This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework 4 - ASP.NET

**Walkthrough: Creating and Exporting an RSA Key Container**

Provides a step-by-step example for creating, exporting, and importing an encryption key to decrypt encrypted Web.config sections from a single encrypted file on multiple servers.

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

Protected Configuration lets you create, delete, export, and import custom encryption keys for use with the RSAProtectedConfigurationProvider provider. This lets you create backup copies of encryption keys or copy an encryption key to multiple Web servers, such as a Web farm, so that an application that has an encrypted Web.config file can be copied to multiple locations.

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

* Create a custom RSA key container.
* Specify a Protected Configuration provider that uses a custom RSA key container.
* Encrypt sections of a Web.config file by using a custom RSA key container.
* Export a custom RSA key container to an XML file.
* Import a custom RSA key container from an XML file.

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 it for this walkthrough. 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.gifCreating a Custom RSA Key Container

In this part of the walkthrough, you will create an RSA key container by using aspnet\_regiis.exe with the **-pc** option. This identifies the RSA key container as a user-level key container. RSA key containers must be identified as either user-level (by using the **-pku** option) or machine-level (by not using the **-pku** option). For more information about machine-level and user-level RSA key containers, see [Understanding Machine-Level and User-Level RSA Key Containers](http://msdn.microsoft.com/en-us/library/f5cs0acs.aspx).

**To create a machine-level RSA key container**

1. Open a command prompt.
   * To do this, in Microsoft Windows, click **Start**, click **Run**, in the **Open** box, type **cmd,** and then click **OK**.
2. At the command prompt, enter the following command to change the directory to the .NET Framework version 2.0 directory:

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

1. Create a new, machine-level RSA key container by running aspnet\_regiis.exe with the following options:
   * The **-pc** option followed by the name of the RSA key container, to create the RSA key pair.
   * The **-exp** option, to make sure that the key is exportable.

The following command will create the **"MyKeys"** key container.

**aspnet\_regiis -pc "MyKeys" -exp**

Do not close the Command Prompt window.

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

Before ASP.NET can decrypt encrypted information that is 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.

**To grant the ASP.NET identity access to the 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_ctl29_ctl00_ctl00_code');" \o "Copy Code)

<%@ Page Language="VB" %>

<%

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

%>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl01_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 appears in the browser.

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

1. At the command prompt, use the following options to grant the identity access to the RSA key container by running aspnet\_regiis.exe:
   * The **-pa** option followed by the RSA key container name **"MyKeys"**.
   * 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 RSA key container **"MyKeys"**.

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

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

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifSpecifying a Protected Configuration Provider

In this part of the walkthrough, you will specify an instance of a Protected Configuration provider in the Web.config file for your ASP.NET application. The instance of a Protected Configuration provider that uses the machine-level RSA key container named "MyKeys" is the one that you created in the preceding procedure.

**To specify an instance of the Protected Configuration provider**

1. Open a text editor, and then open the Web.config file for your ASP.NET application.
   * If you do not have a Web.config file for your ASP.NET application, open a text editor, and then copy the example configuration into a new file. Save the file in your ASP.NET application directory as **web.config**.
2. Make sure that the configuration includes a <connectionStrings> element, as shown in the following example.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl30_ctl00_ctl00_code');" \o "Copy Code)

<configuration>

<connectionStrings>

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

</connectionStrings>

</configuration>

1. Add a <configProtectedData> section that includes an instance of the RSAProtectedConfigurationProvider class named "MyProvider" that uses the machine-level RSA key container named "MyKeys", as shown in the following example.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl30_ctl00_ctl01_code');" \o "Copy Code)

<configuration>

<configProtectedData>

<providers>

<add name="MyProvider"

type="System.Configuration.RsaProtectedConfigurationProvider, System.Configuration, Version=2.0.0.0,

Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a,

processorArchitecture=MSIL"

keyContainerName="MyKeys"

useMachineContainer="true" />

</providers>

</configProtectedData>

<connectionStrings>

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

</connectionStrings>

</configuration>

1. Save, and then close the Web.config file.

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

Now that you have specified an instance of the RSAProtectedConfigurationProvider class that uses the "MyKeys" RSA key container and the identity of your ASP.NET application has read access to "MyKeys", you will encrypt sections of the Web.config file for your ASP.NET application by using "MyKeys", and then ASP.NET will decrypt the sections when it processes the Web.config file.

**To encrypt the <connectionStrings> section of the Web.config file**

1. At the command prompt, run aspnet\_regiis.exe with the following options:
   * The **-pe** option, followed by **"connectionStrings"**, to encrypt the <connectionStrings> element of the Web.config file for your application.
   * The **-app** option, to identify the name of your application.
   * The **-prov** option, followed by **"MyProvider"**, to identify the RSAProtectedConfigurationProvider provider that was specified in the Web.config file in the preceding section.

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" -prov "MyProvider"**

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

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

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl31_ctl00_ctl00_code');" \o "Copy Code)

<configuration>

<configProtectedData>

<providers>

<add name="MyProvider"

type="System.Configuration.RsaProtectedConfigurationProvider, System.Configuration, Version=2.0.0.0,

Culture=neutral, PublicKeyToken= b03f5f7f11d50a3a,

processorArchitecture=MSIL"

keyContainerName="MyKeys"

useMachineContainer="true" />

</providers>

</configProtectedData>

<connectionStrings configProtectionProvider="MyProvider">

<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>

</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 when it processes the file. Therefore, no 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_ctl32_ctl00_ctl00_code');" \o "Copy Code)

<%@ Page Language="VB" %>

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

<script runat="server">

Public Sub Page\_Load()

ConnectionStringsGrid.DataSource = ConfigurationManager.ConnectionStrings

ConnectionStringsGrid.DataBind()

End Sub

</script>

<html>

<body>

<form runat="server">

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

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl32_ctl00_ctl01_code');" \o "Copy Code)

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

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

<script runat="server">

public void Page\_Load()

{

ConnectionStringsGrid.DataSource = ConfigurationManager.ConnectionStrings;

ConnectionStringsGrid.DataBind();

}

</script>

<html>

<body>

<form runat="server">

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

</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.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExporting and Importing an RSA Key Container

You can export an RSA key container to an XML file as a backup copy of the key values or copy the key container to additional Web servers that will host a copy of the application that includes the encrypted Web.config file. Without the specific RSA key container that was used to encrypt the Web.config file, ASP.NET will not be able to decrypt the encrypted configuration values.

**To view the decrypted configuration values**

1. At the command prompt, run aspnet\_regiis.exe with the following options:
   * The **-px** option, followed by **"MyKeys"**, which is the name of the RSA key container that you created in "Creating a Custom RSA Key Container," earlier in this walkthrough.
   * The path of an .xml file to export the key container to.
   * The **-pri** option, to make sure that private key information is exported. Otherwise, the exported key information will only encrypt information, not decrypt it.

For example, the following command exports the machine-level RSA key container named **"MyKeys"** to an .xml file named keys.xml in the root directory of drive C.

**aspnet\_regiis -px "MyKeys" "c:\keys.xml" -pri**

|  |
| --- |
| **Description: NoteNote** |
| To make sure that no one can decrypt the Web.config files that are encrypted by RSA key container, after you export the RSA key container to an .xml file, copy the .xml file to a location that is external to the Web server, and then delete the file from the Web server. |

You now have all information that is required to copy the application by using the encrypted Web.config file to a separate Web server.

1. If you want to copy the application by using the encrypted Web.config file to a separate Web server, go to step 4.
2. If you do not have a second Web server to copy the Web application to, and you want to continue with this walkthrough, complete the following steps to delete the RSA key container from the Web server. Then, treat the Web server as if it were the second Web server.
   * To delete the RSA key container, at the command prompt, run aspnet\_regiis.exe with the **-pz** switch, followed by **"MyKeys"**.

For example, the following command deletes **"MyKeys"**:

**aspnet\_regiis -pz "MyKeys"**

* + Go to step 5.

1. Copy the Web application, including the encrypted Web.config file, to a second Web server.
   * If you are not sure about how to copy the Web application to a second server, copy the whole folder and contents from your existing application to the second Web server, and then to identify your application as a Web application, follow the steps in [How to: Create and Configure Virtual Directories in IIS 5.0 and 6.0](http://msdn.microsoft.com/en-us/library/zwk103ab.aspx).
2. On the second server, open a Command Prompt window. and then enter the following command to change the directory to the .NET Framework version 2.0 directory:

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

1. Copy the .xml file that contains the exported RSA key container to the .NET Framework version 2.0 directory on the second Web server.

In this walkthrough, you will copy the **keys.xml** file to the root directory of drive C.

1. On the second Web server, at the command prompt, run aspnet\_regiis.exe with the following options:
   * The **-pi** option, followed by **"MyKeys"**, which is the name of the exported key container, to import the RSA key container.
   * The path of the .xml file that contains the exported key container

For example, the following command imports the RSA key container named **"MyKeys"**.

**aspnet\_regiis -pi "MyKeys" "c:\keys.xml"**

1. On the second Web server, delete the copy of the .xml file that contains the exported RSA key container.
2. On the second Web server, determine the identity of the ASP.NET application, and then grant that identity access to the imported RSA key container by following the steps in "Granting Read Access to an RSA Encryption Key," earlier in this walkthrough.
3. On the second Web server, view the decrypted configuration settings from the encrypted Web.config file by following the steps in the preceding section.