The Microsoft SRE Inspectors.

Version 0.10 - 25 August, 2010

Abstract

The Security Runtime Engine (SRE) is part of the Microsoft Web Protection Library (WPL); tools Microsoft uses on both internal and external facing web sites to help protect applications from security vulnerabilities.

This document details the standard inspectors supplied with the SRE including their purpose and how to configure them.

For up-to-date versions of this document, the SRE and WPL, including source code visit <http://wpl.codeplex.com>.

This tutorial is based on the September 2010 CTP release of the WPL. This release is an early preview to allow you test the standard SRE inspectors and offer feedback on them.

The WPL is licensed under the Microsoft Public License, an open source license the text of which can be read at <http://www.microsoft.com/opensource/licenses.mspx>.

The standard SRE plugins include two assemblies licensed under a supplement license. This license can be found in the WPL installation directory under the filename SQLBinariesEula.rtf.

This CTP is not suitable for running on production servers.

Contents

Introduction to the SRE 4

Getting Help 4

Getting Started 5

Adding the SRE to your web site 5

Configuring the SRE 6

Changing where plug-ins are loaded from 7

Excluding a path from all inspectors 7

Globally disabling an inspector 7

Programmatically disabling an inspector for a particular page 8

Configuring suspect result handling 8

Configuring logging 9

The Microsoft SRE Inspectors 11

Enabling plug-in configuration 11

Disabling an inspectors for a particular path 11

The AntiXSS Inspector 11

The SQL Injection Inspector 12

The Credit Card Information Leak Inspector 13

The Fake Post-Back Inspector 14

The Click Jack Header Inspector 14

The Cookie Protection Inspector 14

The No Open Header Inspector 15

What next? 16

Code Listings

It is often necessary to split code listings over multiple lines in this document when, in fact, they are a single line. The ⮰ symbol indicates where this split takes place. When entering code containing ⮰ symbols please ensure you put the code on a single line.

**Disclaimer:** This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal reference purposes.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft, IntelliSense, Visual Studio, and Windows are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

# Introduction to the SRE

The Security Runtime Engine (SRE) is an HTTP module which wraps itself around the requests to your web application and the response those requests cause. It provides a common framework to examine, log and act on requests and responses without having to access to an ASP.NET application’s source code.

The SRE has a plug-in model which allows you to write your own “inspectors”. The SRE distribution comes with a number of pre-written inspectors.

This tutorial introduces the basic features of the SRE plug-in with a series of exercises that show you how to:

* Create an inspector which examines in-bound requests.
* Load your inspector into the SRE.

## Getting Help

If you have a question or comment please post it on the [discussion forum](http://wpl.codeplex.com/Thread/List.aspx) on the WPL CodePlex site.

# Getting Started

## Adding the SRE to your web site

By default the SRE will be installed into

C:\Program Files\Microsoft Information Security\Web Protection Library v4.0\SecurityRuntimeEngine

or

C:\Program Files (x86)\Microsoft Information Security\Web Protection Library v4.0\SecurityRuntimeEngine

if you are running on a 64bit operating system.

If you installed the .NET 3.5 version of the SRE the assemblies will be created in a NET35 sub-directory; if you installed the .NET 4.0 version of the SRE the assemblies will be created in a NET40 sub-directory. Please ensure you use the correct assemblies for the version of .NET your application uses. You don’t need to copy the XML file contained in the SRE directory – it is there to support intellisense when you developer your own inspectors and is not needed at runtime.

To wrap the SRE around your web site copy the files in the NET35 or NET40 sub-directory into the bin directory of your web site. You should also copy the wplPlugins directory, creating a wplPlugins directory in your site’s bin directory.

Finally you need to enable the SRE in your application web.config.

If you are using IIS7 edit the system.webServer/modules section and add

<modules runAllManagedModulesForAllRequests="true">

…

**<add name="SecurityRuntimeEngine"   
 preCondition="managedHandler"   
 type="Microsoft.Security.Application.⮰  
SecurityRuntimeEngine.SecurityRuntimeEngine, ⮰  
Microsoft.Security.Application.SecurityRuntimeEngine"/>**

…

</modules>

If you do not have a system.webServer/modules section create one first.

If you are using IIS6 edit the httpModules section and add

<httpModules>

…

<add name="SecurityRuntimeEngine"   
 type="Microsoft.Security.Application.**⮰**  
SecurityRuntimeEngine.SecurityRuntimeEngine, **⮰**Microsoft.Security.Application.SecurityRuntimeEngine"/>

…

</httpModules>

If you are using the Visual Studio built in web-server you should use the IIS6 configuration method. If you are using Visual Studio’s web-server, but wish to deploy to IIS7 you must combine both methods and tweak the validation setting in system.webServer so your web.config looks something like

<?xml version="1.0"?>

<configuration>

<system.web>

<compilation debug="true" />

<httpModules>

**<add name="SecurityRuntimeEngine" ⮰  
 type="Microsoft.Security.Application.⮰  
SecurityRuntimeEngine.SecurityRuntimeEngine, ⮰  
Microsoft.Security.Application.SecurityRuntimeEngine"/>**

</httpModules>

</system.web>

<system.webServer>

**<validation validateIntegratedModeConfiguration="true"/>**

**<modules runAllManagedModulesForAllRequests="true">**

**<remove name="SecurityRuntimeEngine" />**

**<add name="SecurityRuntimeEngine" ⮰  
 preCondition="managedHandler" ⮰  
 type="Microsoft.Security.Application.⮰  
SecurityRuntimeEngine.SecurityRuntimeEngine, ⮰  
Microsoft.Security.Application.SecurityRuntimeEngine"/>**

</modules>

</system.webServer>

</configuration>

You can then test that the module is loaded correctly and finding its plugins by passing a SQL Injection string to any of your pages, for example

http://localhost:1234/default.aspx?test='%20or%201=1--

should be stopped by the SQL Injection inspector and you should see the error page your application normally shows when an exception occurs.

## Configuring the SRE

The SRE has some central configuration settings which will apply to the engine itself and/or to all inspectors hosted by the engine. In order to enable configuration you must first add the configuration details to your application’s web.config. In the configSections section add the following line

<configSections>

**<section name="sreSettings"  
 type="Microsoft.Security.Application.⮰  
SecurityRuntimeEngine.SecurityRuntimeSettings, ⮰  
Microsoft.Security.Application.SecurityRuntimeEngine" />**

</configSections>

If you don’t have a configSections section create one in the configuration section of web.config.

You can now create an empty sreSettings section;

<sreSettings>

</sreSettings>

which will hold the SRE configuration.

### Changing where plug-ins are loaded from

You can change the plug-in directory by adding the plugInDirectory attribute to the sreSettings element;

<sreSettings **plugInDirectory="d:\srePlugins"**>

</sreSettings>

You may also use a relative path, which is based from the bin directory of your web application; for example the following configuration will search all assemblies in your application’s bin directory for plug-ins.

<sreSettings **plugInDirectory="."**>

</sreSettings>

### Excluding a path from all inspectors

You can exclude a request’s path from inspection by adding an excludedPath collection with the sreSettings element. Paths are case insensitive but do not support wildcards or regular expressions. If you are using request rewriting you must list all possible routes that you wish to exclude. For example the following configuration will exclude any request to /unprotected/example.aspx, /unprotected/Example.aspx or any other casing combination:

<sreSettings>

**<excludedPaths>**

**<add path="/unprotected/example.aspx" />**

**</excludedPaths>**

</sreSettings>

### Globally disabling an inspector

You can exclude an inspector globally by listing it in the disabledPlugIns section of the sreSettings element – this is useful if an assembly contains multiple inspectors one or more of which you do not want to use. Inspectors are identified by their fully qualified class type. For example the SQL Injection inspector has a fully qualified class name of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.SqlInjectionRequestInspector, to exclude it the configuration would look as follows;

<sreSettings>

**<disabledPlugIns>**

**<add name="SqlInjectionInspector"**

**type="Microsoft.Security.Application.⮰  
SecurityRuntimeEngine.PlugIns.SqlInjectionRequestInspector"   
 />**

**</disabledPlugIns>**

</sreSettings>

When excluding a plug-in you must also supply a unique name; this allows you to remove the exclusion in child applications, for example;

<sreSettings>

**<disabledPlugIns>**

**<remove name="SqlInjectionInspector"/>**

**</disabledPlugIns>**

</sreSettings>

If you encounter a situation where you have inspectors with identical fully qualified names you can extend the type to include the assembly name and, if necessary version, culture and the public token of the strong name signing key.

### Programmatically disabling an inspector for a particular page

If you have source access to the web application you can use attribute based exclusions on the handler classes your request resolves into to exclude inspectors both globally and for specific inspectors. For page inspectors, which inspect WebForms page elements you can exclude specific properties/controls on the page by applying the attribute to the object or property backing the control..

To suppress all inspections on a handler add the SuppressInspection attribute;

**[SuppressProtection]**

public partial class NoInspection : Page

{

}

To suppress a specific inspector you specify the type;

**[SuppressProtection(typeof(SqlInjectionInspector)]**

public partial class NoInspection : Page

{

}

Multiple attributes can be used to exclude multiple inspectors.

### Configuring suspect result handling

Inspectors can return three types of results, Continue (where the inspector didn’t find anything it was concerned about), Halt (which stops the processing of the request or delivery of the result) and Suspect (where the inspector detects a potential problem). The circumstances around which result is returned is determined by the inspector itself, for example The Credit Card response inspector would not return a Halt status if a parameter is a potential SQL Injection attack because it’s not concerned with SQL Injection attacks.

The SRE allows configuration of how it handles Suspect results with the allowedSuspectResults attribute. The default value for allowedSuspectResults is -1, which halts processing after any suspect result.

The SRE also allows the suspect results to be reset once each type of inspector has completed with the resetSuspectCountBetweenStages attribute. Resetting the count between stages would, for example, mean that any suspect results found during request inspection would be discarded once all the request inspectors have finished and the page inspectors which follow would start with a suspect result count of 0. The default value for resetSuspectCountBetween stages is false.

The following example configuration allows a maximum of 3 suspect results before processing is halted. It also resets the count between each processing stage.

<sreSettings **allowedSuspectResults = 3  
 resetSuspectCountBetweenStages = true**>

</sreSettings>

### Configuring logging

The SRE comes with an API for logging. Logging plug-ins work like Inspector plug-ins can be simply dropped into the plugin directory. Log messages come in four priorities,

1. Informational
2. Warning
3. Error
4. Fatal

The default logging level is Error - you control the minimum level at which messages are logged using the logLevel attribute;

<sreSettings **logLevel = "Warning"**>

</sreSettings>

Setting a log level means any logging messages sent by an inspector below the selected level will not be logged; for example a log level of Warning will not log Informational messages, a log level of Error will not log Information or Warning messages and a log level of Fatal will only log fatal message.

As loggers are setup and architecture specific, some setups may want to log to the event log, others to SQL or via the Enterprise Library the installer does not supply any loggers. The source download from CodePlex demonstrates how to write a logger, using the Windows Debug functions to log messages.

# The Microsoft SRE Inspectors

The following sections detail the configuration process and options available to you when using the inspectors provided with the SRE. The configuration process for third party plug-ins may vary and you should refer to the documentation for the third party plug-in you wish to configure.

## Enabling plug-in configuration

Each plug-in has its own configuration section name and class type, which is documented in the plug-in details below. To enable configuration you add a line to the configSections element in web.config. For example, to enable the AntiXSS Inspector settings you would add the following

<configSections>

…

**<section name="sreAntiXssSettings"  
 type="Microsoft.Security.Application.⮰  
SecurityRuntimeEngine.PlugIns.AntiXssInspectorSettings, ⮰  
Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns" />**

…

</configSections>

If you forget to add the section into the configSections list then configuration of the plug-ins will not take effect.

## Disabling an inspectors for a particular path

All of the Microsoft authored inspectors supports path exclusion, in much the same way as you exclude a path from all inspectors, by adding an excludedPath collection to the plug-in settings section. For example to exclude paths for the AntiXSS inspector you would have an sreAntiXSssSettings section like the following:

<sreAntiXssSettings>

**<excludedPaths>**

**<add path="/unprotected/example.aspx" />**

**<add path="/unprotected/anotherExample.aspx" />**

**</excludedPaths>**

</sreAntiXssSettings>

## The AntiXSS Inspector

The AntiXSS inspector examines any controls you have on a Web Forms page or user control and automatically encodes particular properties based on the control type, using the correct encoding mechanism for that property. This ensures these controls are protected against Cross Site Scripting attacks. You can also add control and encoding definitions for third party controls.

The AntiXSS inspector settings have a name of sreAntiXSSSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.AntiXssInspectorSettings.

The settings for the AntiXSS Inspector are as follows

* **filterDoubleEncoding**: a Boolean value controlling if control values are canonicalized before encoding, preventing double encoding errors. The default for this value is true.
* **encodeDerivedControls**: a Boolean value controlling the encoding process for derived controls. If set to true a web forms control which is derived from a known control will be encoded according to the rules of the parent control. The default for this value is true.
* **markAntiXssOutput**: a Boolean value indicating if automatically encoded values should be highlighted using the color specified in the markAntiXssColor setting.
* **markAntiXssColor**: a known color value (see <http://msdn.microsoft.com/en-us/library/system.drawing.color.aspx> for a list of colors) used to highlight encoded values if markAntiXssOutput is set to true.

An example of setting these values is as follows

<sreAntiXssSettings>

filterDoubleEncoding="true"

encodeDerivedControls="true"

markAntiXssOutput="true"

markAntiXssColor="Red">

<excludedPaths>

<add path="/unprotected/example.aspx" />

<add path="/unprotected/anotherExample.aspx" />

</excludedPaths>

</sreAntiXssSettings>

In addition to these settings the control encoding information is configurated via the encodingTypes collection. The encodingTypes collection is preloaded with settings for the standard web forms controls, detailed in Appendix A – Encoded Controls, Properties and Methods at the end of this document.

To add your own control encoding information you specify the full class name (namespace and class), the property name and the encoding context, which can be one of the following values

* Html
* HtmlAttribute
* Url
* Xml
* XmlAttribute
* SafeHtml

For example the following configuration snippet would add automatic HTML encoding to the Text property of the Microsoft.Example control;

<sreAntiXssSettings>

<encodingTypes>

<add fullClassName="Microsoft.Example"

property="Text"

encodingContext="Html" />

</encodingTypes>

</sreAntiXssSettings>

To remove an encoding you specify an id which is a combination of the full class name and the property, joined by a period; for example to remove the encoding information added in the snippet above you would use the following;

<sreAntiXssSettings>

<encodingTypes>

<remove id="Microsoft.Example.Text" />

</encodingTypes>

</sreAntiXssSettings>

You can also clear all encoding information by using the clear element;

<sreAntiXssSettings>

<encodingTypes>

<clear />

</encodingTypes>

</sreAntiXssSettings>

## The SQL Injection Inspector

The SQL Injection Inspector inspects input into your application via query strings, form fields, HTTP headers and cookies for potential SQL injection strings.

The SQL Injection inspector settings have a name of sreSqlInjectionSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.SqlInjecionInspectorSettings.

The settings for the SQL Injection Inspector are as follows:

* **inspectHeaders**: a Boolean value controlling if HTTP Headers are inspected. This defaults to true.
* **inspectCookies**: a Boolean value controlling if cookies are inspected.
* **ignoredFormParameters**: a collection of form field names which are excluded from checks. By default this collection consists of \_\_VIEWSTATE, \_\_EVENTTARGET and \_\_REQUESTDIGEST.
* **ignoredCookies**: a collection of cookie names which are excluded from checks. By default this collection is empty.

An example of setting these values is as follows

<sreSqlInjectionSettings>

inspectHeaders="true"

inspectCookies="true">

<ignoredFormParameters>

<add name="\_\_VIEWSTATE" />

<add name="\_\_EVENTTARGET" />

<add name="\_\_REQUESTDIGEST" />

</ignoredFormParameters>

<ignoredCookies>

<add name=".ASPXAUTH" />

</ignoredCookies>

<excludedPaths>

<add path="/unprotected/example.aspx" />

<add path="/unprotected/anotherExample.aspx" />

</excludedPaths>

</sreSqlInjectionSettings>

## The Credit Card Information Leak Inspector

The Credit Card Information Inspector Inspector inspects output from your application looking for full credit card numbers. It inspects the following types of output, based on the MIME type of the response:

* text/plain
* text/html
* text/xml
* application/xml
* application/rss+xml
* application/xhtml+xml

The Credit Card Information Leak inspector settings have a name of sreCCInspectorSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.CreditCardInspectorSettings. There is no configuration possible beyond the standard request path exclusions.

## The Fake Post-Back Inspector

The Fake Post Back Inspector inspects output inbound requests where the fields necessary to cause an ASP.NET post-back have been sent, but the request is not a POST request.

The Fake Post Back inspector settings have a name of sreFakeBackSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.FakeBackInspectorSettings. There is no configuration possible beyond the standard request path exclusions.

## The Click Jack Header Inspector

The Click Jack Header inspector examines application responses and inserts the X-FRAME-OPTIONS header if it is not present. More details on this header can be found at <http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx>.

The Click Jack inspector settings have a name of sreClickJackSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.ClickJackInspectorSettings. In addition to request path exclusion the settings for this plug-in allow the header value can be set using the headerValue attribute to either Deny or SameOrigin. The default value is Deny.

An example of setting this value is as follows

<sreClickJackSettings

headerValue="Deny">

<excludedPaths>

<add path="/unprotected/example.aspx" />

<add path="/unprotected/anotherExample.aspx" />

</excludedPaths>

</sreClickJackSettings>

This inspector will not stop processing of a request or response.

## The Cookie Protection Inspector

The Cookie Protection Inspector examines application responses and marks cookies as HTTP-Only to protect them from being accessed via JavaScript and potentially vulnerable to Cross Site Scripting attacks.

The Cookie Protection inspector settings have a name of sreCookieSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.CookieProtectionInspectorSettings. In addition to request path exclusion the settings for this plugin allow a list of cookie names to be excluded from project.

An example of setting exclusions is as follows

<sreCookieSettings>

<excludedCookies>

<add name=".ASPXAUTH" />

<excludedCookies>

<excludedPaths>

<add path="/unprotected/example.aspx" />

<add path="/unprotected/anotherExample.aspx" />

</excludedPaths>

</sreCookieSettings>

This inspector will not stop processing of a request or response.

## The No Open Header Inspector

The No Open Header Inspector examines application responses and adds an X-Download-Options header, which stops browsers displaying an Open button on the dialog that appears when a file is downloaded. You can read more about this header at <http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-v-comprehensive-protection.aspx>.

The No Open Header inspector settings have a name of sreNoOpenSettings and a class of Microsoft.Security.Application.SecurityRuntimeEngine.PlugIns.NoOPenHeaderInspectorSettings. There is no configuration possible beyond the standard request path exclusions.

This inspector will not stop processing of a request or response.

# What next?

We’ve released the September CTP source drop in order to gather your feedback about where we are taking the SRE and how the standard inspectors work for you. We would like you to look at the new SRE, play with it, run it on your development and staging servers, even write your own inspectors. Please give us any feedback, and details of bugs you may find, on the [WPL CodePlex site](http://wpl.codeplex.com/Thread/List.aspx). We don’t promise to change the WPL in the exact way you want, but we do promise to read every piece of feedback we get.

With your feedback we will move from CTP to a Beta to a full release, both source and binaries.

Thanks for taking the time to download the SRE and for any feedback you can give us. We look forward to helping you protect your ASP.NET applications!

## Appendix A – Encoded Controls, Properties and Methods

|  |  |  |
| --- | --- | --- |
| **Control** | **Property** | **Encoding** |
| HtmlControls.HtmlAnchor | HRef | Html Attribute |
| HtmlControls.HtmlHead | Title | Html |
| HtmlControls.HtmlImage | Src | Html Attribute |
| HtmlControls.HtmlInputImage | Src | Html Attribute |
| HtmlControls.HtmlInputRadioButton | Value | Html Attribute |
| WebControls.BaseDataList | Caption | Html Attribute |
| WebControls.Calendar | Caption | Html |
| WebControls.Calendar | NextMonthText | Html |
| WebControls.Calendar | PrevMonthText | Html |
| WebControls.Calendar | SelectMonthText | Html |
| WebControls.Calendar | SelectWeekText | Html |
| WebControls.ChangePassword | CancelDestinationPageUrl | URL |
| WebControls.ChangePassword | ChangePasswordFailureText | Html |
| WebControls.ChangePassword | ChangePasswordTitleText | Html |
| WebControls.ChangePassword | ConfirmNewPasswordLabelText | Html |
| WebControls.ChangePassword | ContinueDestinationPageUrl | URL |
| WebControls.ChangePassword | CreateUserText | Html |
| WebControls.ChangePassword | EditProfileText | Html |
| WebControls.ChangePassword | HelpPageText | Html |
| WebControls.ChangePassword | InstructionText | Html |
| WebControls.ChangePassword | NewPasswordLabelText | Html |
| WebControls.ChangePassword | PasswordHintText | Html |
| WebControls.ChangePassword | PasswordLabelText | Html |
| WebControls.ChangePassword | PasswordRecoveryText | Html |
| WebControls.ChangePassword | SuccessPageUrl | URL |
| WebControls.ChangePassword | SuccessText | Html |
| WebControls.ChangePassword | SuccessTitleText | Html |
| WebControls.ChangePassword | UserNameLabelText | Html |
| WebControls.CheckBox | Text | Html |
| WebControls.CompareValidator | Text | Html |
| WebControls.CreateUserWizard | AnswerLabelText | Html |
| WebControls.CreateUserWizard | CompleteSuccessText | Html |
| WebControls.CreateUserWizard | ConfirmPasswordLabelText | Html |
| WebControls.CreateUserWizard | ContinueDestinationPageUrl | URL |
| WebControls.CreateUserWizard | DuplicateEmailErrorMessage | Html |
| WebControls.CreateUserWizard | DuplicateUserNameErrorMessage | Html |
| WebControls.CreateUserWizard | EditProfileText | Html |
| WebControls.CreateUserWizard | EmailLabelText | Html |
| WebControls.CreateUserWizard | UnknownErrorMessage | Html |
| WebControls.CreateUserWizard | HelpPageText | Html |
| WebControls.CreateUserWizard | InstructionText | Html |
| WebControls.CreateUserWizard | InvalidAnswerErrorMessage | Html |
| WebControls.CreateUserWizard | InvalidEmailErrorMessage | Html |
| WebControls.CreateUserWizard | InvalidPasswordErrorMessage | Html |
| WebControls.CreateUserWizard | InvalidQuestionErrorMessage | Html |
| WebControls.CreateUserWizard | PasswordHintText | Html |
| WebControls.CreateUserWizard | PasswordLabelText | Html |
| WebControls.CreateUserWizard | QuestionLabelText | Html |
| WebControls.CreateUserWizard | UserNameLabelText | Html |
| WebControls.CreateUserWizard | CancelDestinationPageUrl | URL |
| WebControls.CreateUserWizard | FinishDestinationPageUrl | URL |
| WebControls.CreateUserWizard | HeaderText | Html |
| WebControls.CustomValidator | Text | Html |
| WebControls.DataControlFieldCell | Text | Html |
| WebControls.DataControlFieldHeaderCell | Text | Html |
| WebControls.DataGrid | Caption | Html |
| WebControls.DataList | Caption | Html |
| WebControls.DetailsView | Caption | Html |
| WebControls.DetailsView | EmptyDataText | Html |
| WebControls.DetailsView | FooterText | Html |
| WebControls.DetailsView | HeaderText | Html |
| WebControls.FormView | Caption | Html |
| WebControls.FormView | EmptyDataText | Html |
| WebControls.FormView | FooterText | Html |
| WebControls.FormView | HeaderText | Html |
| WebControls.GridView | Caption | Html |
| WebControls.GridView | EmptyDataText | Html |
| WebControls.HyperLink | Text | Html |
| WebControls.Label | Text | Html |
| WebControls.LinkButton | Text | Html |
| WebControls.ListBox | Text | Html |
| WebControls.ListControl | Text | Html |
| WebControls.Literal | Text | Html |
| WebControls.Login | CreateUserText | Html |
| WebControls.Login | DestinationPageUrl | URL |
| WebControls.Login | HelpPageText | Html |
| WebControls.Login | InstructionText | Html |
| WebControls.Login | FailureText | Html |
| WebControls.Login | LoginButtonText | Html |
| WebControls.Login | PasswordLabelText | Html |
| WebControls.Login | PasswordRecoveryText | Html |
| WebControls.Login | RememberMeText | Html |
| WebControls.Login | TitleText | Html |
| WebControls.Login | UserNameLabelText | Html |
| WebControls.LoginStatus | LoginText | Html |
| WebControls.LoginStatus | LogoutPageUrl | URL |
| WebControls.LoginStatus | LogoutText | Html |
| WebControls.Panel | GroupingText | Html |
| WebControls.PasswordRecovery | AnswerLabelText | Html |
| WebControls.PasswordRecovery | GeneralFailureText | Html |
| WebControls.PasswordRecovery | HelpPageText | Html |
| WebControls.PasswordRecovery | QuestionFailureText | Html |
| WebControls.PasswordRecovery | QuestionInstructionText | Html |
| WebControls.PasswordRecovery | QuestionLabelText | Html |
| WebControls.PasswordRecovery | QuestionTitleText | Html |
| WebControls.PasswordRecovery | SubmitButtonText | Html |
| WebControls.PasswordRecovery | SuccessPageUrl | URL |
| WebControls.PasswordRecovery | SuccessText | Html |
| WebControls.PasswordRecovery | UserNameFailureText | Html |
| WebControls.PasswordRecovery | UserNameInstructionText | Html |
| WebControls.PasswordRecovery | UserNameLabelText | Html |
| WebControls.PasswordRecovery | UserNameTitleText | Html |
| WebControls.RadioButton | Text | Html |
| WebControls.RadioButtonList | Text | Html |
| WebControls.RangeValidator | Text | Html |
| WebControls.RegularExpressionValidator | Text | Html |
| WebControls.RequiredFieldValidator | Text | Html |
| WebControls.Table | Caption | Html |
| WebControls.TableCell | Text | Html |
| WebControls.TableHeaderCell | Text | Html |
| WebControls.WebParts.AppearanceEditorPart | GroupingText | Html |
| WebControls.WebParts.BehaviorEditorPart | Title | Html |
| WebControls.WebParts.BehaviorEditorPart | GroupingText | Html |
| WebControls.WebParts.CatalogPart | Title | Html |
| WebControls.WebParts.CatalogPart | GroupingText | Html |
| WebControls.WebParts.CatalogZone | EmptyZoneText | Html |
| WebControls.WebParts.CatalogZone | HeaderText | Html |
| WebControls.WebParts.CatalogZone | InstructionText | Html |
| WebControls.WebParts.CatalogZone | SelectTargetZoneText | Html |
| WebControls.WebParts.CatalogZoneBase | EmptyZoneText | Html |
| WebControls.WebParts.CatalogZoneBase | HeaderText | Html |
| WebControls.WebParts.CatalogZoneBase | InstructionText | Html |
| WebControls.WebParts.CatalogZoneBase | SelectTargetZoneText | Html |
| WebControls.WebParts.ConnectionsZone | ConfigureConnectionTitle | Html |
| WebControls.WebParts.ConnectionsZone | ConnectToConsumerInstructionText | Html |
| WebControls.WebParts.ConnectionsZone | ConnectToConsumerText | Html |
| WebControls.WebParts.ConnectionsZone | ConnectToConsumerTitle | Html |
| WebControls.WebParts.ConnectionsZone | ConnectToProviderInstructionText | Html |
| WebControls.WebParts.ConnectionsZone | ConnectToProviderText | Html |
| WebControls.WebParts.ConnectionsZone | ConnectToProviderTitle | Html |
| WebControls.WebParts.ConnectionsZone | ConsumersTitle | Html |
| WebControls.WebParts.ConnectionsZone | ConsumersInstructionText | Html |
| WebControls.WebParts.ConnectionsZone | EmptyZoneText | Html |
| WebControls.WebParts.ConnectionsZone | ExistingConnectionErrorMessage | Html |
| WebControls.WebParts.ConnectionsZone | GetText | Html |
| WebControls.WebParts.ConnectionsZone | GetFromText | Html |
| WebControls.WebParts.ConnectionsZone | HeaderText | Html |
| WebControls.WebParts.ConnectionsZone | InstructionText | Html |
| WebControls.WebParts.ConnectionsZone | InstructionTitle | Html |
| WebControls.WebParts.ConnectionsZone | NewConnectionErrorMessage | Html |
| WebControls.WebParts.ConnectionsZone | NoExistingConnectionInstructionText | Html |
| WebControls.WebParts.ConnectionsZone | NoExistingConnectionTitle | Html |
| WebControls.WebParts.ConnectionsZone | ProvidersTitle | Html |
| WebControls.WebParts.ConnectionsZone | ProvidersInstructionText | Html |
| WebControls.WebParts.ConnectionsZone | SendText | Html |
| WebControls.WebParts.ConnectionsZone | SendToText | Html |
| WebControls.WebParts.DeclarativeCatalogPart | Title | Html |
| WebControls.WebParts.DeclarativeCatalogPart | GroupingText | Html |
| WebControls.WebParts.EditorPart | Title | Html |
| WebControls.WebParts.EditorPart | GroupingText | Html |
| WebControls.WebParts.EditorZone | EmptyZoneText | Html |
| WebControls.WebParts.EditorZone | ErrorText | Html |
| WebControls.WebParts.EditorZone | HeaderText | Html |
| WebControls.WebParts.EditorZone | InstructionText | Html |
| WebControls.WebParts.EditorZoneBase | EmptyZoneText | Html |
| WebControls.WebParts.EditorZoneBase | ErrorText | Html |
| WebControls.WebParts.EditorZoneBase | HeaderText | Html |
| WebControls.WebParts.EditorZoneBase | InstructionText | Html |
| WebControls.WebParts.ErrorWebPart | AuthorizationFilter | Html |
| WebControls.WebParts.ErrorWebPart | ImportErrorMessage | Html |
| WebControls.WebParts.ErrorWebPart | Title | Html |
| WebControls.WebParts.ErrorWebPart | GroupingText | Html |
| WebControls.WebParts.GenericWebPart | Title | Html |
| WebControls.WebParts.GenericWebPart | AuthorizationFilter | Html |
| WebControls.WebParts.GenericWebPart | ImportErrorMessage | Html |
| WebControls.WebParts.GenericWebPart | GroupingText | Html |
| WebControls.WebParts.ImportCatalogPart | BrowseHelpText | Html |
| WebControls.WebParts.ImportCatalogPart | ImportedPartLabelText | Html |
| WebControls.WebParts.ImportCatalogPart | PartImportErrorLabelText | Html |
| WebControls.WebParts.ImportCatalogPart | Title | Html |
| WebControls.WebParts.ImportCatalogPart | UploadHelpText | Html |
| WebControls.WebParts.ImportCatalogPart | GroupingText | Html |
| WebControls.WebParts.LayoutEditorPart | Title | Html |
| WebControls.WebParts.LayoutEditorPart | GroupingText | Html |
| WebControls.WebParts.PageCatalogPart | Title | Html |
| WebControls.WebParts.PageCatalogPart | GroupingText | Html |
| WebControls.WebParts.Part | Title | Html |
| WebControls.WebParts.Part | GroupingText | Html |
| WebControls.WebParts.PropertyGridEditorPart | Title | Html |
| WebControls.WebParts.PropertyGridEditorPart | GroupingText | Html |
| WebControls.WebParts.ProxyWebPart | AuthorizationFilter | Html |
| WebControls.WebParts.ProxyWebPart | ImportErrorMessage | Html |
| WebControls.WebParts.ProxyWebPart | Title | Html |
| WebControls.WebParts.ProxyWebPart | GroupingText | Html |
| WebControls.WebParts.ToolZone | InstructionText | Html |
| WebControls.WebParts.ToolZone | EmptyZoneText | Html |
| WebControls.WebParts.ToolZone | HeaderText | Html |
| WebControls.WebParts.UnauthorizedWebPart | AuthorizationFilter | Html |
| WebControls.WebParts.UnauthorizedWebPart | ImportErrorMessage | Html |
| WebControls.WebParts.UnauthorizedWebPart | Title | Html |
| WebControls.WebParts.UnauthorizedWebPart | GroupingText | Html |
| WebControls.WebParts.WebPart | AuthorizationFilter | Html |
| WebControls.WebParts.WebPart | ImportErrorMessage | Html |
| WebControls.WebParts.WebPart | Title | Html |
| WebControls.WebParts.WebPart | GroupingText | Html |
| WebControls.WebParts.WebPartZone | EmptyZoneText | Html |
| WebControls.WebParts.WebPartZone | MenuLabelText | Html |
| WebControls.WebParts.WebPartZone | HeaderText | Html |
| WebControls.WebParts.WebPartZoneBase | EmptyZoneText | Html |
| WebControls.WebParts.WebPartZoneBase | MenuLabelText | Html |
| WebControls.WebParts.WebPartZoneBase | HeaderText | Html |
| WebControls.WebParts.WebZone | EmptyZoneText | Html |
| WebControls.WebParts.WebZone | HeaderText | Html |
| WebControls.Wizard | CancelDestinationPageUrl | URL |
| WebControls.Wizard | FinishDestinationPageUrl | URL |
| WebControls.Wizard | HeaderText | Html |