Configuring Many-to-One Client Certificate Mappings for IIS 7/7.5

Posted: Apr 27, 2010 2 comments

Average Rating

Tags Authentication Client Certificate IIS7 IIS7.5

Many-to-one Client certificate mapping is used by the Internet Information Services (IIS) to associate an end user to a windows account when the client certificate is used for the user authentication. The user session is executed under the context of this mapped windows account by IIS. For this to work we need to ensure that the certificate to account mapping is configured correctly in IIS.

In IIS 6.0, the user had the option to configure Many-to-One client certificate mapping through the IIS Manager User Interface. In IIS 7/7.5, we don't have such an interface for either One-to-One or Many-to-One mappings. This post talks about the Configuration Editor IIS 7/7.5 extension that can be used to achieve the mappings either for One-to-One or Many-to-One. Here we will talk in specific about Many-to-1 mapping.

IIS 7 or IIS 7.5 Schema

This is the schema for the IIS Client Certificate Mapping authentication feature in IIS 7 or IIS 7.5.

```
<sectionSchema name="system.webServer/security/authentication/iisClientCertificateMappingAuthentication">
    <attribute name="enabled" type="bool" defaultValue="false" />
    <attribute name="manyToOneCertificateMappingsEnabled" type="bool" defaultValue="true" />
    ...
    <element name="manyToOneMappings">
      <collection addElement="add" clearElement="clear">
        <attribute name="name" type="string" required="true" isUniqueKey="true" validationType="nonEmptyString" />
        <attribute name="description" type="string" />
        <attribute name="enabled" type="bool" defaultValue="true" />
        <attribute name="permissionMode" type="enum" defaultValue="Allow">
          <enum name="Allow" value="1" />
          <enum name="Deny" value="2" />
        </attribute>
        <element name="rules">
          <collection addElement="add" clearElement="clear">
            <attribute name="certificateField" type="enum" required="true" isCombinedKey="true">
              <enum name="Subject" value="1" />
              <enum name="Issuer" value="2" />
            </attribute>
            <attribute name="certificateSubField" type="string" caseSensitive="true" required="true" isCombinedKey="true" />
            <attribute name="matchCriteria" type="string" caseSensitive="true" required="true" isCombinedKey="true" />
            <attribute name="compareCaseSensitive" type="bool" isCombinedKey="true" defaultValue="true" />
          </collection>
        </element>
        <attribute name="userName" type="string" validationType="nonEmptyString" />
        <attribute name="password" type="string" caseSensitive="true" encrypted="true" defaultValue="[enc:AesProvider::enc]" />
```

```
      </collection>
    </element>
    ...
  </sectionSchema>
```

Prerequisites

These are the prerequisites needed for this walkthrough.

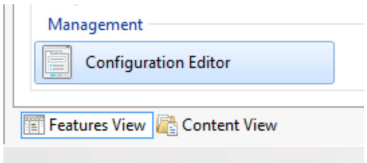
1. We have installed IIS Client Certificate Mapping module on the server.
2. A Web Site is configured with an HTTPS binding which can accept SSL connections.
3. We have a client certificate installed on the client.
4. IIS 7 Administration Pack is installed on the IIS 7.0 server. NOTE: Configuration Editor is shipped by default on IIS 7.5.

Walkthrough

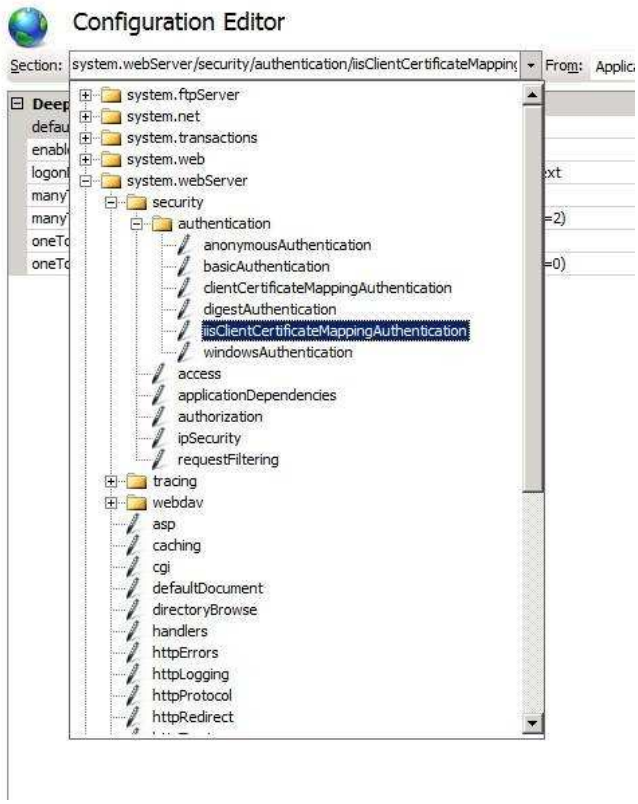
Step 1:

1. Launch the IIS manager and select your web site which is being configured for client certificate authentication.

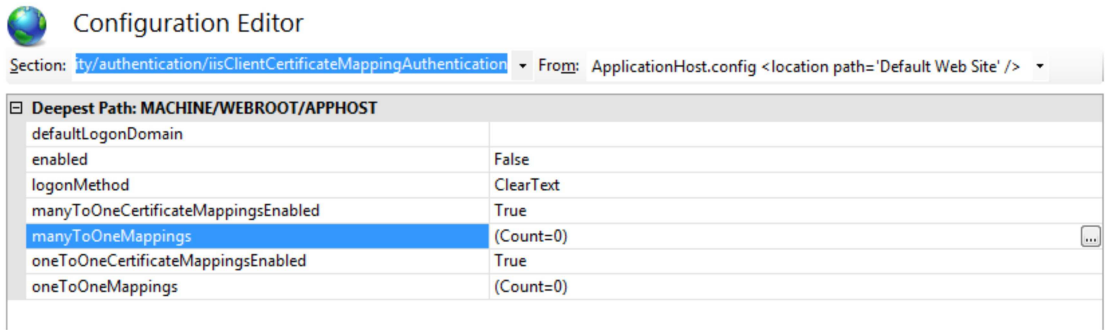
2. In the features View select Configuration Editor under Management section in the Features View.



3. Go to "**system.webServer/security/authentication/iisClientCertificateMappingAuthentication**" in the drop down box as shown below:



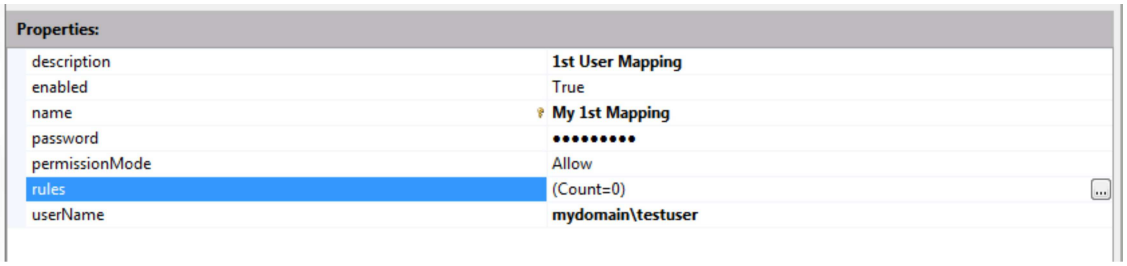
You will see a window to configure Many-to-One or One-to-One certificate mappings here. This is the UI provided through Configuration editor from where we can setup all the mapping configurations.



4. We can go ahead and modify the properties through this GUI.

- Set **enabled** to true
- Set **manyToOneCertificateMappingsEnabled** to True
- Select **manyToOneMappings** and click on the extreme end at the Ellipsis button to launch the new window for configuring mappings.

5. Under this new window go ahead and Add a new item. You can modify the properties from within the window as shown below:



6. Click on the Ellipsis button for **rules** and this will give you an option to add multiple patterns for matching based on certificate properties.

Properties:

certificateField	**Subject**
certificateSubField	**CN**
compareCaseSensitive	True
matchCriteria	**Test User**

Properties:

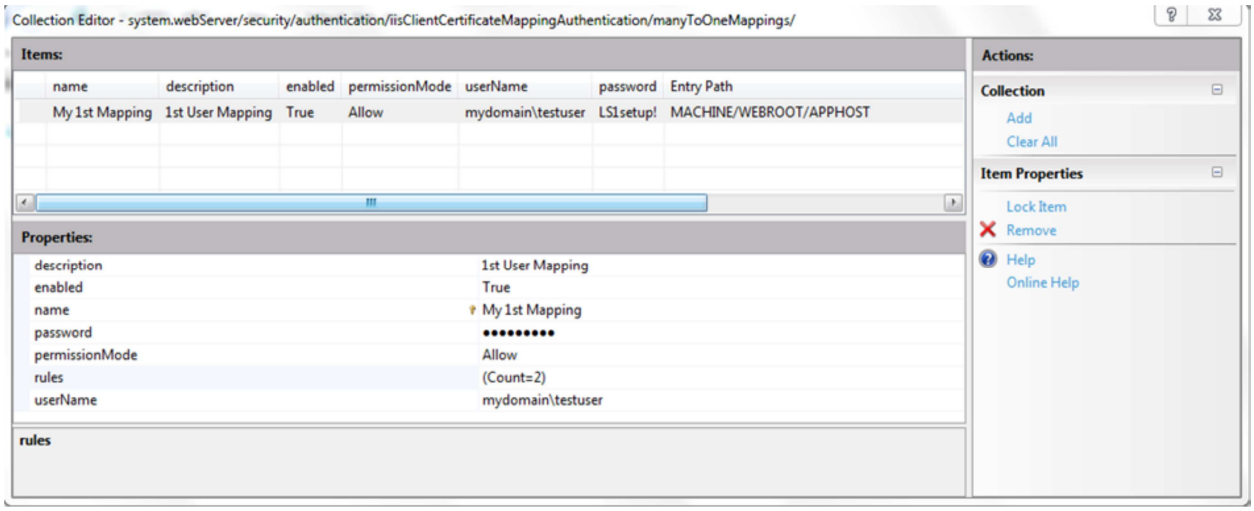
certificateField	**Issuer**
certificateSubField	**CN**
compareCaseSensitive	True
matchCriteria	**My IT Enterprise**

Items:

certificateField	certificateSubField	matchCriteria	compareCaseSensitive	Entry Path
Subject	CN	Test User	True	MACHINE/WEBROOT/APPHOST
Issuer	CN	My IT Enterprise	True	MACHINE/WEBROOT/APPHOST

So here above we have two entries for rules for mapping the certificate. In the above case we are using two different fields named Subject and the Issuer in the certificate field and based on the **matchcriteria** property map the certificate to the account mydomain\testuser.

Shown below is how the final mapping for a specific windows account looks like. As you can see there are two entries for rules for this account.



Similarly we can have other mappings for various accounts based on the fields "Issuer" and "Subject" in the Certificate.

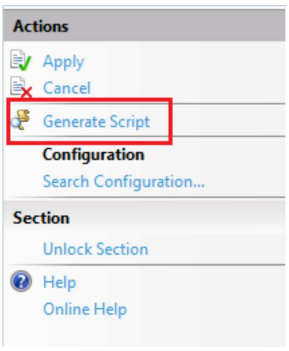
We can also use Configuration Editor to configure One-to-One mapping. One can follow the instructions in the article listed below to configure One-to-One mappings.

http://learn.iis.net/page.aspx/478/configuring-one-to-one-client-certificate-mappings/

Appendix

So far what we have seen is achieved using the Configuration Editor which gives you a graphical interface to easily set the configuration.

You can achieve the same thing using APPCMD command, in fact the Configuration Editor does the same thing in the background and adds these settings in the ApplicationHost.config file. Configuration Editor gives you an option to run these commands manually, it generates the scripts to achieve that from its UI itself.



These are the Code snippets to perform the same steps as above to configure mapping. They were was generated using Configuration Editor's Script Generation feature.

AppCmd instructions

```
appcmd.exe set config "Default Web Site" -section:system.webServer/security/authentication/iisClientCertificateMappingAuthentication /en
appcmd.exe set config "Default Web Site" -section:system.webServer/security/authentication/iisClientCertificateMappingAuthentication /+"
appcmd.exe set config "Default Web Site" -section:system.webServer/security/authentication/iisClientCertificateMappingAuthentication /+"
```

C# Code

```
using System;
using System.Text;
using Microsoft.Web.Administration;
internal static class Sample {
    private static void Main() {

        using(ServerManager serverManager = new ServerManager()) {
            Configuration config = serverManager.GetApplicationHostConfiguration();

            ConfigurationSection iisClientCertificateMappingAuthenticationSection = config.GetSection("system.webServer/security/authenticat
            iisClientCertificateMappingAuthenticationSection["enabled"] = true;
            iisClientCertificateMappingAuthenticationSection["manyToOneCertificateMappingsEnabled"] = true;

            ConfigurationElementCollection manyToOneMappingsCollection = iisClientCertificateMappingAuthenticationSection.GetCollection("many

            ConfigurationElement addElement = manyToOneMappingsCollection.CreateElement("add");
            addElement["name"] = @"My 1st  Mapping";
            addElement["description"] = @"1st User Mapping";
            addElement["userName"] = @"mydomain\testuser";
            addElement["password"] = @"abcdef";

            ConfigurationElementCollection rulesCollection = addElement.GetCollection("rules");

            ConfigurationElement addElement1 = rulesCollection.CreateElement("add");
            addElement1["certificateField"] = @"Subject";
            addElement1["certificateSubField"] = @"CN";
            addElement1["matchCriteria"] = @"Test User";
            rulesCollection.Add(addElement1);
            manyToOneMappingsCollection.Add(addElement);

            serverManager.CommitChanges();
        }
    }
```

```
}
```

Scripting (JavaScript)

```
var adminManager = new ActiveXObject('Microsoft.ApplicationHost.WritableAdminManager');
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";
var iisClientCertificateMappingAuthenticationSection = adminManager.GetAdminSection("system.webServer/security/authentication/iisClientCertifi
iisClientCertificateMappingAuthenticationSection.Properties.Item("enabled").Value = true;
iisClientCertificateMappingAuthenticationSection.Properties.Item("manyToOneCertificateMappingsEnabled").Value = true;
var manyToOneMappingsCollection = iisClientCertificateMappingAuthenticationSection.ChildElements.Item("manyToOneMappings").Collection;
var addElement = manyToOneMappingsCollection.CreateNewElement("add");
addElement.Properties.Item("name").Value = "My 1st  Mapping";
addElement.Properties.Item("description").Value = "1st User Mapping";
addElement.Properties.Item("userName").Value = "mydomain\\testuser";
addElement.Properties.Item("password").Value = "abcdef";
var rulesCollection = addElement.ChildElements.Item("rules").Collection;
var addElement1 = rulesCollection.CreateNewElement("add");
addElement1.Properties.Item("certificateField").Value = "Subject";
addElement1.Properties.Item("certificateSubField").Value = "CN";
addElement1.Properties.Item("matchCriteria").Value = "Test User";
rulesCollection.AddElement(addElement1);
manyToOneMappingsCollection.AddElement(addElement);
adminManager.CommitChanges();
```

Read the complete post here

Submit a Comment

You are logged in as
Anonymous

Plain text is accepted.
URLs starting with **http://** are converted to links.

Submit Comment