This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework 4 - ASP.NET

**Walkthrough: Encrypting Configuration Information Using Protected Configuration**

Provides a step-by-step example for encrypting sections of a configuration file for an ASP.NET application.

Protected Configuration helps improve the security of an application by letting you encrypt sensitive information that is stored in a Web.config file. You can use aspnet\_regiis.exe to encrypt sections of the Web.config file and manage encryption keys. ASP.NET decrypts the configuration file when it processes the file. Therefore, decryption does not require any additional code. For more information about Protected Configuration, see [Encrypting Configuration Information Using Protected Configuration](http://msdn.microsoft.com/en-us/library/53tyfkaw.aspx).

During this walkthrough, you will learn how to do the following:

* Encrypt sections of a Web.config file by using the default Protected Configuration provider.
* Access decrypted configuration information in an ASP.NET page.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifPrerequisites

In order to complete this walkthrough, you will need the following:

* Microsoft Internet Information Services (IIS) installed and configured on the computer that will host the site.

For detailed information about how to install and configure IIS, see the online Help that is included with the IIS installation or go to [Internet Information Services (IIS) 6.0 Technical Resources](http://go.microsoft.com/fwlink/?linkid=37112).

* An ASP.NET Web site.

If you already have a Web site, you can use that site. Otherwise, for detailed information about how to create a virtual directory or Web site, see [How to: Create and Configure Virtual Directories in IIS 5.0 and 6.0](http://msdn.microsoft.com/en-us/library/zwk103ab.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifGranting Read Access to an RSA Encryption Key

Before ASP.NET can decrypt encrypted information in the Web.config file, the identity of your ASP.NET application must have read access to the encryption key that is used to encrypt and decrypt the encrypted sections. This walkthrough uses the default [RsaProtectedConfigurationProvider](http://msdn.microsoft.com/en-us/library/system.configuration.rsaprotectedconfigurationprovider.aspx) provider that is specified in the Machine.config file and named "RsaProtectedConfigurationProvider". The RSA key container that is used by the default [RsaProtectedConfigurationProvider](http://msdn.microsoft.com/en-us/library/system.configuration.rsaprotectedconfigurationprovider.aspx) provider is named "NetFrameworkConfigurationKey".

**To grant the ASP.NET identity read access to the default RSA key container**

1. Open a text editor, and then copy the following code into a new file.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_ctl00_ctl02_code');" \o "Copy Code)

<%@ Page Language="VB" %>

<%

Response.Write(System.Security.Principal.WindowsIdentity.GetCurrent().Name)

%>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl26_ctl00_ctl03_code');" \o "Copy Code)

<%@ Page Language="C#" %>

<%

Response.Write(System.Security.Principal.WindowsIdentity.GetCurrent().Name);

%>

1. Save the file in your application directory as **identity.aspx**.
2. To determine the identity of your ASP.NET application, open identity.aspx in a browser.

The impersonated identity of your ASP.NET application is displayed in the browser.

|  |
| --- |
| **Description: NoteNote** |
| Because you are using IIS for this walkthrough, do not use impersonation for the authentication for your site. That is, for this walkthrough, use only anonymous authentication as the identity of your ASP.NET application. If the identity of your application is the user ID that you are currently logged on as, such as DOMAIN\*myuserid*, in the Web.config file for the application, disable impersonation. To disable impersonation in the Web.config file, open the Web.config file, and then remove the <identity> element. After you remove the <identity> element, update identity.aspx in your browser to display the modified identity for the application. |

1. At the command prompt, run aspnet\_regiis.exe with the following options:
   * The **-pa** option, followed by the name of the RSA key container for the default [RsaProtectedConfigurationProvider](http://msdn.microsoft.com/en-us/library/system.configuration.rsaprotectedconfigurationprovider.aspx).
   * The identity of your ASP.Net application, as determined in the preceding step.

For example, the following command grants the NETWORK SERVICE account access to the machine-level "NetFrameworkConfigurationKey" RSA key container.

|  |
| --- |
| **Description: NoteNote** |
| On a computer that is running Microsoft Windows Server 2003 with impersonation for an ASP.NET application that is disabled in the Web.config file, the identity for the application is the identity of the application pool. By default, this is the NETWORK SERVICE account (on earlier versions of Windows, the identity is the local ASPNET account). |

**aspnet\_regiis -pa "NetFrameworkConfigurationKey" "NT AUTHORITY\NETWORK SERVICE"**

Do not close the Command Prompt window.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifEncrypting Sections of the Web.config File

Now that the identity of your ASP.NET application has read access to the RSA key container for the default [RsaProtectedConfigurationProvider](http://msdn.microsoft.com/en-us/library/system.configuration.rsaprotectedconfigurationprovider.aspx) object, you will encrypt sections of the Web.config file for your ASP.NET application by using that key container. Then, ASP.NET decrypts the sections when it processes the Web.config file.

**To encrypt the <connectionStrings> and <machineKey> sections of the Web.config file**

1. In a text editor, open the Web.config file for your application.
   * If you do not have a Web.config file for your ASP.NET application, open a text editor, copy the example configuration into a new file, and then save the file in your ASP.NET application directory as **web.config**.
2. Make sure that there is both a <connectionStrings> child element and a <machineKey> child element for the <system.web> element, as shown in the following example.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl27_ctl00_ctl01_code');" \o "Copy Code)

<configuration>

<connectionStrings>

<add name="SqlServices" connectionString="Data Source=localhost;Integrated Security=SSPI;Initial Catalog=Northwind;" />

</connectionStrings>

<system.web>

<machineKey validationKey="D61B3C89CB33A2F1422FF158AFF7320E8DB8CB5CDA1742572A487D94018787EF42682B202B746511891C1BAF47F8D25C07F6C39A104696DB51F17C529AD3CABE"

decryptionKey="FBF50941F22D6A3B229EA593F24C41203DA6837F1122EF17" />

</system.web>

</configuration>

1. Close the Web.config file.
2. At the command prompt, change the directory to the .NET Framework version 2.0 directory by typing the following command:

**cd \WINDOWS\Microsoft.Net\Framework\v2.0.\***

1. At the command prompt, run aspnet\_regiis.exe with the following options:
   * The **-pe** option and the string **"connectionStrings"** to encrypt the **connectionStrings** element of the Web.config file for your application.
   * The **-app** option and the name of your application.

For example, the following command encrypts the <connectionStrings> section of the Web.config file for an application named MyApplication.

**aspnet\_regiis -pe "connectionStrings" -app "/MyApplication"**

1. Repeat the preceding step for the <machineKey> child element of the <system.web> element, as shown in the following example:

**aspnet\_regiis -pe "system.web/machineKey" -app "/MyApplication"**

Do not close the Command Prompt window.

1. Open Web.config, and then view the encrypted contents.

The contents will look similar to the following example Web.config file.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl27_ctl00_ctl02_code');" \o "Copy Code)

<configuration>

<connectionStrings configProtectionProvider="RsaProtectedConfigurationProvider">

<EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"

xmlns="http://www.w3.org/2001/04/xmlenc#">

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />

<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

<EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5" />

<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

<KeyName>RSA Key

</KeyName>

</KeyInfo>

<CipherData>

<CipherValue>WcFEbDX8VyLfAsVK8g6hZVAG1674ZFc1kWH0BoazgOwdBfinhcAmQmnIn0oHtZ5tO2EXGl+dyh10giEmO9NemH4YZk+iMIln+ItcEay9CGWMXSen9UQLpcQHQqMJErZiPK4qPZaRWwqckLqriCl9X8x9OE7jKIsO2Ibapwj+1Jo=

</CipherValue>

</CipherData>

</EncryptedKey>

</KeyInfo>

<CipherData>

<CipherValue>

</CipherValue>

</CipherData>

</EncryptedData>

</connectionStrings>

<system.web>

<machineKey configProtectionProvider="RsaProtectedConfigurationProvider">

<EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"

xmlns="http://www.w3.org/2001/04/xmlenc#">

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />

<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

<EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">

<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5" />

<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">

<KeyName>RSA Key

</KeyName>

</KeyInfo>

<CipherData>

<CipherValue>IwUopItbWX0mJdGWtAqE1LlsG3u5RBRlAXs9/GZj3HEfeUXduHVF76q6Ip88YqlfLthH+DMBYdOZAF+hCOmS2agfTo1tKUvELRGIljS/BqEYxUO+/IOz9tllAw8ZlGF7AVCzptgIejI+iLXEZfMKW7f6EMGeb5vaaKXHIkYZwcM=

</CipherValue>

</CipherData>

</EncryptedKey>

</KeyInfo>

<CipherData>

<CipherValue>ivVyERVPNUzIb/i7/NUbRkxsxh8IG959vycwrzJO0vYWxHZ5i03SfrLbsGUV17+FxZ6lbcrVaF5FY3zVm7dRMRvQpVFwaVcL

</CipherValue>

</CipherData>

</EncryptedData>

</machineKey>

</system.web>

</configuration>

1. Close the Web.config file.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifAccessing Decrypted Configuration Settings

ASP.NET automatically decrypts the contents of the Web.config file when it processes the file. Therefore, no additional steps are required to decrypt the encrypted configuration settings for use by other ASP.NET features or to access the values in your code. However, you can follow these steps, if you want to view the decrypted settings.

**To view the decrypted configuration values**

1. Open a text editor, and then copy the following ASP.NET code into a new file.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl28_ctl00_ctl00_code');" \o "Copy Code)

<%@ Page Language="VB" %>

<%@ Import Namespace="System.Configuration" %>

<%@ Import Namespace="System.Web.Configuration" %>

<script runat="server">

Public Sub Page\_Load()

ConnectionStringsGrid.DataSource = ConfigurationManager.ConnectionStrings

ConnectionStringsGrid.DataBind()

Dim config As System.Configuration.Configuration = \_

WebConfigurationManager.OpenWebConfiguration(Request.ApplicationPath)

Dim key As MachineKeySection = \_

CType(config.GetSection("system.web/machineKey"), MachineKeySection)

DecryptionKey.Text = key.DecryptionKey

ValidationKey.Text = key.ValidationKey

End Sub

</script>

<html>

<body>

<form runat="server">

<asp:GridView runat="server" CellPadding="4" id="ConnectionStringsGrid" />

<P>

MachineKey.DecryptionKey = <asp:Label runat="Server" id="DecryptionKey" /><BR>

MachineKey.ValidationKey = <asp:Label runat="Server" id="ValidationKey" />

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl28_ctl00_ctl01_code');" \o "Copy Code)

<%@ Page Language="C#" %>

<%@ Import Namespace="System.Configuration" %>

<%@ Import Namespace="System.Web.Configuration" %>

<script runat="server">

public void Page\_Load()

{

ConnectionStringsGrid.DataSource = ConfigurationManager.ConnectionStrings;

ConnectionStringsGrid.DataBind();

Configuration config = WebConfigurationManager.OpenWebConfiguration(Request.ApplicationPath);

MachineKeySection key =

(MachineKeySection)config.GetSection("system.web/machineKey");

DecryptionKey.Text = key.DecryptionKey;

ValidationKey.Text = key.ValidationKey;

}

</script>

<html>

<body>

<form runat="server">

<asp:GridView runat="server" CellPadding="4" id="ConnectionStringsGrid" />

<P>

MachineKey.DecryptionKey = <asp:Label runat="Server" id="DecryptionKey" /><BR>

MachineKey.ValidationKey = <asp:Label runat="Server" id="ValidationKey" />

</form>

</body>

</html>

1. Save the file as **walkthrough.aspx,** and then view the file in the browser.

You will see the decrypted values from your encrypted Web.config file.

1. To keep the sensitive information from the Web site private, delete the walkthrough.aspx file when you are finished with this walkthrough.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifNext Steps

You can decrypt the encrypted Web.config file contents, if you want to, by running aspnet\_regiis.exe with the **-pd** option. The syntax is the same as the syntax for encrypting Web.config file contents with the **-pe** option, except that you do not specify a Protected Configuration provider. The appropriate provider is identified using the **configProtectionProvider** element for the **protected** section. For example, the following commands decrypt the <connectionStrings> element and the <machineKey> child element of the <system.web> element in the Web.config file for the ASP.NET application named MyApplication.

**aspnet\_regiis -pd "connectionStrings" -app "/MyApplication"**

**aspnet\_regiis -pd "system.web/machineKey" -app "/MyApplication"**