Working with SSL and Client Certificate in Web API

Several common authentication schemes are not secure over plain HTTP. In particular, Basic authentication and forms authentication send unencrypted credentials. To be secure, these authentication schemes *must* use SSL. In addition, SSL client certificates can be used to authenticate clients.

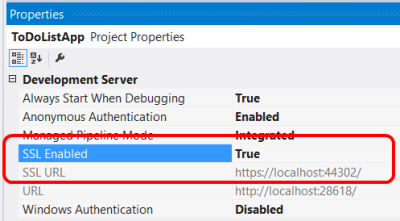
**Enabling SSL on the Server**

To set up SSL in IIS 7 or later:

* Create or get a certificate. For testing, you can create a self-signed certificate.
* Add an HTTPS binding.

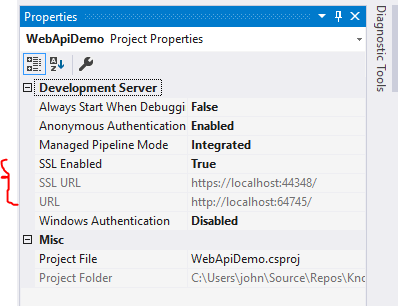
For details, see [How to Set Up SSL on IIS 7](https://www.iis.net/learn/manage/configuring-security/how-to-set-up-ssl-on-iis).

For local testing, you can enable SSL in IIS Express from Visual Studio. In the Properties window, set **SSL Enabled** to **True**. Note the value of **SSL URL**; use this URL for testing HTTPS connections.

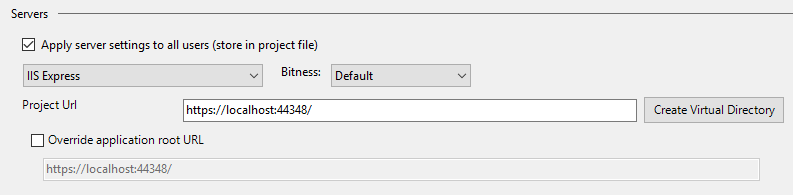


If you develop web project in Visual Studio 2019, it will take care of setting up Server Certificate and SSL in one step for you automatically.

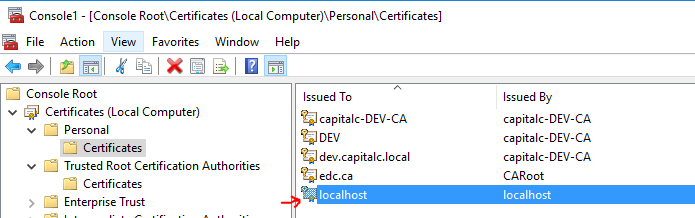
To set up SSL in IISExpress, right click web project 🡪 Properties:



🡪 Web tab 🡪 Servers section:



If you open mmc.exe (Microsoft Management Console) you will see a certificate “localhost” being installed as Server Certificate:



**Enforcing SSL in a Web API Controller**

If you have both an HTTPS and an HTTP binding, clients can still use HTTP to access the site. You might allow some resources to be available through HTTP, while other resources require SSL. In that case, use an action filter to require SSL for the protected resources. The following code shows a Web API authentication filter that checks for SSL:

C#

public class RequireHttpsAttribute : AuthorizationFilterAttribute

{

public override void OnAuthorization(HttpActionContext actionContext)

{

if (actionContext.Request.RequestUri.Scheme != Uri.UriSchemeHttps)

{

actionContext.Response = new HttpResponseMessage(System.Net.HttpStatusCode.Forbidden)

{

ReasonPhrase = "HTTPS Required"

};

}

else

{

base.OnAuthorization(actionContext);

}

}

}

Add this filter to any Web API actions that require SSL:

C#C

public class ValuesController : ApiController

{

[RequireHttps]

public HttpResponseMessage Get() { ... }

}

**SSL Client Certificates**

SSL provides authentication by using Public Key Infrastructure certificates. The server must provide a certificate that authenticates the server to the client. It is less common for the client to provide a certificate to the server, but this is one option for authenticating clients. To use client certificates with SSL, you need a way to distribute signed certificates to your users. For many application types, this will not be a good user experience, but in some environments (for example, enterprise) it may be feasible.

| **SSL CLIENT CERTIFICATES** | |
| --- | --- |
| **Advantages** | **Disadvantages** |
| - Certificate credentials are stronger than username/password. - SSL provides a complete secure channel, with authentication, message integrity, and message encryption. | - You must obtain and manage PKI certificates. - The client platform must support SSL client certificates. |

To configure IIS to accept client certificates, open IIS Manager and perform the following steps:

1. Click the site node in the tree view.
2. Double-click the **SSL Settings** feature in the middle pane.
3. Under **Client Certificates**, select one of these options:
   * **Accept**: IIS will accept a certificate from the client, but does not require one.
   * **Require**: Require a client certificate. (To enable this option, you must also select "Require SSL")

You can also set these options in the ApplicationHost.config file:

The whole configuration is inside the solution folder in the .vs\config\applicationhost.config configuration file which you best open in a text editor outside of Visual Studio.

Search for an xml element called <access> inside a <security> element:

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | <security>  <access sslFlags="None" /> | |

The default configuration has no support for SSL client certificates. You need to modify the sslFlags attribute to include these options: Ssl, SslNegotiateCert, SslRequireCert

|  |
| --- |
| <security>  <access sslFlags="Ssl, SslNegotiateCert, SslRequireCert" /> |

The next step is to find the element <iisClientCertificateMappingAuthentication>:

|  |
| --- |
| <sectionGroup name="system.webServer">  <sectionGroup name="security">  <section name="access" overrideModeDefault="Deny" />  <section name="applicationDependencies" overrideModeDefault="Deny" />  <sectionGroup name="authentication">  <section name="anonymousAuthentication" overrideModeDefault="Allow" />  <section name="basicAuthentication" overrideModeDefault="Allow" />  <section name="clientCertificateMappingAuthentication" overrideModeDefault="**Allow**" />  <section name="digestAuthentication" overrideModeDefault="Allow" />  <section name="iisClientCertificateMappingAuthentication" overrideModeDefault="**Allow**" />  <section name="windowsAuthentication" overrideModeDefault="Allow" />  </sectionGroup> |

You now need to restart Visual Studio and IIS Express. IIS Express can be restarted using the icon in your system tray.

The **SslNegotiateCert** flag means IIS will accept a certificate from the client, but does not require one (equivalent to the "Accept" option in IIS Manager). To require a certificate, set the **SslRequireCert** flag.

**Creating a Client Certificate for Testing**

For testing purposes, you can use [MakeCert.exe](https://docs.microsoft.com/en-us/windows/desktop/SecCrypto/makecert) to create a client certificate. First, create a test root authority:

Console

makecert.exe -n "CN=Development CA" -r -sv TempCA.pvk TempCA.cer

Makecert will prompt you to enter a password for the private key.

Next, add the certificate to the test server's "Trusted Root Certification Authorities" store, as follows:

1. Open MMC.
2. Under **File**, select **Add/Remove Snap-In**.
3. Select **Computer Account**.
4. Select **Local computer** and complete the wizard.
5. Under the navigation pane, expand the "Trusted Root Certification Authorities" node.
6. On the **Action** menu, point to **All Tasks**, and then click **Import** to start the Certificate Import Wizard.
7. Browse to the certificate file, TempCA.cer.
8. Click **Open**, then click **Next** and complete the wizard. (You will be prompted to re-enter the password.)

Now create a client certificate that is signed by the first certificate:

Console

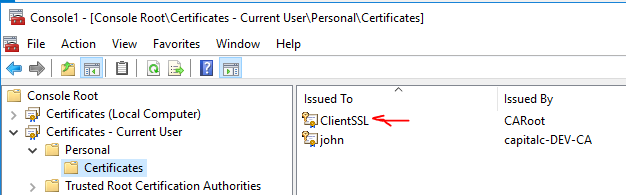
makecert.exe -pe -ss My -sr CurrentUser -a sha1 -sky exchange -n "CN=name"

-eku 1.3.6.1.5.5.7.3.2 -sk SignedByCA -ic TempCA.cer -iv TempCA.pvk

**\*.cer**: only has the public key (this is what you typically exchange with integration partners); it can be used to verify tokens or client authentication requests, and it is what is received by an HTTP client from a server in the SSL handshake.

**\*.pfx**: includes both the public and private key for the associated certificate (NEVER share this outside your organization); it can be used for TLS/SSL on web site, for digitally signing messages or authorization tokens, or for authenticating to a partner system.

* Trusted Root Certificate install \*.cer file
* Server Certificate and Client Certificate are install using \*.pfx files so that each has its own private key installed.
* Install Server Certificate in IIS also use \*.pfx file



**Using Client Certificates in Web API**

On the server side, you can get the client certificate by calling [GetClientCertificate](https://msdn.microsoft.com/library/system.net.http.httprequestmessageextensions.getclientcertificate.aspx) on the request message. The method returns null if there is no client certificate. Otherwise, it returns an **X509Certificate2** instance. Use this object to get information from the certificate, such as the issuer and subject. Then you can use this information for authentication and/or authorization.

C#

X509Certificate2 cert = Request.GetClientCertificate();

string issuer = cert.Issuer;

string subject = cert.Subject;