Integrating DoubleClue Authentication into an ASP .Net Web Application with SAML2

Content

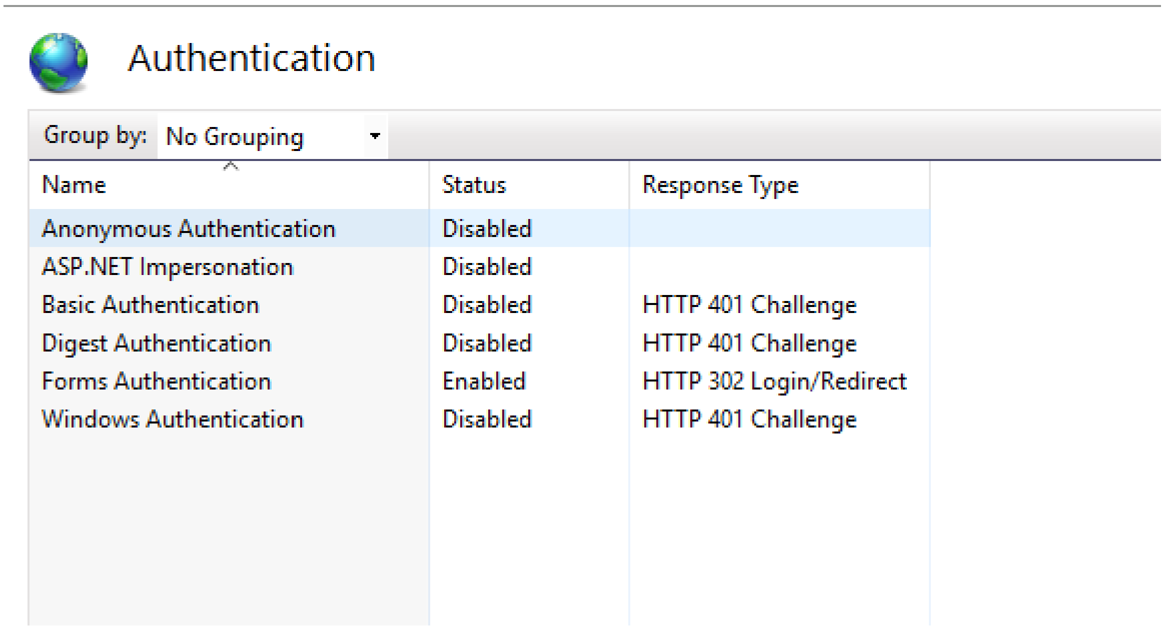
1. Simple Integration for Double Clue SAML in Asp .Net Web Application 2
1.1. Enabling Forms Authentication in IIS 2
1.2 Add the SustainSys Package Files to the Project 2
1.3 Preparing DCEM to be an Identity Provider 3
1.4 Modify the Application’s Configuration Files 4
2. Integrating Double Clue SAML in Asp .Net Core 3.1 / 5.0 Web Application 6
2.1 Prerequisites........ 6
2.2 Required NuGet Packages 7
2.3 Implementation........ 7
2.4 Configure Idp 9
3. Appendix........ 9
3.1 Further Information........ 9
3.2 Full Code Sample for Web.Config........ 10

1. Simple Integration of DoubleClue in Asp .Net Web Applications with SAML

This guide is intended for administrators who want to secure their users' access to websites and applications hosted with Windows Internet Information Services (IIS) with DoubleClue Multi-Factor-Authentication (MFA). This can be achieved by using SustainSys module to add the SAML2 standard to the ASP.net Framework of IIS and set up DoubleClue as a SAML Identity Provider (IdP). SustainSys is a free open source software that is published under the MIT license.

1.1. Enabling Forms Authentication in IIS

- Go to the IIS Manager and here to the side section
- Search for the application that you want to secure with DoubleClue MFA and select it
- Select the authentication tool
- Enable Forms Authentication



Name	Status	Response Type
Anonymous Authentication	Disabled	
ASP.NET Impersonation	Disabled	
Basic Authentication	Disabled	HTTP 401 Challenge
Digest Authentication	Disabled	HTTP 401 Challenge
Forms Authentication	Enabled	HTTP 302 Login/Redirect
Windows Authentication	Disabled	HTTP 401 Challenge

1.2 Add the SustainSys Package Files to the Project

To connect your Windows IIS via SAML, you need to add the files of the SustainSys Modules to the project. Different components of IIS need different packages, which can be downloaded at https://www.nuget.org/packages. We provide more information and direct links in chapter 1.4 Configuring the Application's web.config.

Alternatively, you can contact us at support@doubleclue.com and we will send you the requested files.

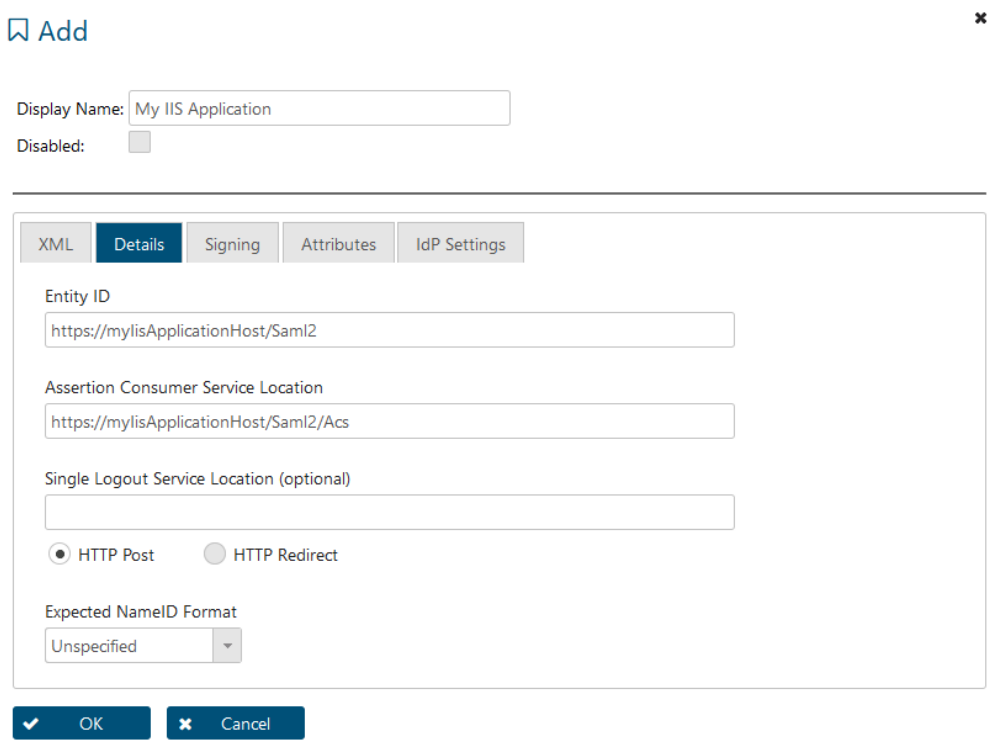
Once you have downloaded the files, add them to the project directory. You can normally find this in the “bin” folder of the web application or website you want to connect with your DoubleClue Enterprise Management (DCEM). It is the same folder in which the ***web.config*** is located. If the “bin” folder does not exist in the directory, you need to create a folder and call it “bin.

For the authentication to work properly, it is important that the whole directory of the web application or website are accessible for users when they log in. Therefore, it is advised to allow full access to the folder for authenticated users. This can be done in the preferences under the security tab of the parent folder.

1.3 Preparing DCEM to be an Identity Provider

You can find general information on how to prepare DCEM to be an Identity Provider in chapter 12 of the “DCEM_Manual_EN.pdf”.

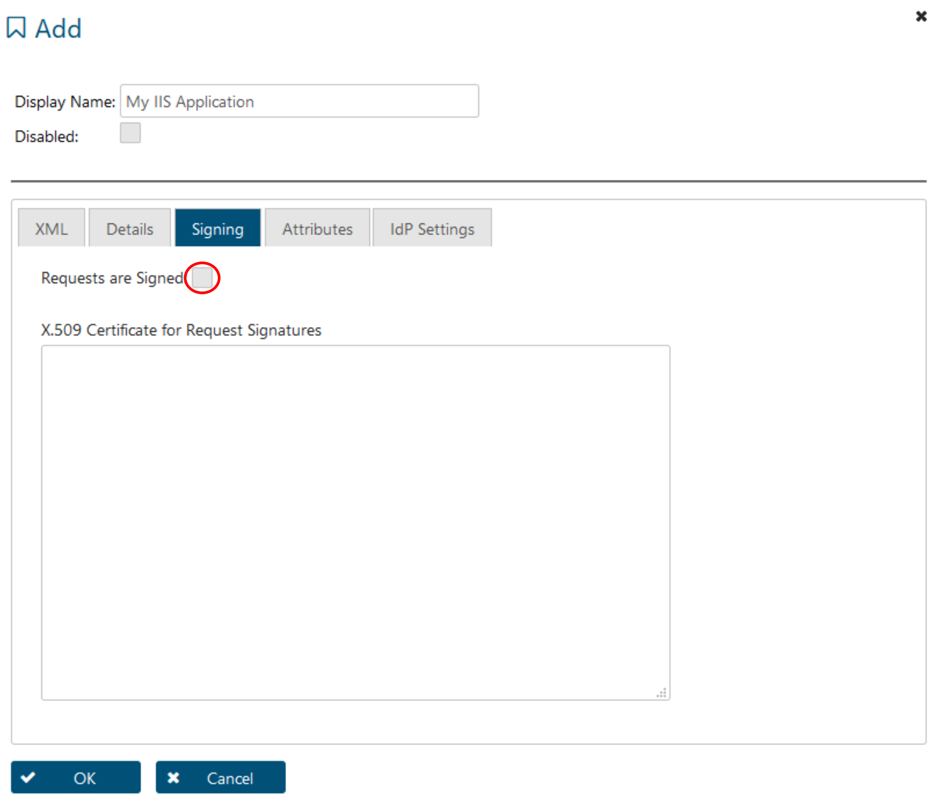
1. Add a new Service Provider to DCEM and choose the custom template.
2. Choose a globally unique Entity ID
3. Setup the “Assertion Consumer Service Location” myIisHost/Saml2/Acs. Ideally, the URL uses an SSL encryption, as Personally Identifiable Information (PII) will be exchanged with the service provider.



Display Name: My IIS Application
Disabled: ☐

Entity ID: https://myIisApplicationHost/Saml2
Assertion Consumer Service Location: https://myIisApplicationHost/Saml2/Acs
Single Logout Service Location (optional):
● HTTP Post ○ HTTP Redirect
Expected NameID Format: Unspecified

4. It is not necessary to sign SAML requests in this scenario. We therefore advice to go to the Signing and uncheck the “Requests are Signed” checkbox here. If you want to sign requests, see DCEM_Manual_EN.pdf chapter 12.



Display Name: My IIS Application

Disabled:

Requests are Signed

X.509 Certificate for Request Signatures

1.4 Modify the Application's Configuration Files

For the installed modules to be functional, you need to configure the respective configurations files of the IIS hosted applications.

The Nuget package for sustain sys, the Sustainsys.Saml2.HttpModule, is available at https://www.nuget.org/packages/Sustainsys.Saml2.HttpModule

In this chapter, you will find the code parts that need to be modified, depending on the used IIS components.

Implement the following changes into the **web.config** in your web application. The file is automatically added when creating project on visual studio:

1. Define sections and integrate the dll-files added in step 3.

Code Sample:

```
<section name="system.identityModel"
type="System.IdentityModel.Configuration.SystemIdentityModelSection,
System.IdentityModel, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089" />
    <section name="system.identityModel.services"
type="System.IdentityModel.Services.Configuration.SystemIdentityModelServicesSect
ion, System.IdentityModel.Services, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089" />
    <section name="sustainsys.saml2"
type="Sustainsys.Saml2.Configuration.SustainsysSaml2Section, Sustainsys.Saml2" />
```

2. Add the forms authentication.

   Code Sample:

```
<system.web>
    <authentication mode="Forms">
      <forms loginUrl="~/Saml2/SignIn" />
    </authentication>
</system.web>
```

3. Add the session authentication module. The public keys are part of the downloaded dll-modules.

   Code Sample:

```
<system.webServer>
    <modules>
      <add name="SessionAuthenticationModule"
type="System.IdentityModel.Services.SessionAuthenticationModule,
System.IdentityModel.Services, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" preCondition="managedHandler" />
   </modules>
  </system.webServer>
```

4. Add the Saml2 configuration to the file. You have to modify some of the content in this section according to the settings in your DCEM. Enter the entityID , which you can find in your DCEM under SAML -> Settings. If your DCEM doesn’t have an Entity ID yet, you can give it one. You can freely choose the Entity ID as long as it is globally unique. It is recommended to use your URL as the Entity ID. Also provide a return URL to which the user will be redirected after the successful authentication and add a link to import the meta data of your DCEM.

Code Sample:

```
<sustainsys.saml2 entityId="YourDoubleClueSamlEntityId"
returnUrl="http://YourWebApplicationStartPage"
authenticateRequestSigningBehavior="Never">
    <identityProviders>
      <add entityId="YourDoubleClue.com" wantAuthnRequestsSigned="false"
loadMetadata="true"
metadataLocation="YourDoubleClue.com/dcem/saml/idp_metadata.xml"
allowUnsolicitedAuthnResponse="true" binding="HttpRedirect" />
    </identityProviders>
    <serviceCertificates>
      <!--<add fileName="~/App_Data/Sustainsys.Saml2.Tests.pfx"/>-->
    </serviceCertificates>
  </sustainsys.saml2>
```

5. It isn’t necessary to sign requests. We therefore advice to set wantAuthnRequestsSigned="false" (see Code Sample above) and to uncheck the box saying “Requests are Signed” under “Signing” in the Service Provider entry in DCEM.
   If you want to sign requests, see DCEM_Manual_EN.pdf chapter 12 and https://readthedocs.org/projects/saml2/downloads/pdf/latest/ chapter 3.16.

You can find a full code sample for the web.config of the Sustainsys.Saml2.HttpModule in chapter 3.2 Full Code Sample for Web.Config.

2. Integrating Double Clue SAML in Asp .Net Core 3.1 / 5.0 Web Application

In order to integrate SAML2 Authentication with DoubleClue as an IDP into your application, we recommend using the ITfoxtec Identity Saml2 package. A sample project is available for download to use as reference when including SAML in a web application at https://github.com/ITfoxtec/ITfoxtec.Identity.Saml2/tree/master/test/TestWebAppCore. Please check the license of the ITfoxtec library to ensure that it fits your needs of this project.

⚠ Do not use the ITfoxtec library and the Sustainsys library at the same time as this may cause an error. If your setup requires you to use both, ensure to define the correct library as a reference in the code.

2.1 Prerequisites

- .NET Core (tested with version 3.1)
- Editor/IDE such as Visual Studio Code or Visual Studio

2.2 Required NuGet Packages

- System.ServiceModel.Security
- ITfoxtec.Identity.Saml2.MvcCore
- BuildBundlerMinifier.core (used in sample not necessary)

The required packages can be found via the Nuget Package Manager.

2.3 Implementation

After implementing libraries, make the following changes to the *Startup.cs*. The *Startup.cs* is automatically created and by default located in the parent folder of your project.

See the example below of the *ConfigureServices* function. After making project with web.net core you get the startup.cs (automatically created) – in parent folder of the project.

```
public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<Saml2Configuration>(Configuration.GetSection("Saml2"));
            services.Configure<Saml2Configuration>(saml2Configuration =>
            {
                saml2Configuration.SigningCertificate =
CertificateUtil.Load(AppEnvironment.MapToPhysicalFilePath(Configuration["Saml2:SigningCertificateFile"]),
                    Configuration["Saml2:SigningCertificatePassword"], X509KeyStorageFlags.MachineKeySet);

                saml2Configuration.AllowedAudienceUris.Add(saml2Configuration.Issuer);

                var entityDescriptor = new EntityDescriptor();
                entityDescriptor.ReadIdPSsoDescriptorFromUrl(new Uri(Configuration["Saml2:IdPMetadata"]));
                if (entityDescriptor.IdPSsoDescriptor != null)
                {
                    saml2Configuration.AllowedIssuer = entityDescriptor.EntityId.ToString();
                    saml2Configuration.SingleSignOnDestination =
entityDescriptor.IdPSsoDescriptor.SingleSignOnServices.First().Location;
                    saml2Configuration.SingleLogoutDestination =
entityDescriptor.IdPSsoDescriptor.SingleLogoutServices.First().Location;

saml2Configuration.SignatureValidationCertificates.AddRange(entityDescriptor.IdPSsoDescriptor.SigningCertific
ates);
                    if (entityDescriptor.IdPSsoDescriptor.WantAuthnRequestsSigned.HasValue)
                    {
                        saml2Configuration.SignAuthnRequest =
entityDescriptor.IdPSsoDescriptor.WantAuthnRequestsSigned.Value;
                    }
                }
                else
                {
                    throw new Exception("IdPSsoDescriptor not loaded from metadata.");
                }
            });

            services.AddSaml2(slidingExpiration: true);
            services.AddControllersWithViews();
        }
```

In the *appsettings.json,* change the parameters used in the configuration services function. Some parameters such as *Issuer* and *SignatureAlgorithm* must be the same as the configured parameters from within the *SAML > Service Providers* section in the DoubleClue Management interface. An example of such parameters for connecting with DoubleClue is found below.

```
"Saml2": {
    "IdPMetadata": "https://doubleclue.examplegroup.de/dcem/saml/idp_metadata.xml",
    "Issuer": "SamlNetCoreApplication",
    "SignatureAlgorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
    "SigningCertificateFile": "C:/Apps/netcoreapp/doubleclue.p12",
    "SigningCertificatePassword": "test1",
    "CertificateValidationMode": "None",
    "RevocationMode": "NoCheck"
  }
```

Implement some further changes in the *Startup.cs*, make sure the line *app.UseSaml2();* is included just beneath the line *app.UseRouting().* See the example of the *Configure* function below.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
                app.UseHsts();
            }

            app.UseStaticFiles();
            app.UseRouting();
            app.UseSaml2();
            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
```

After the application has been set up to implement DoubleClue via SAML for authentication, you need to set up a controller to handle the authentication routing within the web application. To achieve this, create the *AuthController.cs* in your controllers' folder or copy it from the sample project into your project.

The class *AuthController.cs* contains the handlers for authentication, login, assertion, logout and other functionality. The assertion handler handles the response from DoubleClue and therefore the routing path has to be compatible with the one set up in the DoubleClue interface. In the example, the *ACS Location* is set up as *http://localhost:XX/Auth/AssertionConsumerService* since in the handlers we can find routing to this file path within the *AuthController.cs*

In the Assertion Consumer Service route, there is a class called *ClaimsTransform*. This helper class parses the claims out of the SAML Response from DoubleClue. The claims obtained from response contain attributes which could be default or even additional custom attributes set up from the DoubleClue Management Interface.

[AllowAnonymous] is a tag that can be used to functions/classes which dot require authentication to be accessed. [Authorize] is used to force authentication if user is not authorized when accessing the respective functionality.

When hosted on IIS the only authentication method enabled should *Anonymous Authentication*.

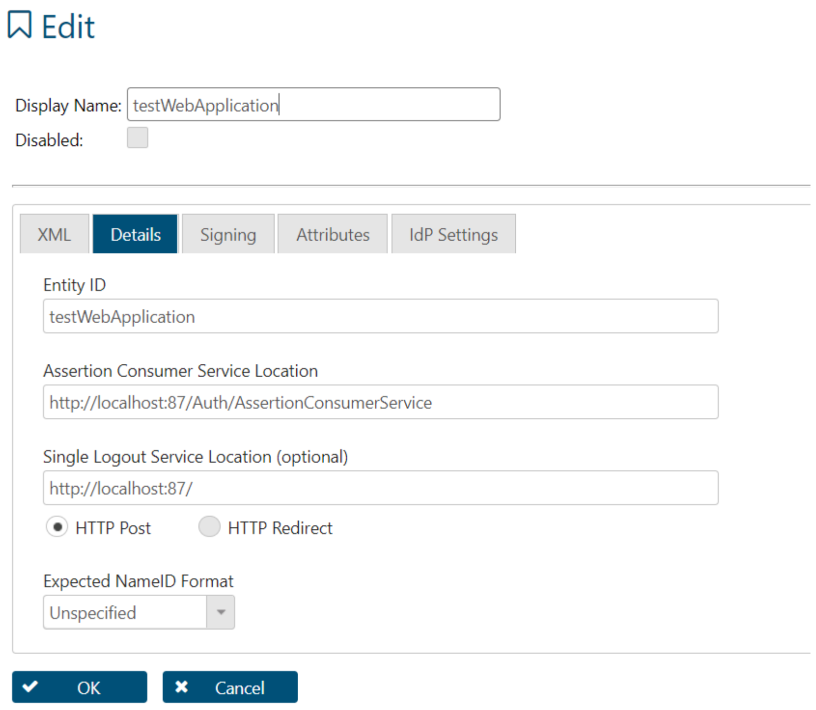
2.4 Configure Idp

To register a service provider in DoubleClue, access the DoublClue Enterprise Management (DCEM). You need to have administration rights to access the section *SAML > Service Providers* (as seen in section 1.3).

First add a new SP or use an existing one. The *Entity ID* in the details section needs to match with the *issuer* value in the SAML configuration on the web application. The ACS Location should be routing to the ACS handler in *AuthController.cs* and Single Logout Service Location is the location DoubleClue redirects to after performing the logout.

The signing section should be set to enabled and the certificate should be inserted in the text area provided.

Attributes (claims in the web application) are to be set as needed by the web application. Idp settings should match with the SAML configuration. Below you can see an example.



Display Name: testWebApplication
Disabled:

Entity ID
testWebApplication

Assertion Consumer Service Location
http://localhost:87/Auth/AssertionConsumerService

Single Logout Service Location (optional)
http://localhost:87/

[✓ HTTP Post] HTTP Redirect

Expected NameID Format
Unspecified

3. Appendix

3.1 Further Information

https://readthedocs.org/projects/saml2/downloads/pdf/latest/
https://resources.infosecinstitute.com/form-authentication-asp-net-security-part-3/#gref
https://github.com/ITfoxtec/ITfoxtec.Identity.Saml2/

3.2 Full Code Sample for Web.Config

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>

  <configSections>

    <!-- Add these sections below any existing. -->

    <section name="system.identityModel"
type="System.IdentityModel.Configuration.SystemIdentityModelSection,
System.IdentityModel, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089" />

    <section name="system.identityModel.services"
type="System.IdentityModel.Services.Configuration.SystemIdentityMode
lServicesSection, System.IdentityModel.Services, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089" />

    <section name="sustainsys.saml2"
type="Sustainsys.Saml2.Configuration.SustainsysSaml2Section,
Sustainsys.Saml2" />

  </configSections>

  <appSettings>

    <add key="webpages:Version" value="3.0.0.0" />

    <add key="webpages:Enabled" value="false" />

    <add key="ClientValidationEnabled" value="true" />

    <add key="UnobtrusiveJavaScriptEnabled" value="true" />

  </appSettings>

  <system.web>

    <authentication mode="Forms">

      <forms loginUrl="~/Saml2/SignIn" />

    </authentication>

    <compilation targetFramework="4.7.2" />

    <httpRuntime targetFramework="4.7.2" />

  </system.web>

  <system.webServer>

    <modules>

      <!-- Add the SessionAuthenticatioModule -->
      <add name="SessionAuthenticationModule"
type="System.IdentityModel.Services.SessionAuthenticationModule,
System.IdentityModel.Services, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />

       <add name="Saml2AuthenticationModule"
type="Sustainsys.Saml2.HttpModule.Saml2AuthenticationModule,
Sustainsys.Saml2.HttpModule" />

     </modules>

  </system.webServer>

  <sustainsys.saml2 entityId="yourDoubleClue"
returnUrl="https://yourIisApplication/welcome"
authenticateRequestSigningBehavior="Always"
outboundSigningAlgorithm="SHA512">

    <nameIdPolicy allowCreate="true" format="Unspecified" />

    <identityProviders>

      <add entityId="yourDoubleClue"
metadataLocation="https://yourDoubleClue.com/dcem/saml/idp_metadata.
xml" allowUnsolicitedAuthnResponse="true"
wantAuthnRequestsSigned="true" loadMetadata="false">

           <!--<signingCertificate storeName="TrustedPublisher"
storeLocation="LocalMachine"

                x509FindType="FindBySubjectName"
findValue="company.com"/>-->

      </add>

    </identityProviders>

     <serviceCertificates>

       <add use="Signing"
findValue="fce56b9cc41a0933d630ac228265425f90748217" storeName="My"
storeLocation="LocalMachine" x509FindType="FindByThumbprint" />

    </serviceCertificates>

  </sustainsys.saml2>

  <system.identityModel.services>

    <federationConfiguration>

      <cookieHandler requireSsl="false" />

    </federationConfiguration>

  </system.identityModel.services>

  <system.identityModel>
```

```
    <identityConfiguration>

      <securityTokenHandlers>

        <add
type="System.IdentityModel.Services.Tokens.MachineKeySessionSecurity
TokenHandler, System.IdentityModel.Services, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" />

        <remove
type="System.IdentityModel.Tokens.SessionSecurityTokenHandler,
System.IdentityModel, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />

      </securityTokenHandlers>

    </identityConfiguration>

  </system.identityModel>

</configuration>
```