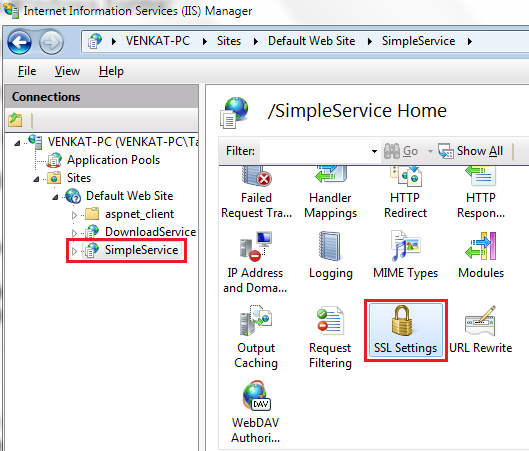
**Creating self-signed certificates**  
There are several ways to create self signed test certificates. Let us explore 2 of the easier options available. The easiest and simpler approach is to use IIS to create these certificates. In IIS 7.5  
**1.** Click on the "Server Name"  
**2.** Double click "Server Certificates" feature  
**3.** Click on "Create Self Signed Certificate" link, under "Actions"  
**4.** Specify a friendly name for the certificate and click OK. The friendly name is not part of the certificate itself, but is used by the server administrator to easily distinguish the certificate.

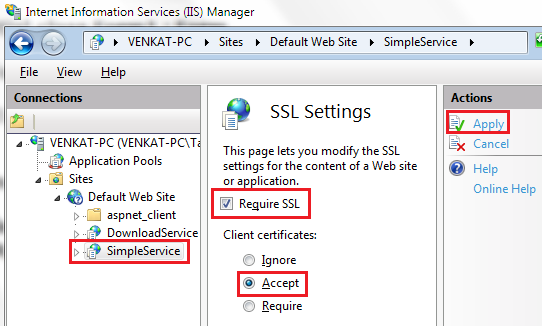
Cool to use computer name in the friendly Cert name.

cmd prompt-Enter hostname to get current PC name.

**Associating an asp.net web application with a specific certificate**  
Add HTTPS site binding, if it is not already present  
1. Open IIS  
2. Expand the "Server Name"  
3. Expand "Sites"  
4. Select "Default Web Site"  
5. Click "Binding" under "Edit Site" in "Actions" pane.  
6. In the "Site Bindings" window, Click "Add"  
7. Select Type = "https" and the SSL Certificate and click "OK"  
8. Click "Close" on "Site Bindings" window

At this point, you will be able to access your application using both HTTP and HTTPS protocol. When the site is accessed over HTTPS, you may receive a browser warning about the authenticity of the website. In a later video session we will discuss about resolving this.

 In IIS, configure **SSL settings**for the **Service**.   
**a)** To do this, select the **"Service"**and double click on **"SSL Settings"**   
   
b) Select **"Require SSL"**check box and **"Accept"**radio button under **"Client certificates"**and then click on **"Apply"**link button.



 In **web.config,**include the following bindings section, to customize the **wsHttpBinding**to use **Transport**security and **Basic**client credential type.

<bindings>

  <wsHttpBinding>

    <binding name="wsHttp">

      <security mode="Transport">

        <transport clientCredentialType="Basic"></transport>

      </security>

    </binding>

  </wsHttpBinding>

</bindings>

Associate the above customization with the **service endpoint**using**bindingConfiguration**attribute as shown below.

<endpoint address="SimpleService"

          binding="wsHttpBinding"

          contract="SimpleService.ISimpleService"

          bindingConfiguration="wsHttp"/>

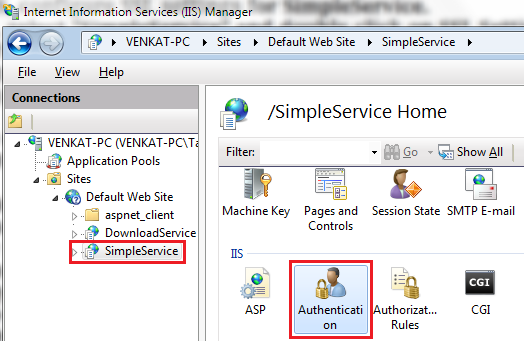
At this point, trying to navigate tohttps://localhost/SimpleService/SimpleService.svc will throw the following error.  
Security settings for this service require 'Basic' Authentication but it is not enabled for the IIS application that hosts this service.  
  
To fix the above error, **enable 'Basic' Authentication** in IIS. To do this  
**a)** In IIS, select **"Simple Service"**under **"Default Web Site"**and then double click on**"Authentication"**icon.

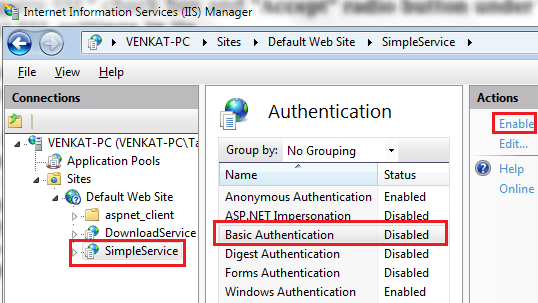
**NOTE: If you don't have Basic Authentication available do the following...**

Go to Programs and Features, Turn On or Off Windows Components, and enable Basic Authentication under IIS.

The complete path for Windows 7 and 8 is:

* Control panel
* Programs and Features
* Turn Windows Features on or off
* Expand: Internet Information Services => World Wide Web Services => Security
* Select "Basic Authentication"



Select **"Basic Authentication"**and then click on **"Enable"**link   


Trying to navigate to https://localhost/SimpleService/SimpleService.svc will throw the following error.  
The HttpGetEnabled property of ServiceMetadataBehavior is set to true and the HttpGetUrl property is a relative address, but there is no http base address.  Either supply an http base address or set HttpGetUrl to an absolute address.  
  
To fix the above error, change <serviceMetadata httpGetEnabled="true"/> to <serviceMetadata httpsGetEnabled="true" />. Save the changes and then navigate tohttps://localhost/SimpleService/SimpleService.svc. At this point we should be able to see the **SimpleService** WSDL.

NOTE: As of this document CROME has a bug that will not show https with self signed cert--------Use IE

In the client application, delete the **SimpleService**service reference, and then add the service reference again to **SimpleService.**This time usehttps://localhost/SimpleService/SimpleService.svc instead ofhttp://localhost/SimpleService/SimpleService.svc. Notice that we are using **Https**instead of **Http.**At this point, notice the configuration in the **app.config**file of the **client application.**Notice that instead of the **defaults of wsHttpBinding,**we are now using**"Transport"**security mode and **"Basic"**client credential type.  
<security mode="Transport">  
<transport clientCredentialType="Basic"/>  
  
Run the client application, and then click **"Call Service"**button. The following error will be thrown.  
The username is not provided. Specify username in ClientCredentials.  
  
To fix the above error, modify the code in **btnCallService\_Click**() event handler method in **Form1.cs**of the **client application**as shown below.

privatevoidbtnCallService\_Click**(**objectsender**,**EventArgse**)**

**{**

SimpleService.SimpleServiceClientclient=new

SimpleService.SimpleServiceClient**();**

client.ClientCredentials.UserName.UserName="WindowsUserName"**;**

client.ClientCredentials.UserName.Password="Password"**;**

MessageBox.Show**(**client.GetMessage**(**"Pragim"**));**

**}**

Run the client application, and then click **"Call Service"**button. We now get the out put as expected.  Inspect the logged messages and notice that the body of the SOAP message is not encrypted, which implies that the **wsHttpBinding**is now using **Transport** security.