurs in the script, and only test cases with errors are logged.

Enabling TrueLog

For new Silk4NET scripts, TrueLog is enabled by default. To enable TrueLog for existing Silk4NET scripts, which are using the Visual Studio Unit Testing Framework, you have to replace the `TestClass` attribute of all test classes in the script with the `SilkTestClass` attribute.

To enable TrueLog:

1. Open the script that contains the test class for you want to enable TrueLog.
2. Add the `SilkTestClass` attribute to the test class.

The TrueLog is created in the `TestResults` sub-directory of the directory, in which the Visual Studio solution file and the results of the Visual Studio Unit Testing Framework are located. The Visual Studio solution file is the file in which the Silk4NET scripts are located. When the Silk4NET test execution is complete, a dialog box opens, and you can click **Explore Results** to review the TrueLog for the completed test.

Examples

To enable TrueLog for a class in a Visual Basic script, use the following code:

```
<SilkTestClass()> Public Class MyTestClass
  <TestMethod()> Public Sub MyTest()
    ' my test code
  End Sub
End Sub
```

To enable TrueLog for a class in a C# script, use the following code:

```
[SilkTestClass]
public class MyTestClass {
  [TestMethod]
  public void MyTest() {
    // my test code
  }
}
```

Why is TrueLog Not Displaying Non-ASCII Characters Correctly?

TrueLog Explorer is a MBCS-based application, meaning that to be displayed correctly, every string must be encoded in MBCS format. When TrueLog Explorer visualizes and customizes data, many string conversion operations may be involved before the data is displayed.

Sometimes when testing UTF-8 encoded Web sites, data containing characters cannot be converted to the active Windows system code page. In such cases, TrueLog Explorer will replace the non-convertible characters, which are the non-ASCII characters, with a configurable replacement character, which usually is '?'.

To enable TrueLog Explorer to accurately display non-ASCII characters, set the system code page to the appropriate language, for example Japanese.

Using Silk4NET with Team Foundation Server

This section describes how you can use Visual Studio Team Foundation Server (TFS) to execute Silk4NET tests.

For information about new features, supported platforms and versions, known issues, and work-arounds, refer to the *Silk Test Release Notes*, available from *http://supportline.microfocus.com/productdoc.aspx*.

Executing Silk4NET Tests in TFS

Note: For detailed information on the steps in this task that describe functionality of TFS or Visual Studio, refer to the documentation of these products.

You can use a TFS to execute Silk4NET tests:

1. In Visual Studio, open the **Team Explorer View** and click **Connect to Team Project** to connect to the TFS.
2. In the **Team Explorer View**, add your Silk4NET project to the TFS.
3. *Optional:* Add additional Silk4NET tests to your Silk4NET project.
4. In the **Team Explorer View**, check in the tests into the TFS.
5. Right-click on the solution that includes your Silk4NET project and click **Add** > **New Item** > **Test** > **Test Settings File** to create a new test settings file.
6. Configure the test controller that you would like to use to execute the tests.
7. In the **Team Explorer View**, create a new build definition.
8. Add your test settings file to the build definition.
9. Configure the build definition so that automated tests for your Silk4NET test project assembly are run after the build.
10. Follow the instructions in *How to: Set Up Your Test Agent to Run Tests that Interact with the Desktop* to enable the interaction between Silk4NET and the AUT.
11. In the **Team Explorer View**, run the build definition to execute the Silk4NET tests.
12. *Optional:* Analyze the TrueLog files.

Locating TrueLog Files for Silk4NET Tests Executed with TFS

When you execute Silk4NET tests with TFS, the **Playback Complete** dialog box is not displayed after the execution is finished, and the TrueLog files for the tests are not written on your local machine. Locate the generated TrueLog files to analyze the results of the Silk4NET tests that you have executed with TFS.

1. Connect to the Test Agent, which is the machine on which TFS has executed the tests.
2. Go to the default location, `<system drive>/Builds`, to access the TrueLog file.

   For example `C:/Builds`.
3. If the *Builds* directory does not exist in the default location on the Test Agent, the default value has been changed. Use the following steps to find out where the TrueLog files are written:

a) Go to the machine on which the TFS runs.
b) Click **Start** > **All Programs** > **Microsoft Visual Studio Team Foundation Server <version number>** > **Team Foundation Server Administrative Console**. The **Team Foundation Server Administrative Console** opens.
c) Click **Build Configuration**.
d) Click **Properties** under the name of the Build Agent machine. The Build Agent Properties dialog box opens. The directory into which the TrueLog files are written is defined in the **Working Directory** text field.

Setting Script Options

Specify script options for recording, browser and custom attributes, classes to ignore, synchronization, and the replay mode.

Note: For each Silk4NET project, Silk4NET creates the `config.Silk4net` configuration file. Silk4NET stores the base state of the application under test and all options in this file. The options are then used during replay.

Setting TrueLog Options

Enable TrueLogs to capture bitmaps and to log information for Silk4NET.

Logging bitmaps and controls in TrueLogs may adversely affect the performance of Silk4NET. Because capturing bitmaps and logging information can result in large TrueLog files, you may want to log test cases with errors only and then adjust the TrueLog options for test cases where more information is needed.

The results of test runs can be examined in TrueLog Explorer. For additional information about TrueLog Explorer, refer to the *Silk TrueLog Explorer User Guide*, located in **Start** > **Programs** > **Silk** > **Silk Test** > **Documentation**.

To enable TrueLog and customize the information that the TrueLog collects for Silk4NET, perform the following steps:

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **TrueLog** tab.
3. In the **Logging** area, check the **Enable TrueLog** check box.
   - Click **All testcases** to log activity for all test cases, both successful and failed. This is the default setting.
   - Click **Testcases with errors** to log activity for only those test cases with errors.
4. In the **TrueLog file** field, type the path to and name of the TrueLog file, or click **Browse** and select the file.

   This path is relative to the machine on which the agent is running. The default path is the path of the Silk4J project folder, and the default name is the name of the suite class, with a `.xlg` suffix.

   Note: If you provide a local or remote path in this field, the path cannot be validated until script execution time.
5. Select the **Screenshot mode**.

   Default is **None**.
6. *Optional:* Set the **Delay**.

   This delay gives Windows time to draw the application window before a bitmap is taken. You can try to add a delay if your application is not drawn properly in the captured bitmaps.
7. Click **OK**.

Setting Recording Preferences

Set the shortcut key combination to pause recording and specify whether absolute values and mouse move actions are recorded.

Note: All the following settings are optional. Change these settings if they will improve the quality of your test methods.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **Recording** tab.
3. To set `Ctrl+Shift` as the shortcut key combination to use to pause recording, check the **OPT_ALTERNATE_RECORD_BREAK** check box.

   By default, `Ctrl+Alt` is the shortcut key combination.
4. To record absolute values for scroll events, check the **OPT_RECORD_SCROLLBAR_ABSOLUT** check box.
5. To record mouse move actions, check the **OPT_RECORD_MOUSEMOVES** check box.
6. If you record mouse move actions, in the **OPT_RECORD_MOUSEMOVE_DELAY** text box, specify how many milliseconds the mouse has to be motionless before a `MouseMove` is recorded

   By default this value is set to 200.
7. To record `TextClick` actions instead of `Click` actions, on those objects where `TextClick` actions usually are preferable to `Click` actions, check the **OPT_RECORD_TEXT_CLICK** check box.
8. Click **OK**.

Setting Browser Recording Options

Specify browser attributes to ignore while recording and whether to record native user input instead of DOM functions.

Note: All the following settings are optional. Change these settings if they will improve the quality of your test methods.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **Browser** tab.
3. In the **Locator attribute name exclude list** grid, type the attribute names to ignore while recording. For example, if you do not want to record attributes named `height`, add the `height` attribute name to the grid.

   Separate attribute names with a comma.
4. In the **Locator attribute value exclude list** grid, type the attribute values to ignore while recording. For example, if you do not want to record attributes assigned the value of `x-auto`, add `x-auto` to the grid.

   Separate attribute values with a comma.
5. To record native user input instead of DOM functions, check the **OPT_XBROWSER_RECORD_LOWLEVEL** check box.

   For example, to record `Click` instead of `DomClick` and `TypeKeys` instead of `SetText`, check this check box.

   If your application uses a plug-in or AJAX, use native user input. If your application does not use a plug-in or AJAX, we recommend using high-level DOM functions, which do not require the browser to be focused or active during playback. As a result, tests that use DOM functions are faster and more reliable.
6. Click **OK**.

Setting Custom Attributes

Silk4NET includes a sophisticated locator generator mechanism that guarantees locators are unique at the time of recording and are easy to maintain. Depending on your application and the frameworks that you

use, you might want to modify the default settings to achieve the best results. You can use any property that is available in the respective technology as a custom attribute given that they are either numbers (integers, doubles), strings, item identifiers, or enumeration values.

A well-defined locator relies on attributes that change infrequently and therefore requires less maintenance. Using a custom attribute is more reliable than other attributes like caption or index, since a caption will change when you translate the application into another language, and the index might change when another object is added.

In xBrowser, WPF, Java SWT, and Swing applications, you can also retrieve arbitrary properties (such as a WPFButton that defines *myCustomProperty*) and then use those properties as custom attributes. To achieve optimal results, add a custom automation ID to the elements that you want to interact with in your test. In Web applications, you can add an attribute to the element that you want to interact with, such as `<div myAutomationId= "my unique element name" />`. Or, in Java SWT, the developer implementing the GUI can define an attribute (for example `testAutomationId`) for a widget that uniquely identifies the widget in the application. A tester can then add that attribute to the list of custom attributes (in this case, `testAutomationId`), and can identify controls by that unique ID. This approach can eliminate the maintenance associated with locator changes.

If multiple objects share the same attribute value, such as a caption, Silk4NET tries to make the locator unique by combining multiple available attributes with the "and" operation and thus further narrowing down the list of matching objects to a single object. Should that fail, an index is appended. Meaning the locator looks for the *nth* control with the caption *xyz*.

If more than one object is assigned the same custom attribute value, all the objects with that value will return when you call the custom attribute. For example, if you assign the unique ID, `loginName` to two different text fields, both fields will return when you call the `loginName` attribute.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **Custom Attributes** tab.
3. From the **Select a tech domain** list box, select the technology domain for the application that you are testing.

   Note: You cannot set custom attributes for Flex or Windows API-based client/server (Win32) applications.
4. Add the attributes that you want to use to the list.

   If custom attributes are available, the locator generator uses these attributes before any other attribute. The order of the list also represents the priority in which the attributes are used by the locator generator. If the attributes that you specify are not available for the objects that you select, Silk4NET uses the default attributes for the application that you are testing.

   Separate attribute names with a comma.

   Note: To include custom attributes in a Web application, add them to the html tag. For example type, `<input type='button' bcauid='abc' value='click me' />` to add an attribute called `bcauid`.

   Note: To include custom attributes in a Java SWT application, use the `org.swt.widgets.Widget.setData(String key, Object value)` method.

   Note: To include custom attributes in a Swing application, use the `SetClientProperty("propertyName", "propertyValue")` method.
5. Click **OK**.

Setting Classes to Ignore

Specify the names of any classes that you want to ignore during recording and playback.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **Transparent Classes** tab.
3. In the **Transparent classes** grid, type the name of the class that you want to ignore during recording and playback.

   Separate class names with a comma.
4. Click **OK**.

Setting WPF Classes to Expose During Recording and Playback

Specify the names of any WPF classes that you want to expose during recording and playback. For example, if a custom class called *MyGrid* derives from the WPF `Grid` class, the objects of the *MyGrid* custom class are not available for recording and playback. `Grid` objects are not available for recording and playback because the `Grid` class is not relevant for functional testing since it exists only for layout purposes. As a result, `Grid` objects are not exposed by default. In order to use custom classes that are based on classes that are not relevant to functional testing, add the custom class, in this case *MyGrid*, to the **OPT_WPF_CUSTOM_CLASSES** option. Then you can record, playback, find, verify properties, and perform any other supported actions for the specified classes.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **WPF** tab.
3. In the **Custom WPF class names** grid, type the name of the class that you want to expose during recording and playback.

   Separate class names with a comma.
4. Click **OK**.

Setting Synchronization Options

Specify the synchronization and timeout values for Web applications.

Note: All the following settings are optional. Change these settings if they will improve the quality of your test methods.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **Synchronization** tab.
3. To specify the synchronization algorithm for the ready state of a web application, from the **OPT_XBROWSER_SYNC_MODE** list box, choose an option.

   The synchronization algorithm configures the waiting period for the ready state of an invoke call. By default, this value is set to **AJAX**.
4. In the **Synchronization exclude list** text box, type the entire URL or a fragment of the URL for any service or Web page that you want to exclude.

   Some AJAX frameworks or browser applications use special HTTP requests, which are permanently open in order to retrieve asynchronous data from the server. These requests may let the synchronization hang until the specified synchronization timeout expires. To prevent this situation, either use the HTML synchronization mode or specify the URL of the problematic request in the **Synchronization exclude list** setting.

   For example, if your Web application uses a widget that displays the server time by polling data from the client, permanent traffic is sent to the server for this widget. To exclude this service from the synchronization, determine what the service URL is and enter it in the exclusion list.

   For example, you might type:

- http://example.com/syncsample/timeService
- timeService
- UICallBackServiceHandler

Separate multiple entries with a comma.

Note: If your application uses only one service, and you want to disable that service for testing, you must use the HTML synchronization mode rather than adding the service URL to the exclusion list.

5. To specify the maximum time, in milliseconds, to wait for an object to be ready, type a value in the **OPT_SYNC_TIMEOUT** text box.

   By default, this value is set to **300000**.
6. To specify the time, in milliseconds, to wait for an object to be resolved during replay, type a value in the **OPT_WAIT_RESOLVE_OBJDEF** text box.

   By default, this value is set to **5000**.
7. To specify the time, in milliseconds, to wait before the agent attempts to resolve an object again, type a value in the **OPT_WAIT_RESOLVE_OBJDEF_RETRY** text box.

   By default, this value is set to **500**.
8. Click **OK**.

Setting Replay Options

Specify whether you want to ensure that the object that you want to test is active and whether to override the default replay mode. The replay mode defines whether controls are replayed with the mouse and keyboard or with the API. Use the default mode to deliver the most reliable results. When you select another mode, all controls use the selected mode.

1. Click **Silk4NET** and choose **Edit Options**. The **Script Options** dialog box opens.
2. Click the **Replay** tab. The **Replay Options** page displays.
3. From the **OPT_REPLAY_MODE** list box, select one of the following options:

   - **Default** – Use this mode for the most reliable results. By default, each control uses either the mouse and keyboard (low level) or API (high level) modes. With the default mode, each control uses the best method for the control type.
   - **High level** – Use this mode to replay each control using the API.
   - **Low level** – Use this mode to replay each control using the mouse and keyboard.
4. To ensure that the object that you want to test is active, check the **OPT_ENSURE_ACTIVE_OBJDEF** check box.
5. Click **OK**.

Silk4NET Sample Tests

The Silk4NET sample tests are packaged in a Visual Studio solution which you can open and view as well as run against the Silk Test sample applications.

Download and install the sample applications from *http://supportline.microfocus.com/websync/SilkTest.aspx*. After you have installed the sample applications, click **Start** > **Programs** > **Silk** > **Silk Test** > **Samples** > **Silk4NET Samples** and select the folder that corresponds to the Visual Studio version that you are using. Open the Visual Studio solution file, `Silk4NET Samples.sln`, of the Silk4NET sample tests.

In addition to the installed Silk4NET sample applications, the set of Silk4NET sample tests includes several tests for the following Silk Test Web-based sample applications:

Insurance Co. Web site	*http://demo.borland.com/InsuranceWebExtJS/*
Green Mountain Outpost Web	*http://demo.borland.com/gmopost/*

Dynamic Object Recognition

Dynamic object recognition enables you to write test methods that use XPath queries to find and identify objects. Dynamic object recognition uses a `Find` or `FindAll` method to identify an object in a test method. For example, the following query finds the first button with the caption "ok" that is a child of a given window:

```
Dim okButton = window.Find("//PushButton[@caption=ok]")
```

Examples of the types of test environments where dynamic object recognition works well include:

- In any application environment where the graphical user interface is undergoing changes.

  For example, to test the **Check Me** check box in a dialog that belongs to a menu where the menu and the dialog name are changing, using dynamic object recognition enables you to test the check box without concern for what the menu and dialog name are called. You can then verify the check box name, dialog name, and menu name to ensure that you have tested the correct component.
- In a Web application that includes dynamic tables or text.

  For example, to test a table that displays only when the user points to a certain item on the Web page, use dynamic object recognition to have the test method locate the table without regard for which part of the page needs to be clicked in order for the table to display.
- In an Eclipse environment that uses views.

  For example, to test an Eclipse environment that includes a view component, use dynamic object recognition to identify the view without regard to the hierarchy of objects that need to open prior to the view.

Benefits of Using Dynamic Object Recognition

The benefits of using dynamic object recognition include:

- Dynamic object recognition uses a subset of the XPath query language, which is a common XML-based language defined by the World Wide Web Consortium, W3C.
- Dynamic object recognition requires a single object rather than a repository of objects for the application that you are testing. Using XPath queries, a test case can locate an object using a `Find` command followed by a supported XPath construct.

XPath Basic Concepts

Silk4NET supports a subset of the XPath query language. For additional information about XPath, see *http://www.w3.org/TR/xpath20/*.

Basic Concepts

XPath expressions rely on the *current context*, the position of the object in the hierarchy on which the `Find` method was invoked. All XPath expressions depend on this position, much like a file system. For example:

- `"//Shell"` finds all shells in any hierarchy relative to the current object.
- `"Shell"` finds all shells that are direct children of the current object.

Additionally, some XPath expressions are context sensitive. For example, `myWindow.Find(xPath)` makes `myWindow` the current context.

Supported XPath Subset

Silk4NET supports a subset of the XPath query language. Use a `FindAll` or a `Find` command followed by a supported construct to create a test case.

The following table lists the constructs that Silk4NET supports.

Supported XPath Construct	Sample	Description
Attribute	`MenuItem[@caption='abc']`	Finds all menu items with the given caption attribute in their object definition that are children of the current context. The following attributes are supported: caption (without caption index), priorlabel (without index), windowid.
Index	`MenuItem[1]`	Finds the first menu item that is a child of the current context. Indices are 1-based in XPath.
Logical Operators: and, or, not, =, !=	`MenuItem[not(@caption='a' or @windowid!='b') and @priorlabel='p']`	
.	`TestApplication.Find("//Dialog[@caption='Check Box']/./.")`	Finds the context on which the `Find` command was executed. For instance, the sample could have been typed as `TestApplication.Find("//Dialog[@caption='Check Box']")`.
..	`Desktop.Find("//PushButton[@caption='Previous']/../PushButton[@caption='Ok']")`	Finds the parent of an object. For instance, the sample finds a PushButton with the caption "Ok" that has a sibling PushButton with the caption "Previous."
/	`/Shell`	Finds all shells that are direct children of the current object.  **Note:** "`/Shell`" is equivalent to "`Shell`".
/	`/Shell/MenuItem`	Finds all menu items that are a child of the current object.
//	`//Shell`	Finds all shells in any hierarchy relative to the current object.
//	`//Shell//MenuItem`	Finds all menu items that are direct or indirect children of a Shell that is a direct child of the current object.
//	`//MenuItem`	Finds all menu items that are direct or indirect children of the current context.

Supported XPath Construct	Sample	Description
*	`*[@caption='c']`	Finds all objects with the given caption that are a direct child of the current context.
*	`//MenuItem/*/Shell`	Finds all shells that are a grandchild of a menu item.

The following table lists the XPath constructs that Silk4NET does not support.

Unsupported XPath Construct	Example
Comparing two attributes with each other	`PushButton[@caption = @windowid]`
An attribute name on the right side is not supported. An attribute name must be on the left side.	`PushButton['abc' = @caption]`
Combining multiple XPath expressions with 'and' or 'or'.	`PushButton [@caption = 'abc'] or .// Checkbox`
More than one set of attribute brackets	`PushButton[@caption = 'abc] [@windowid = '123']` (use `PushButton [@caption = 'abc and @windowid = '123']` instead)
More than one set of index brackets	`PushButton[1][2]`

Any construct that does not explicitly specify a class or the class wildcard, such as including a wildcard as part of a class name	`//[@caption = 'abc']`
	(use `//*[@caption = 'abc']` instead)
	`"//*Button[@caption='abc']"`

XPath Samples

The following table lists sample XPath queries and explains the semantics for each query.

XPath String	Description
`desktop.Find("/Shell[@caption='SWT Test Application']")`	Finds the first top-level Shell with the given caption.
`desktop.Find("//MenuItem[@caption='Control']")`	Finds the MenuItem in any hierarchy with the given caption.
`myShell.Find("//MenuItem[@caption!='Control']")`	Finds a MenuItem in any child hierarchy of myShell that does not have the given caption.
`myShell.Find("Menu[@caption='Control']/MenuItem[@caption!='Control']")`	Looks for a specified MenuItem with the specified Menu as parent that has myShell as parent.
`myShell.Find("//MenuItem[@caption='Control' and @windowid='20']")`	Finds a MenuItem in any child hierarchy of myWindow with the given caption and windowid.
`myShell.Find("//MenuItem[@caption='Control' or @windowid='20']")`	Finds a MenuItem in any child hierarchy of myWindow with the given caption or windowid.

XPath String	Description
`desktop.FindAll("/Shell[2]/*/PushButton")`	Finds all PushButtons that have an arbitrary parent that has the second top-level shell as parent.
`desktop.FindAll("/Shell[2]//PushButton")`	Finds all PushButtons that use the second shell as direct or indirect parent.
`myBrowser.Find("//FlexApplication[1]//FlexButton[@caption='ok']")`	Looks up the first FlexButton within the first FlexApplication within the given browser.
`myBrowser.FindAll("//td[@class='abc*']//a[@class='xyz']")`	Finds all link elements with attribute class `xyz` that are direct or indirect children of `td` elements with attribute class `abc*`.

Troubleshooting Performance Issues for XPath

When testing applications with a complex object structure, for example complex web applications, you may encounter performance issues, or issues related to the reliability of your scripts. This topic describes how you can improve the performance of your scripts by using different locators than the ones that Silk4NET has automatically generated during recording.

Note: In general, we do not recommend using complex locators. Using complex locators might lead to a loss of reliability for your tests. Small changes in the structure of the tested application can break such a complex locator. Nevertheless, when the performance of your scripts is not satisfying, using more specific locators might result in tests with better performance.

The following is a sample element tree for the application MyApplication:

```
Root
  Node id=1
    Leaf id=2
    Leaf id=3
    Leaf id=4
    Leaf id=5
  Node id=6
    Node id=7
      Leaf id=8
      Leaf id=9
    Node id=9
      Leaf id=10
```

You can use one or more of the following optimizations to improve the performance of your scripts:

- If you want to locate an element in a complex object structure , search for the element in a specific part of the object structure, not in the entire object structure. For example, to find the element with the identifier 4 in the sample tree, if you have a query like `Root.Find("//Leaf[@id='4']")`, replace it with a query like `Root.Find("/Node[@id='1']/Leaf[@id='4']")`. The first query searches the entire element tree of the application for leafs with the identifier 4. The first leaf found is then returned. The second query searches only the first level nodes, which are the node with the identifier 1 and the node with the identifier 6, for the node with the identifier 1, and then searches in the subtree of the node with the identifier 1 for all leafs with the identifier 4.
- When you want to locate multiple items in the same hierarchy, first locate the hierarchy, and then locate the items in a loop. If you have a query like `Root.FindAll("/Node[@id='1']/Leaf")`, replace it with a loop like the following:

```
Public Sub Main()
  Dim node As TestObject

  node = _desktop.Find("//Node[@id='1']")
  For i As Integer = 1 To 4 Step 1
```

```
    node.Find("/Leaf[@id='"+i+"']")
  Next

End Sub
```

Locator Spy

Use the **Locator Spy** to identify the caption or the XPath locator string for GUI objects. You can copy the relevant XPath locator strings and attributes into methods in your scripts. Additionally, you can manually edit the attributes of the XPath locator strings in your test scripts and validate the changes in the **Locator Spy**. Using the **Locator Spy** ensures that the XPath query string is valid.

Supported Attribute Types

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests. If necessary, you can change the attribute type in one of the following ways:

- Manually typing another attribute type and value.
- Specifying another preference for the default attribute type by changing the **Preferred attribute list** values.

Attributes for Adobe Flex Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Flex applications include:

- automationName
- caption (similar to automationName)
- automationClassName (e.g. `FlexButton`)
- className (the full qualified name of the implementation class, e.g. `mx.controls.Button`)
- automationIndex (the index of the control in the view of the FlexAutomation, e.g. `index:1`)
- index (similar to automationIndex but without the prefix, e.g. `1`)
- id (the id of the control)
- windowId (similar to id)
- label (the label of the control)
- All dynamic locator attributes

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Attributes for Java AWT/Swing Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Java AWT/Swing include:

- caption

- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text input field, the caption of the closest label at the left side or above the control is used.
- name
- accessibleName
- *Swing only*: All custom object definition attributes set in the widget with `SetClientProperty("propertyName", "propertyValue")`

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

Attributes for Java SWT Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Java SWT include:

- caption
- all custom object definition attributes

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

Attributes for MSUIA Applications

Note: MSUIA is deprecated. For new tests, use the WPF technology domain.

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for MSUIA applications include:

- acceleratorkey
- accesskey
- automationid
- classname
- controltype
- frameworkid
- haskeyboardfocus
- helptext
- isenabled
- isoffscreen
- ispassword
- caption
- name
- nativewindowhandle
- orientation

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

Locator Attributes for Identifying Rumba Controls

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests. Supported attributes include:

caption	The text that the control displays.
priorlabel	Since input fields on a form normally have a label explaining the purpose of the input, the intention of **priorlabel** is to identify the text input field, **RumbaTextField**, by the text of its adjacent label field, **RumbaLabel**. If no preceding label is found in the same line of the text field, or if the label at the right side is closer to the text field than the left one, a label on the right side of the text field is used.
StartRow	This attribute is not recorded, but you can manually add it to the locator. Use **StartRow** to identify the text input field, **RumbaTextField**, that starts at this row.
StartColumn	This attribute is not recorded, but you can manually add it to the locator. Use **StartColumn** to identify the text input field, **RumbaTextField**, that starts at this column.
All dynamic locator attributes.	For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and `*`.

Attributes for SAP Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for SAP include:

- automationId
- caption

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and `*`.

Locator Attributes for Identifying Silverlight Controls

Supported locator attributes for Silverlight controls include:

- *automationId*
- *caption*
- *className*
- *name*
- All dynamic locator attributes

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and `*`.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

To identify components within Silverlight scripts, you can specify the *automationId*, *caption*, *className*, *name* or any dynamic locator attribute. The *automationId* can be set by the application developer. For example, a locator with an *automationId* might look like `//SLButton[@automationId="okButton"]`.

We recommend using the *automationId* because it is typically the most useful and stable attribute.

Attribute Type	Description	Example
automationId	An identifier that is provided by the developer of the application under test. The Visual Studio designer automatically assigns an *automationId* to every control that is created with the designer. The application developer uses this ID to identify the control in the application code.	`//SLButton[@automationId="okButton"]`
caption	The text that the control displays. When testing a localized application in multiple languages, use the *automationId* or *name* attribute instead of the *caption*.	`//SLButton[@caption="Ok"]`
className	The simple .NET class name (without namespace) of the Silverlight control. Using the *className* attribute can help to identify a custom control that is derived from a standard Silverlight control that Silk4NET recognizes.	`//SLButton[@className='MyCustomButton']`
name	The name of a control. Can be provided by the developer of the application under test.	`//SLButton[@name="okButton"]`

Attention: The *name* attribute in XAML code maps to the locator attribute *automationId*, not to the locator attribute *name*.

During recording, Silk4NET creates a locator for a Silverlight control by using the *automationId*, *name*, *caption*, or *className* attributes in the order that they are listed in the preceding table. For example, if a control has an *automationId* and a *name*, Silk4NET uses the *automationId*, if it is unique, when creating the locator.

The following table shows how an application developer can define a Silverlight button with the text "Ok" in the XAML code of the application:

XAML Code for the Object	Locator to Find the Object from Silk Test
`<Button>Ok</Button>`	`//SLButton[@caption="Ok"]`
`<Button Name="okButton">Ok</Button>`	`//SLButton[@automationId="okButton"]`
`<Button AutomationProperties.AutomationId="okButton">Ok</Button>`	`//SLButton[@automationId="okButton"]`
`<Button AutomationProperties.Name="okButton">Ok</Button>`	`//SLButton[@name="okButton"]`

Attributes for Web Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Web applications include:

- caption (supports wildcards *?* and *)
- all DOM attributes (supports wildcards *?* and *)

Note: Empty spaces are handled differently by each browser. As a result, the `textContent` and `innerText` attributes have been normalized. Empty spaces are skipped or replaced by a single

space if an empty space is followed by another empty space. Empty spaces are detected spaces, carriage returns, line feeds, and tabs. The matching of such values is normalized also. For example:

```
<a>abc
abc</a>
```

Uses the following locator:

```
//A[@innerText='abc abc']
```

Attributes for Windows API-based Client/Server Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Windows API-based client/server applications include:

- caption
- windowid
- priorlabel: Helps to identify text input fields by the text of its adjacent label field. Every input field of a form usually has a label that explains the purpose of the input. For controls that do not have a caption, the attribute **priorlabel** is automatically used in the locator. For the **priorlabel** value of a control, for example a text box, the caption of the closest label at the left side or above the control is used.

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

Attributes for Windows Forms Applications

When a locator is constructed, the attribute type is automatically assigned based on the technology domain that your application uses. The attribute type and value determines how the locator identifies objects within your tests.

Supported attributes for Windows Forms applications include:

- automationid
- caption
- windowid
- priorlabel (For controls that do not have a caption, the priorlabel is used as the caption automatically. For controls with a caption, it may be easier to use the caption.)

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

Attributes for Windows Presentation Foundation (WPF) Applications

Supported attributes for WPF applications include:

- *automationId*
- *caption*
- *className*
- *name*
- All dynamic locator attributes.

Note: Attribute names are case sensitive. The locator attributes support the wildcards ? and *.

For additional information on dynamic locator attributes, see *Dynamic Locator Attributes*.

Object Recognition

To identify components within WPF scripts, you can specify the *automationId*, *caption*, *className*, or *name*. The name that is given to an element in the application is used as the *automationId* attribute for the locator if available. As a result, most objects can be uniquely identified using only this attribute. For example, a locator with an *automationId* might look like: `//WPFButton[@automationId='okButton']"`.

If you define an *automationId* and any other attribute, only the *automationId* is used during replay. If there is no *automationId* defined, the *name* is used to resolve the component. If neither a *name* nor an *automationId* are defined, the *caption* value is used. If no caption is defined, the *className* is used. We recommend using the *automationId* because it is the most useful property.

Attribute Type	Description	Example
automationId	An ID that was provided by the developer of the test application.	`//WPFButton[@automationId='okButton']"`
name	The name of a control. The Visual Studio designer automatically assigns a name to every control that is created with the designer. The application developer uses this name to identify the control in the application code.	`//WPFButton[@name='okButton']"`
caption	The text that the control displays. When testing a localized application in multiple languages, use the automationId or name attribute instead of the caption.	`//WPFButton[@automationId='Ok']"`
className	The simple .NET class name (without namespace) of the WPF control. Using the class name attribute can help to identify a custom control that is derived from a standard WPF control that Silk4NET recognizes.	`//WPFButton[@className='MyCustomButton']"`

During recording, Silk4NET creates a locator for a WPF control by using the *automationId*, *name*, *caption*, or *className* attributes in the order that they are listed in the preceding table. For example, if a control has a *automationId* and a *name*, Silk4NET uses the *automationId* when creating the locator.

The following example shows how an application developer can define a *name* and an *automationId* for a WPF button in the XAML code of the application:

```
<Button Name="okButton" AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

Dynamic Locator Attributes

In a locator for identifying a control during replay you can use a pre-defined set of locator attributes, for example *caption* and *automationId*, which depend on the technology domain. But you can also use every property, including dynamic properties, of a control as locator attribute. A list of available properties for a certain control can be retrieved with the `GetPropertyList` method. All returned properties can be used for identifying a control with a locator.

Note: You can use the `GetProperty` method to retrieve the actual value for a certain property of interest. You can then use this value in your locator.

Example

If you want to identify a button on a dialog box in a WPF application you can type:

```
Dim button = dialog.Find("//WPFButton[@IsDefault=true]")
```

or alternatively

```
Dim button = dialog.WPFButton("@IsDefault=true")
```

This works because Silk Test Workbench exposes a property called `IsDefault` for the WPF button control.

Example

If you want to identify a button in a WPF application with the font size 12 you can type:

```
Dim button = dialog.Find("//WPFButton[@FontSize=12]")
```

or alternatively

```
Dim button = dialog.WPFButton("@FontSize=12")
```

This works because the underlying control in the application under test, in this case the WPF button, has a property called `FontSize`.

Enhancing Tests

This section describes how you can enhance a test.

Calling Windows DLLs

This section describes how you can call DLLs. You can call a DLL either within the process of the Open Agent or in the application under test (AUT). This allows the reuse of existing native DLLs in test scripts.

DLL calls in the Open Agent are typically used to call global functions that do not interact with UI controls in the AUT.

DLL calls in the AUT are typically used to call functions that interact with UI controls of the application. This allows Silk4NET to automatically synchronize the DLL call during playback.

Note: In 32-bit applications, you can call 32-bit DLLs, while in 64-bit applications you can call 64-bit DLLs. The Open Agent can execute both 32-bit and 64-bit DLLs.

Note: The .NET framework also provides built-in support for DLL calling, which is called P/Invoke. P/Invoke can be used in Visual Basic scripts to call DLL functions within the process that executes the script. However, in contrast to calling DLL functions with Silk Test Workbench in the application under test, there is no automatic synchronization.

Note: You can only call DLLs with a C interface. If you want to call .NET assemblies, which also have the file extension .dll, do not use the DLL calling feature but instead just add a reference to the assembly in your .NET script.

Calling a Windows DLL from Within a Script

A declaration for a DLL starts with an interface that has a Dll attribute. The syntax of the declaration is the following:

dllname	The name of or the full path to the DLL file that contains the functions you want to call from your scripts. Environment variables in the DLL path are automatically resolved. You do not have to use double backslashes (\\) in the path, single backslashes (\) are sufficient.
DllInterfaceName	The identifier that is used to interact with the DLL in a script.
FunctionDeclaration	A function declaration of a DLL function you want to call.

DLL Function Declaration Syntax

A function declaration for a DLL typically has the following form:

For functions that do not have a return value, the declaration has the following form:

return-type	The data type of the return value.
function-name	The name of the function.
arg-list	A list of the arguments that are passed to the function. The list is specified as follows: **data-type** The data type of the argument.

identifier The name of the argument.

Passing Arguments to DLL Functions

DLL functions are written in C, so the arguments that you pass to these functions must have the appropriate C data types. The following data types are supported:

Use this data type for arguments or return values with the following data types:

- int
- INT
- long
- LONG
- DWORD
- BOOL
- WPARAM
- HWND

The type works for all DLL arguments that have a 4-byte value.

Use this data type for arguments or return values with the C data types long and int64. The type works for all DLL arguments that have an 8-byte value.

Use this data type for arguments or return values with the C data types short and WORD. The type works for all DLL arguments that have a 2-byte value.

Use this data type for arguments or return values with the C data type bool.

String Use this for arguments or return values that are Strings in C.

Use this for arguments or return values with the C data type double.

Use this for arguments with the C data type RECT. cannot be used as a return value.

Use this for arguments with the C data type POINT. Point cannot be used as a return value.

Use this for arguments with the C data type HWND. TestObject cannot be used as a return value, however you can declare DLL functions that return a HWND with an Integer as the return type.

Note: The passed TestObject must implement the interface so that Silk4NET is able to determine the window handle for the TestObject that should be passed into the DLL function. Otherwise an exception is thrown when calling the DLL function.

List Use this for arrays for user defined C structs. Lists cannot be used as a return value.

Note: When you use a List as an parameter, the list that is passed in must be large enough to hold the returned contents.

Note: A C struct can be represented by a List, where every list element corresponds to a struct member. The first struct member is represented by the first element in the list, the second struct members is represented by the second element in the list, and so on.

Note: Any argument that you pass to a DLL function must have one of the preceding data types.

Passing String Arguments to DLL Functions

Strings that are passing into a DLL function or that are returned by a DLL function are treated by default as Unicode Strings. If your DLL function requires ANSI String arguments, use the `CharacterSet` property of the `DllFunctionOptions` attribute.

Example

```
<Dll( "user32.dll" )> Public Interface IUserDll32Functions
  <DllFunctionOptions(CharacterSet:=CharacterSet.Ansi)>
Function SendMessageA( _
    ByVal obj As TestObject, ByVal message As Integer , ByVal
wParam As Integer , ByRef lParam As String ) As Integer
End Interface
```

Passing a String back from a DLL call as a ByRef argument works per default if the String's size does not exceed 256 characters length. If the String that should be passed back is longer than 256 characters, you need to pass a Visual Basic String in that is long enough to hold the resulting String.

Example

Use the following code to create a String with 1024 blank characters:

```
Dim longEmptyString = New String ( " "c , 1024 )
```

Pass this String as a ByRef argument into a DLL function and the DLL function will pass back Strings of up to 1024 characters of length.

When passing a String back from a DLL call as a function return value, the DLL should implement a DLL function called `FreeDllMemory` that accepts the C String pointer returned by the DLL function and that frees the previously allocated memory. If no such function exists the memory will be leaked.

Aliasing a DLL Name

If a DLL function has the same name as a reserved word in Visual Basic, or the function does not have a name but an ordinal number, you need to rename the function within your declaration and use the alias statement to map the declared name to the actual name.

Example

For example, the `Exit` statement is reserved by the Visual Basic compiler. Therefore, to call a function named `exit`, you need to declare it with another name, and add an alias statement, as shown here:

```
<Dll("mydll.dll")> Public Interface IMyDllFunctions
  <DllFunctionOptions(Alias:="exit")> Sub MyExit()
End Interface
```

Conventions for Calling DLL Functions

The following calling conventions are supported when calling DLL functions:

- __stdcall
- __cdecl

The __*stdcall* calling convention is used by default when calling DLL functions. This calling convention is used by all Windows API DLL functions.

You can change the calling convention for a DLL function by using the `CallingConvention` property of the `DllFunctionOptions` attribute.

Example

The following code example declares a DLL function with the __*decl* calling convention:

```
<Dll("msvcrt.dll")> Public Interface IMsVisualCRuntime
```

```
<DllFunctionOptions(CallingConvention:=CallingConvention.Cdecl)>
 Function cos(ByVal input As Double) As Double
End Interface
```

Windows Accessibility

This section describes how you can use Windows Accessibility (Accessibility) to ease the recognition of objects at the class level.

Using Accessibility

Win32 uses the Accessibility support for controls that are recognized as generic controls. When Win32 locates a control, it tries to get the accessible object along with all accessible children of the control.

Objects returned by Accessibility are either of the class `AccessibleControl`, `Button` or `CheckBox`. `Button` and `Checkbox` are treated specifically because they support the normal set of methods and properties defined for those classes. For all generic objects returned by Accessibility the class is `AccessibleControl`.

Example

If an application has the following control hierarchy before Accessibility is enabled:

- Control
  - Control
- Button

When Accessibility is enabled, the hierarchy changes to the following:

- Control
  - Control
    - Accessible Control
    - Accessible Control
      - Button
- Button

Enabling Accessibility

If you are testing a Win32 application and cannot recognize objects, you should first enable Accessibility. Accessibility is designed to enhance object recognition at the class level.

To enable Accessibility:

1. Click . The dialog box opens.
2. Click **Advanced**.
3. Select the **Use Microsoft Accessibility** option. Accessibility is turned on.

Dynamic Invoke

Dynamic invoke enables you to directly call methods, retrieve properties, or set properties, on an actual instance of a control in the application under test. You can also call methods and properties that are not available in the API for this control. Dynamic invoke is especially useful when you are working with custom

controls, where the required functionality for interacting with the control is not exposed through the Silk Test Workbench API.

Note: You can use dynamic invoke with scripts. Dynamic invoke is not available in visual tests.

Call dynamic methods on objects with the `Invoke` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Call multiple dynamic methods on objects with the `InvokeMethods` method. To retrieve a list of supported dynamic methods for a control, use the `GetDynamicMethodList` method.

Retrieve dynamic properties with the `GetProperty` method and set dynamic properties with the `SetProperty` method. To retrieve a list of supported dynamic properties for a control, use the `GetPropertyList` method.

For example, to call a method named `SetTitle`, which requires the title to be set as an input parameter of type string, on an actual instance of a control in the application under test, type the following:

```
control.Invoke("SetTitle", "my new title")
```

Note: Typically, most properties are read-only and cannot be set.

Note: Reflection is used in most technology domains to call methods and retrieve properties.

Text Recognition Support

Text recognition methods enable you to conveniently interact with test applications that contain highly customized controls, which cannot be identified using object recognition. You can use *text clicks* instead of coordinate-based clicks to click on a specified text string within a control.

For example, you can simulate selecting the first cell in the second row of the following table:

CustomerName	FirstOrder	ID	IsActive	CreditCard
Bob Villa	01.01.2008	0	☑	MasterCard
Brian Miller	02.01.2008	1	☐	Visa
Caral Rudd	03.01.2008	2	☑	American Ex...
Dan Rundgren	04.01.2008	3	☐	MasterCard
Devie Yingstein	05.01.2008	4	☑	Visa

Specifying the text of the cell results in the following code line:

Text recognition methods are supported for the following technology domains:

- Win32.
- WPF.
- Windows Forms.
- Java SWT and Eclipse.
- Java AWT/Swing.

  Note: For Java Applets, and for Swing applications with Java versions prior to version 1.6.10, text recognition is supported out-of-the-box. For Swing applications with Java version 1.6.10 or later, you have to add the following command-line element when starting the application:

  ```
  -Dsun.java2d.d3d=false
  ```

  For example:

  ```
  javaw.exe -Dsun.java2d.d3d=false -jar mySwingApplication.jar
  ```
- xBrowser.

Text recognition methods

The following methods enable you to interact with the text of a control:

`TextCapture`	Returns the text that is within a control. Also returns text from child controls.
`TextClick`	Clicks on a specified text within a control.
`TextRectangle`	Returns the rectangle of a certain text within a control or a region of a control.
`TextExists`	Determines whether a given text exists within a control or a region of a control.

Text click recording

When text click recording is enabled, records `TextClick` methods instead of clicks with relative coordinates. Use this approach for controls where `TextClick` recording produces better results than normal coordinate-based clicks. You can insert text clicks in your script for any control, even if the text clicks are not recorded.

If you do not whish to record a `TextClick` action, you can turn off text click recording and record normal clicks.

The text recognition methods prefer whole word matches over partially matched words. recognizes occurrences of whole words previously than partially matched words, even if the partially matched words are displayed before the whole word matches on the screen. If there is no whole word found, the partly matched words will be used in the order in which they are displayed on the screen.

Example

The user interface displays the text *the hostname is the name of the host*. The following code clicks on *host* instead of *hostname*, although *hostname* is displayed before *host* on the screen: The following code clicks on the substring *host* in the word *hostname* by specifying the second occurrence:

Custom Controls

You can create custom classes for custom controls for which Silk4NET does not offer any dedicated support. Creating custom classes offers the following advantages:

- Better locators for scripts.
- An easy way to write reusable code for the interaction with the custom control.

Example

Suppose that a custom tab control is recognized by Silk4NET as the generic class `Control`. Using the custom control support of Silk4NET has the following advantages:

Better object recognition because the custom control class name can be used in a locator.	Many objects might be recognized as `Control`. The locator requires an index to identify the specific object. For example, the object might be identified by the locator `//Control[13]`. When you create a custom class for this control, for example the class `MyTabControl`, you can use the locator `//MyTabControl`. By creating the custom class, you do not require the high index, which would be a fragile object identifier if the application under test changed.

You can implement reusable playback actions for the control in scripts.	Without using the custom classes, when you want to select a tab in your custom tab controls, you can write code like the following:
	`tabControl.TextClick("<TabName>")`
	When you are using custom classes, you can encapsulate the behavior for selecting a tab into a method by adding the following code to your custom class, which is the class that gets generated when you specify the custom control in the user interface:
	`Public Sub SelectTab(tabText As String)`
	`  TextClick(tabText)`
	`End Sub`
	The custom class looks like the following:
	`Public Class MyTabControl`
	`  Inherits Control`
	`  Public Sub New(objectHandle As ObjectHandle)`
	`    MyBase.New(objectHandle)`
	`  End Sub`
	`  Public Sub SelectTab(tabText As String)`
	`    TextClick(tabText)`
	`  End Sub`
	`End Class`
	You can now use the newly created method `SelectTab` in a script like the following:
	`tabControl.SelectTab("<TabName>")`

Supporting a Custom Control

To create a custom class for a custom control for which Silk4NET does not offer any dedicated support.

1. Click **Silk4NET** > **Manage Custom Controls**. The **Manage Custom Controls** dialog box opens.
2. In the **Silk4NET Custom Controls Output Directory** field, type in a name or click **Browse** to select the script that will contain the custom control.
3. Click on the tab of the technology domain for which you want to create a new custom class.
4. Click **Add**.
5. Click one of the following:
   - Click **Identify new custom control** to directly select a custom control in your application with the **Identify Object** dialog box.
   - Click **Add new custom control** to manually add a custom control to the list.

   A new row is added to the list of custom controls.
6. In the **Silk Test base class** column, select an existing base class from which your class will derive.
   This class should be the closest match to your type of custom control.
7. In the **Silk Test class** column, enter the name to use to refer to the class.
   This is what will be seen in locators. For example: `//MyTabControl` instead of `//Control[13]`.

Note: After you add a valid class, it will become available in the **Silk Test base class** list. You can then reuse it as a base class.

8. In the **Custom control class name** column, enter the fully qualified class name of the class that is being mapped.

   For example: `Infragistics.Win.UltraWinTabControl.UltraTabControl`.

9. *Only for Win32 applications:* In the **Class mapping** column, set the class mapping to true to map the name of a custom control class to the name of a standard Silk Test class.

   When you map the custom control class to the standard Silk Test class, you can use the functionality supported for the standard Silk Test class in your test.

10. Click **OK**.

11. *Only for scripts:*

    a) Add custom methods and properties to your class for the custom control.

    b) Use the custom methods and properties of your new class in your script.

    Note: The custom methods and properties are not recorded.

Note: Do not rename the custom class or the base class in the script file. Changing the generated classes in the script might result in unexpected behavior. Use the script only to add properties and methods to your custom classes. Use the **Manage Custom Controls** dialog box to make any other changes to the custom classes.

Custom Controls Options

Silk4NET > **Manage Custom Controls**.

In the **Silk4NET Custom Controls Output Directory**, define the script file into which the new custom classes should be generated.

The following **Custom Controls** options are available:

Option	Description
Silk Test base class	Select an existing base class to use that your class will derive from. This class should be the closest match to your type of custom control.
Silk Test class	Enter the name to use to refer to the class. This is what will be seen in locators. For example: `//MyTabControl` instead of `//Control[13]`.
Custom control class name	Enter the fully qualified class name of the class that is being mapped. For example: `Infragistics.Win.UltraWinTabControl.UltraTabControl`.
Class mapping	This option is available only for Win32 applications. Set the class mapping to true to map the name of a custom control class to the name of a standard Silk Test class. When you map the custom control class to the standard Silk Test class, you can use the functionality supported for the standard Silk Test class in your test.

Note: After you add a valid class, it will become available in the **Silk Test base class** list. You can then reuse it as a base class.

xBrowser Frequently Asked Questions

This section includes a collection of questions that you might encounter when testing your Web application.

How do I Verify the Font Type Used for the Text of an Element?

You can access all attributes of the `currentStyle` attribute of a DOM element by separating the attribute name with a ":".

Windows Internet Explorer 8 or earlier	`wDomElement.GetProperty("currentStyle:fontName")`
All other browsers, for example Windows Internet Explorer 9 or later and Mozilla Firefox	`wDomElement.GetProperty("currentStyle:font-name")`

What is the Difference Between textContents, innerText, and innerHtml?

- `textContents` is all text contained by an element and all its children that are for formatting purposes only.
- `innerText` returns all text contained by an element and all its child elements.
- `innerHtml` returns all text, including html tags, that is contained by an element.

Consider the following html code.

```
<div id="mylinks">
  This is my <b>link collection</b>:
  <ul>
    <li><a href="www.borland.com">Bye bye <b>Borland</b> </a></li>
    <li><a href="www.microfocus.com">Welcome to <b>Micro Focus</b></a></li>
  </ul>
</div>
```

The following table details the different properties that return.

Code	Returned Value
`browser.DomElement("//div[@id='mylinks']").GetProperty("textContents")`	`This is my link collection:`
`browser.DomElement("//div[@id='mylinks']").GetProperty("innerText")`	`This is my link collection:Bye bye Borland Welcome to Micro Focus`
`browser.DomElement("//div[@id='mylinks']").GetProperty("innerHtml")`	`This is my <b>link collection</b>: <ul> <li><a href="www.borland.com">Bye bye <b>Borland</b></a></li>`

Code	Returned Value
	Welcome to Micro Focus

I Configured innerText as a Custom Class Attribute, but it Is Not Used in Locators

A maximum length for attributes used in locator strings exists. `InnerText` tends to be lengthy, so it might not be used in the locator. If possible, use `textContents` instead.

What Should I Take Care Of When Creating Cross-Browser Scripts?

When you are creating cross-browser scripts, you might encounter one or more of the following issues:

- Different attribute values. For example, colors in Windows Internet Explorer are returned as `"#FF0000"` and in Mozilla Firefox as `"rgb(255,0,0)"`.
- Different attribute names. For example, the font size attribute is called `"fontSize"` in Windows Internet Explorer 8 or earlier and is called `"font-size"` in all other browsers, for example Windows Internet Explorer 9 or later and Mozilla Firefox.
- Some frameworks may render different DOM trees.

How Can I See Which Browser I Am Currently Using?

The `BrowserApplication` class provides a property `"browsertype"` that returns the type of the browser. You can add this property to a locator in order to define which browser it matches.

For information about new features, supported platforms and versions, known issues, and work-arounds, refer to the *Silk Test Release Notes*, available from *http://supportline.microfocus.com/productdoc.aspx*.

Examples

To get the browser type, type the following into the locator:

```
browserApplication.GetProperty("browsertype")
```

Additionally, the `BrowserWindow` provides a method `GetUserAgent` that returns the user agent string of the current window.

Which Locators are Best Suited for Stable Cross-Browser Testing?

The built in locator generator attempts to create stable locators. However, it is difficult to generate quality locators if no information is available. In this case, the locator generator uses hierarchical information and indices, which results in fragile locators that are suitable for direct record and replay but ill-suited for stable, daily execution. Furthermore, with cross-browser testing, several AJAX frameworks might render different DOM hierarchies for different browsers.

To avoid this issue, use custom IDs for the UI elements of your application.

Logging Output of My Application Contains Wrong Timestamps

This might be a side effect of the synchronization. To avoid this problem, specify the HTML synchronization mode.

My Test Script Hangs After Navigating to a New Page

This can happen if an AJAX application keeps the browser busy (open connections for Server Push / ActiveX components). Try to set the HTML synchronization mode. Check the *Page Synchronization for xBrowser* topic for other troubleshooting hints.

Recorded an Incorrect Locator

The attributes for the element might change if the mouse hovers over the element. Silk4NET tries to track this scenario, but it fails occasionally. Try to identify the affected attributes and configure Silk4NET to ignore them.

Rectangles Around Elements in Windows Internet Explorer are Misplaced

- Make sure the zoom factor is set to 100%. Otherwise, the rectangles are not placed correctly.
- Ensure that there is no notification bar displayed above the browser window. Silk4NET cannot handle notification bars.

Link.Select Does Not Set the Focus for a Newly Opened Window in Internet Explorer

This is a limitation that can be fixed by changing the Browser Configuration Settings. Set the option to always activate a newly opened window.

DomClick(x, y) Is Not Working Like Click(x, y)

If your application uses the `onclick` event and requires coordinates, the `DomClick` method does not work. Try to use `Click` instead.

FileInputField.DomClick() Will Not Open the Dialog

Try to use `Click` instead.

The Move Mouse Setting Is Turned On but All Moves Are Not Recorded. Why Not?

In order to not pollute the script with a lot of useless `MoveMouse` actions, Silk4NET does the following:

- Only records a `MoveMouse` action if the mouse stands still for a specific time.
- Only records `MoveMouse` actions if it observes activity going on after an element was hovered over. In some situations, you might need to add some manual actions to your script.

I Need Some Functionality that Is Not Exposed by the xBrowser API. What Can I Do?

You can use `ExecuteJavaScript()` to execute JavaScript code directly in your Web application. This way you can build a workaround for nearly everything.

Why Are the Class and the Style Attributes Not Used in the Locator?

These attributes are on the ignore list because they might change frequently in AJAX applications and therefore result in unstable locators. However, in many situations these attributes can be used to identify objects, so it might make sense to use them in your application.

Dialog is Not Recognized During Replay

When recording a script, Silk4NET recognizes some windows as `Dialog`. If you want to use such a script as a cross-browser script, you have to replace `Dialog` with `Window`, because some browsers do not recognize `Dialog`.

For example, the script might include the following line:

```
/BrowserApplication//Dialog//PushButton[@caption='OK']
```

Rewrite the line to enable cross-browser testing to:

```
/BrowserApplication//Window//PushButton[@caption='OK']
```

Why Do I Get an Invalidated-Handle Error When I Call the WaitForProperty Method?

This topic describes what you can do when you call the `WaitForProperty` method and Silk4NET displays the following error message: `The handle for this object has been invalidated.`

This message indicates that something caused the object, on which you called `WaitForProperty`, to disappear. For example, if something causes the browser to navigate to a new page, during the `WaitForProperty` call in a Web application, all objects on the previous page are automatically invalidated.

Sometimes the reason for this problem is the built-in synchronization. For example, suppose that the application under test includes a shopping cart, and you have added an item to this shopping cart. You are waiting for the next page to be loaded and for the shopping cart to change its status to `contains items`. If the action, which adds the item, returns too soon, the shopping cart on the first page will be waiting for the status to change while the new page is loaded, causing the shopping cart of the first page to be invalidated. This behavior will result in an invalidated-handle error.

As a workaround, you should wait for an object that is only available on the second page before you verify the status of the shopping cart. As soon as the object is available, you can verify the status of the shopping cart, which is then correctly verified on the second page.

Why Are Clicks Recorded Differently in Windows Internet Explorer 10?

When you record a `Click` on a `DomElement` in Windows Internet Explorer 10 and the `DomElement` is dismissed after the `Click`, then the recording behavior might not be as expected. If another `DomElement` is located beneath the initial `DomElement`, Silk Test records a `Click`, a `MouseMove`, and a `ReleaseMouse`, instead of recording a single `Click`.

A possible workaround for this unexpected recording behavior depends on the application under test. Usually it is sufficient to delete the unnecessary `MouseMove` and `ReleaseMouse` events from the recorded script.

Index

A

Accessibility
    overview 45
adding Silk4NET tests
    Silk4NET projects 16
Adobe Flex
    attributes 35
AJAX applications
    script hangs 52
analyzing test results
    Silk4NET 20
application configurations
    adding 12
    definition 12
    errors 13
    modifying 12
    removing 12
    troubleshooting 13
attribute types
    Adobe Flex 35
    Java AWT 35
    Java Swing 35
    Java SWT 36
    overview 35
    SAP 37
    Silverlight 37
    Web applications 38
    Windows 39
    Windows Forms 39
    xBrowser 38
attribute values
    finding with Locator Spy 19

B

base state
    definition 11
    modifying 11
browser type
    GetProperty 51
browsers
    setting preferences 26
browsertype
    using 51

C

Chrome
    cross-browser scripts 51
class names
    finding with Locator Spy 19
classes
    exposing 28
    ignoring 27
contact information 8, 9
creating Silk4NET projects
    Visual Studio 14
creating visual execution logs
    TrueLog 21
    TrueLog Explorer 21
custom attributes
    setting 26
custom controls
    creating custom classes 48
    dialog box 49
    overview 47
    supporting 48
Customer Care 8, 9

D

Dialog
    not recognized 53
dlls
    aliasing names 44
    calling conventions 44
    function declaration syntax 42
    passing arguments to functions 43
    passing string arguments to functions 43
downloads 8, 9
dynamic invoke
    overview 45
dynamic locator attributes
    about 41
dynamic object recognition
    overview 31
    sample queries 33

E

enabling TrueLog
    TrueLog Explorer 21
exposing WPF classes 28

F

FAQs
    xBrowser 50
Firefox
    cross-browser scripts 51
    locators 51
Flex
    attributes 35

G

getting started
    Silk4NET 10

I

identifying controls
    dynamic locator attributes 41
    Locator Spy 35
ignoring classes 27
innerHTML 50
innerText 50, 51

Internet Explorer
cross-browser scripts 51
link.select focus issue 52
locators 51
Internet Explorer 10
unexpected Click behavior 54
invalidated-handle error
troubleshooting 53

J

Java AWT
attribute types 35
attributes 35
Java Swing
attributes 35
Java SWT
attribute types 36
custom attributes 26

L

locator attributes
dynamic 41
Rumba controls 37
Silverlight controls 37
WPF controls 39
Locator Spy
adding locators to test methods 19
overview 35
locators
attributes 26
incorrect in xBrowser 52
xBrowser 51

M

mouse move actions 25

O

OPT_ALTERNATE_RECORD_BREAK
options 25
OPT_ENSURE_ACTIVE_OBJDEF 29
OPT_RECORD_MOUSEMOVE_DELAY
options 25
OPT_RECORD_MOUSEMOVES
options 25
OPT_RECORD_SCROLLBAR_ABSOLUT
options 25
OPT_REPLAY_MODE 29
OPT_WAIT_RESOLVE_OBJDEF 28
OPT_WAIT_RESOLVE_OBJDEF_RETRY 28
OPT_XBROWSER_RECORD_LOWLEVEL 26
OPT_XBROWSER_SYNC_EXCLUDE_URLS 28
OPT_XBROWSER_SYNC_MODE 28
OPT_XBROWSER_SYNC_TIMEOUT 28

P

Product Support 8, 9
product updates 6
projects
adding Silk4NET tests 16
Silk4NET 14

R

Record Break keys 25
recording
preferences 25
Silk4NET tests 17
replay
Dialog not recognized 53
options 29
Rumba
locator attributes 37
Rumba locator attributes
identifying controls 37
running tests
Silk4NET 19

S

SAP
attribute types 37
custom attributes 26
scripts
specifying options 25
scroll events 25
serial number 8, 9
SetText 26
shortcut key combination 25
Silk4NET
about 6
basic workflow 10
manually creating tests 18
projects 14
tests 16
Silk4NET projects
adding tests 16
Silk4NET tests
analyzing results 20
manually creating 18
recording 17
running 19
Silverlight
attribute types 37
locator attributes 37
specifying options
scripts 25
SupportLine 8, 9
Swing
attributes 35
synchronization options 28

T

Team Foundation Server
locating TrueLogs 23
using with Silk4NET tests 23
test cases
sample queries 33

test methods
adding locators 19
tests
enhancing 42
Silk4NET 16
text click recording
overview 46
text recognition
overview 46
textContents 50
TFS
using with Silk4NET tests 23
timestamps 52
troubleshooting XPath 34
TrueLog
configuring 25
creating visual execution logs 21
enabling 21, 25
replacement characters for non-ASCII 22
wrong non-ASCII characters 22
TrueLog Explorer
configuring 25
creating visual execution logs 21
enabling 25
enabling TrueLog 21
TrueLogs
Team Foundation Server 23
TypeKeys 26

U

unexpected Click behavior
Windows Internet Explorer 54
updates 6

W

WaitForProperty method
invalidated-handle error 53
Web applications
custom attributes 26
supported attributes 38
WebSync 8, 9
Windows
attribute types 39
Windows Accessibility
overview 45
Windows applications
custom attributes 26
Windows Forms
attribute types 39
custom attributes 26
Windows Internet Explorer
misplaced rectangles 52
Windows Internet Explorer 10
unexpected Click behavior 54
Windows Presentation Foundation (WPF)
locator attributes 39
works order number 8, 9
WPF
exposing classes 28
locator attributes 39
WPF applications
custom attributes 26
WPF locator attributes
identifying controls 39

X

xBrowser
attribute types 38
browser type distinctions 51
class and style not in locators 53
cross-browser scripts 51
custom attributes 26
Dialog not recognized 53
DomClick not working like Click 52
exposing functionality 53
FAQs 50
FieldInputField.DomClick not opening dialog 52
font type verification 50
innerText not being used in locators 51
link.select focus issue 52
mouse move recording 53
navigating to new pages 52
recording an incorrect locator 52
recording locators 51
textContents, innerText, innerHTML 50
timestamps 52
Windows Internet Explorer misplaces rectangles 52
XPath
basic concepts 31
creating query strings 35
overview 31
samples 33
supported subset 32
troubleshooting 34