

# **SilkTest<sup>®</sup> 2010**

## **Silk4NET User Guide**

**Borland<sup>®</sup>**  
(A MICRO FOCUS COMPANY)

**MICRO  
FOCUS<sup>®</sup>**  
*Leading the Evolution™*

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

**Copyright 2009-2010 Micro Focus (IP) Limited. All Rights Reserved. SilkTest contains derivative works of Borland Software Corporation, Copyright 2010 Borland Software Corporation (a Micro Focus company).**

**MICRO FOCUS and the Micro Focus logo, among others, are trademarks or registered trademarks of Micro Focus (IP) Limited or its subsidiaries or affiliated companies in the United States, United Kingdom and other countries.**

**BORLAND, the Borland logo and SilkTest are trademarks or registered trademarks of Borland Software Corporation or its subsidiaries or affiliated companies in the United States, United Kingdom and other countries.**

**All other marks are the property of their respective owners.**

# Contents

- Silk4NET .....4**
- Creating a Silk4NET Project in Visual Studio .....4
- Creating a Silk4NET Test Overview .....4
- Adding a Silk4NET Test Template to a Visual Studio Project .....5
- Manually Creating a Silk4NET Test in Visual Studio .....6
- Recording a Test .....6
- Dragging and Dropping Recorded Test Steps .....7
- Exporting Recorded Test Steps to Visual Studio .....7
- Exporting a Silk4NET Project to Visual Studio .....8
- Running Silk4NET Tests .....8
- Analyzing Test Results .....9
- Silk4NET Sample Tests .....9

# Silk4NET

Silk4NET is SilkTest's plug-in for Microsoft Visual Studio that allows you to efficiently create and manage functional, regression, and localization tests directly in the Visual Studio. With Silk4NET you can develop tests using either Visual Basic .NET or C#, run the tests as a part of a test plan in Microsoft's test environment or as a part of your build process, and view the test results all from within Visual Studio.

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

## Creating a Silk4NET Project in Visual Studio

1. Click **File > New > Project**.

The **New Project** dialog box appears.

2. Under **Installed Templates**, expand **Visual C#** or **Visual Basic**, and then select **Silk4NET**.

3. Type the project name, location, solution, and solution name, and then click **OK**.

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

## Creating a Silk4NET Test Overview

You can use the Silk4NET test template as a starting point for manually creating tests. This template contains import namespaces for the required character encoding classes, SilkTest's Open Agent API, and Visual Studio's unit testing support. Additionally, this template contains a generic class and method in which you can create specific test steps designed to test functionality in your applications.

Another method is to use the SilkTest Recorder to automate the creation of tests by recording user actions made against a test application. The SilkTest Recorder records the test in a neutral format that you can then export to your Visual Studio project in either Visual Basic .NET or C#.

The SilkTest Recorder also supports a drag-and-drop feature and a clipboard feature. Both features allow you to quickly move recorded test step into an existing Silk4NET test in Visual Studio. In this way, you can use a mixed-method approach in which you record specific parts of a test and manually code others.

# Adding a Silk4NET Test Template to a Visual Studio Project

1. Choose **Project > Add New Item**.  
The **Add New Item** dialog box appears.
2. Under **Installed Templates**, choose **Common Items > Silk4NET**.  
The Silk4NET test template appears in the middle pane.
3. Select **Silk4NET Test**, and then click **Add**.  
A Silk4NET test template is added to your project with a default file name of `UnitTest1.cs` or `UnitTest1.vb` depending on your project's default programming language.

This template contains import namespaces for SilkTest's Open Agent API and Visual Studio's unit testing support. Additionally, this template contains a generic class and method in which you can create specific test steps, as shown in the following examples:

```
'Visual Basic .NET
Imports System.Text
Imports Microsoft.VisualStudio.TestTools.UnitTesting
Imports SilkTest.Ntf

<TestClass(>
Public Class UnitTest1

    Private ReadOnly _desktop As Desktop = Agent.Desktop

    <TestMethod(>
    Public Sub TestMethod1()
    End Sub

End Class
```

```
//C#
Using System;
Using Microsoft.VisualStudio.TestTools.UnitTesting;
Using SilkTest.Ntf;

namespace SilkTest.Ntf.SampleScripts.CSharp
{
    [TestClass]
    public class UnitTest1
    {
        private readonly Desktop _desktop = Agent.Desktop;

        [TestMethod]
        public void TestMethod1()
        {
            // Put your test here.
        }
    }
}
```

# Manually Creating a Silk4NET Test in Visual Studio

1. Add a Silk4NET test template to your project. Choose **Project > Add New Item > Silk4NET**, and then click **Add**.  
The Silk4NET test template is added to your project.
2. Optionally, 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:\\WINXP\\system32\\notepad.exe",
"/Window[@caption='Untitled - Notepad']")
baseState.WorkingDirectory = "%USERPROFILE%"
baseState.Execute()
```

```
//C#
BaseState baseState = new BaseState("C:\\WINXP\\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.



**Note:** You can use the SilkTest Recorder clipboard feature to quickly copy and paste recorded test steps into a manually created Silk4NET test.

## Recording a Test

1. On the Visual Studio menu bar, select **Silk4NET > Start Recorder**.  
The SilkTest Recorder appears.
2. Create a new script and start recording.  
For more information, refer to the *SilkTest Recorder Help*.

The Recorder saves the test actions in a neutral format from which you have the following options:

- Drag-and-drop the recorded test steps into an existing Silk4NET test. The test steps are automatically converted to either Visual Basic .NET or C# depending on your default programming language.
- Export the recorded test steps as a Silk4NET test that you can add to an existing Visual Studio project.
- Export the recorded test steps in a new test and create a new Silk4NET project in which to save the test.

## Dragging and Dropping Recorded Test Steps


You must have created a Silk4NET test in which to drag-and-drop the recorded test steps.

1. In Visual Studio, open the Silk4NET test in which to drag-and-drop the recorded test steps.
2. On the Visual Studio menu bar, select **Silk4NET > Start Recorder**.  
The SilkTest Recorder appears.
3. Select **Settings > Global Preferences**.  
The **Preferences** dialog box appears.
4. In the **Default client** list, select your preferred programming language.
5. Click **OK**.
6. Create a new test and start recording.  
For more information, refer to the *SilkTest Recorder Help*. After stopping the recording, the recorded test steps appear in a neutral format on the **Actions** tab.
7. Select the steps to move into the Silk4NET test.
8. Drag-and-drop the test steps into the Silk4NET test.

The test steps are automatically converted into the default programming language and pasted in the test.

## Exporting Recorded Test Steps to Visual Studio

1. On the SilkTest Recorder menu bar, choose **File > Export**.  
The **Export** wizard opens.
2. Double-click **Export as NTF Script**.  
The **Export to NTF** page opens.
3. From the **Export to** list box, select one of the following options:
  - **Clipboard** – Copies the recorded test steps to the clipboard. Choose this option to copy and paste the recorded test steps into an existing Silk4NET test of a Visual Studio project.

 **Note:** You do not have to specify the source location or base state when selecting this option.

  - **NTF Script** – Exports the recorded test steps as a test you can add to an existing Visual Studio project. Choose this option if you want to create a new test or overwrite an existing test.
4. From the **Programming language** list box, specify whether the test uses Visual Basic .NET or C#.
5. In the **Test method** box, specify a name for the test method.  
For example, type **TestAutoInput**.
6. In the **Namespace** box, specify the container name for the test.
7. In the **Test class** box, specify the class name to which the test belongs.  
For example, type **AutoTests**.
8. In the **Source folder** box, specify the location to which to export the test.  
Optionally, click and navigate to the folder that you want to use.
9. To include the base state in the exported test, check the **Use base state** check box.

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.

**10. Click Finish.**

Recorder creates a Silk4NET test and exports it to the specified location or to the clipboard.

**11. Add the exported Silk4NET project to your Visual Studio project.** From the Visual Studio menu bar, choose **Project > Add Existing Item**, and then select the exported test.

## Exporting a Silk4NET Project to Visual Studio

1. On the Silk4NET menu bar, choose **File > Export**.  
The **Export** wizard opens.
2. Double-click **Export as Silk4NET Project**.  
The **Export as Silk4NET Project** page opens.
3. From the **Programming language** list box, specify whether the project uses Visual Basic .NET or C#.
4. In the **Project location** text box, specify the location to which to export the project.  
*Optional:* Click and navigate to the folder that you want to use.
5. In the **Project name** text box, specify the project name.  
For example, type **Visual Basic .NET Sample Project**.
6. In the **Namespace** text box, specify the container name for the project.
7. In the **Test class** text box, specify the class name to which the test belongs.  
For example, type **AutoTests**.
8. In the **Test method** text box, specify a name for the test method.  
For example, type **TestAutoInput**.
9. Click **Finish**.  
The SilkTest Recorder creates a new project that includes the recorded test and exports the project to the specified location. From this location, you can open the project in Visual Studio.

## Running Silk4NET Tests

1. Select **Test > Windows > Test View**.  
The **Test View** window opens and displays a list of tests. You can filter the list of projects by selecting filter criteria such as **Project**, **Namespace**, and **Test class** from the **Group by list** located at the top of the window.
2. Select the test(s) to run.
3. Click the **Run selection** button on the **Test View** window toolbar.

In addition to running tests from the **Test View** window, run commands are also available on the **Test** menu, the context-sensitive menu of an opened test, and the **Test Results** window.



## Analyzing Test Results

1. Run a Silk4NET test.
2. On the Visual Studio menu bar, choose **Test > Windows > Test Results**.  
The **Test Results** window appears.

The **Test Results** window displays the test run and the status of the run. To view the details of the test run, right-click the run, and then select **View Test Results Details**.

## 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 SilkTest sample applications.

To access the Silk4Net sample tests, you must install the SilkTest sample applications, which you can download from the following location: [http://techpubs.borland.com/silk\\_gauntlet/SilkTest/](http://techpubs.borland.com/silk_gauntlet/SilkTest/). After you have installed the sample applications, choose **Start > All Programs > Silk > SilkTest 2010 > Samples > Silk4NET Samples** to open the folder containing 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 SilkTest Web-based sample applications:

**Insurance Co. Web site** <http://demo.borland.com/InsuranceWebExtJS/>

**Green Mountain Outpost Web** <http://demo.borland.com/gmopost/>