OIDC sso agent for .NET Framework

Table of Contents

★ Introduction

Suppose you have an ASP.NET web application or else you are going to create a new one. One of your major concerns would be to provide a secure mechanism of handling user authentication and authorization. With the introduction of this OIDC SSO Agent, you will never have to worry about that at all. Moreover, you can just incorporate this agent to your ASP.NET web application and it will take care of all the things related to OIDC authentication mechanism.

★ How to incorporate to your asp.net web application?

If you plan to use OIDC SSO Agent, all you have to do is following simple steps below and then you have a web application that authenticates users with your favourite Identity Provider (Wso2 Identity Server).

Let’s get started. The process of incorporating SAML authentication with wso2 identity server via SAML agent can be explained in 6 steps.

1. Add the agent.dll reference to your Asp.NET web application(You can get this the git repo)

2. Configure - the mandatory properties in your ASP.NET web application’s web.config file. Following image shows how does it looks like after adding those properties to your web.config file.

Property	**Description**	**Default Value**
EnableOIDCSSOLogin	Enable OIDC authentication	false
OIDCSSOURL	SSO URL	oidcsso
OIDC.spName	Service Provider Identifier	*null*
OIDC.ClientId	Client key which was generated during OIDC configuration for Service Provider	*null*
OIDC.ClientSecret	Client Secret which was generated during OIDC configuration for Service Provider	*null*
OIDC.CallBackUrl	Callback URL	*null*
OIDC.GrantType	Grant Type	code
OIDC.AuthorizeEndpoint	Authorization Endpoint of the IDP which is used to get an authorization code.	https://localhost:9443/oauth2/authorize
OIDC.TokenEndpoint	Token endpoint of the IDP used to receive an access token	https://localhost:9443/oauth2/token
OIDC.UserInfoEndpoint	User info endpoint of the IDP which is used to fetch user details	https://localhost:9443/oauth2/userinfo?schema=openid
OIDC.Scope	Scope of the request as per the OIDC spec	openid
OIDC.EnableSLO	Enable single logout	
OIDC.SLOURL	Single logout URL	oidclogout
OIDC.EnableIDTokenValidation	Enable ID token validation	false

OIDC.PostLogoutRedirectUri	Post logout redirect URL	*null*
OIDC.SessionIFrameEndpoint	OP Session IFrame Endpoint	*null*

Below is a sample to demonstrate this step. You can edit the values as per needed:

```
<appSettings>
    <add key="EnableOIDCSSOLogin" value="true" />
    <add key="OIDCSSOURL" value="oidcsso" />
    <add key="OIDC.spName" value="music-store" />
    <add key="OIDC.ClientId" value="6G4s9GSYLd2USGB9f_Bf7kI6RHka" />
    <add key="OIDC.ClientSecret" value="_gWqRvvxrcxg_rZgraGX4d0fnS4a" />
    <add key="OIDC.CallBackUrl" value="http://localhost:58521/music-store/callback" />
    <add key="OIDC.GrantType" value="code" />
    <add key="OIDC.AuthorizeEndpoint" value="https://localhost:9443/oauth2/authorize" />
    <add key="OIDC.TokenEndpoint" value="https://localhost:9443/oauth2/token" />
    <add key="OIDC.UserInfoEndpoint"
            value="https://localhost:9443/oauth2/userinfo?schema=openid" />
    <add key="OIDC.Scope" value="openid" />
    <add key="OIDC.IdPEntityId" value="localhost" />
    <add key="OIDC.IdPURL" value="https://localhost:9443/" />
    <add key="OIDC.EnableSLO" value="true" />
    <add key="OIDC.SLOURL" value="oidclogout" />
    <add key="OIDC.EnableIDTokenValidation" value="true" />
    <add key="OIDC.PostLogoutRedirectUri"
            value="http://localhost:58521/music-store/Default" />
    <add key="OIDC.SessionIFrameEndpoint"
            value="https://localhost:9443/oidc/checksession" />
  </appSettings>
```

3. Next, if you want to validate ID token signature, you need to have a valid certificate. [*Note: It is highly recommended to use your own PKCS12 in your production environment*].
   For testing purposes you can get the **wso2carbon.jks** from the wso2 Identity server (**<IS_HOME> / repository/ resources/ security/ wso2carbon.jks**) and convert it to a **PKCS12** using keytool utility. Then, add the .p12 to the Local Machine certificate Store. However, below steps guide you through the process which was described above.
   - You get keytool by default with java installation and it could be found under the directory: C:\Program Files\Java\jre<Version>\bin , with the name keytool.exe .

- You can use the below command to convert the wso2carbon.jks to wso2carbon.p12
  keytool -importkeystore -srckeystore wso2carbon.jks -destkeystore wso2carbon.p12 -srcstoretype JKS -deststoretype PKCS12 -deststorepass [PASSWORD_PKCS12]
- Then, run microsoft management console(i.e: mmc.exe) as administrator, menu *File -> Add/Remove Snap-in..*, select "Certificates", press Add, select radio button "Computer account", and then you can install wso2carbon.p12

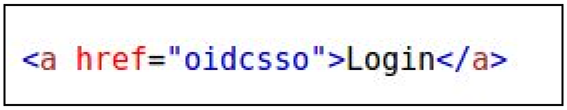
4. Register the “**FilteringHttpModule**” in your ASP.NET web application to handle the requests related to OIDC authentication mechanism.[*Note: The above mentioned **FilteringHttpModule** class is extended from IHttpModule. Click here for more information on IHttpModules.*]

5. Add the following code to the **global.asax** of your ASP.NET web application to enable session access from the agent.

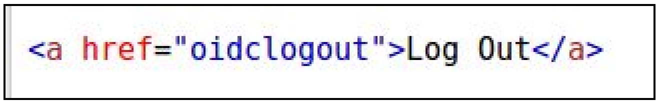
```
public override void Init()
{
   MapRequestHandler += EnableSession;
   base.Init();
}

void EnableSession(object sender, EventArgs e)
{
   HttpContext.Current.SetSessionStateBehavior(SessionStateBehavior.Required);
}
```

6. Set your application’s login controls to refer oidc intensive segments. That is suppose you have a login link in your web application. All you have to do is set the attribute href to “oidcsso”. And in the places that you have logout controls, it should be “oidclogout”.

   *[Note: “oidcsso” and “oidclogout” are values that were configured under Step No: 2 for the properties **OIDCSSOURL** and **OIDC.SLOURL** respectively. However, “oidcsso” and “oidclogout” are the default values for those two properties.]*



```
<a href="oidcsso">Login</a>
```



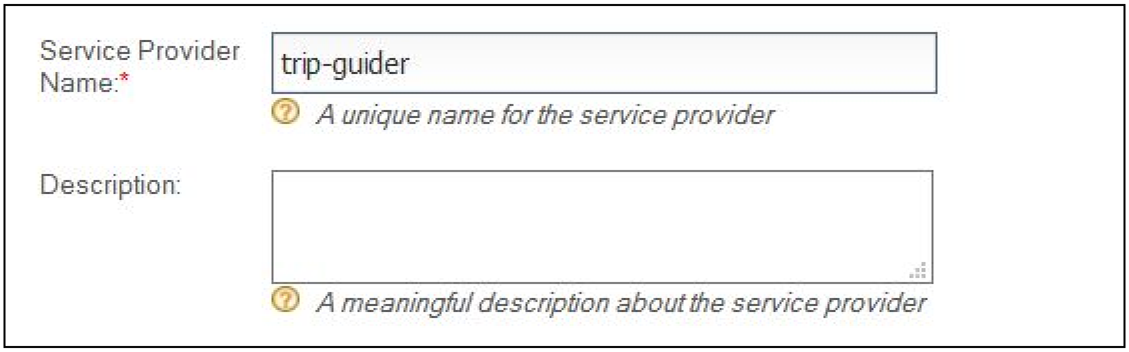
```
<a href="oidclogout">Log Out</a>
```

Upon successful completion of the 6 steps above, you ASP.NET web application is enabled with OIDC authentication.

★ Running samples

The given solution contains agent source code and two samples to demonstrate agent. These two samples can be used to demonstrate SAML Authentication as well as SSO and SLO. Sample 1 is analogous to a app that provides trip guidance. Sample 2 is a simple music store. First, you have to create two service providers in the IDP(wso2 Identity server) and configure OIDC authentication for both of them.

1. First create a Service provider in WSO2 Identity Server for Trip Guider app.



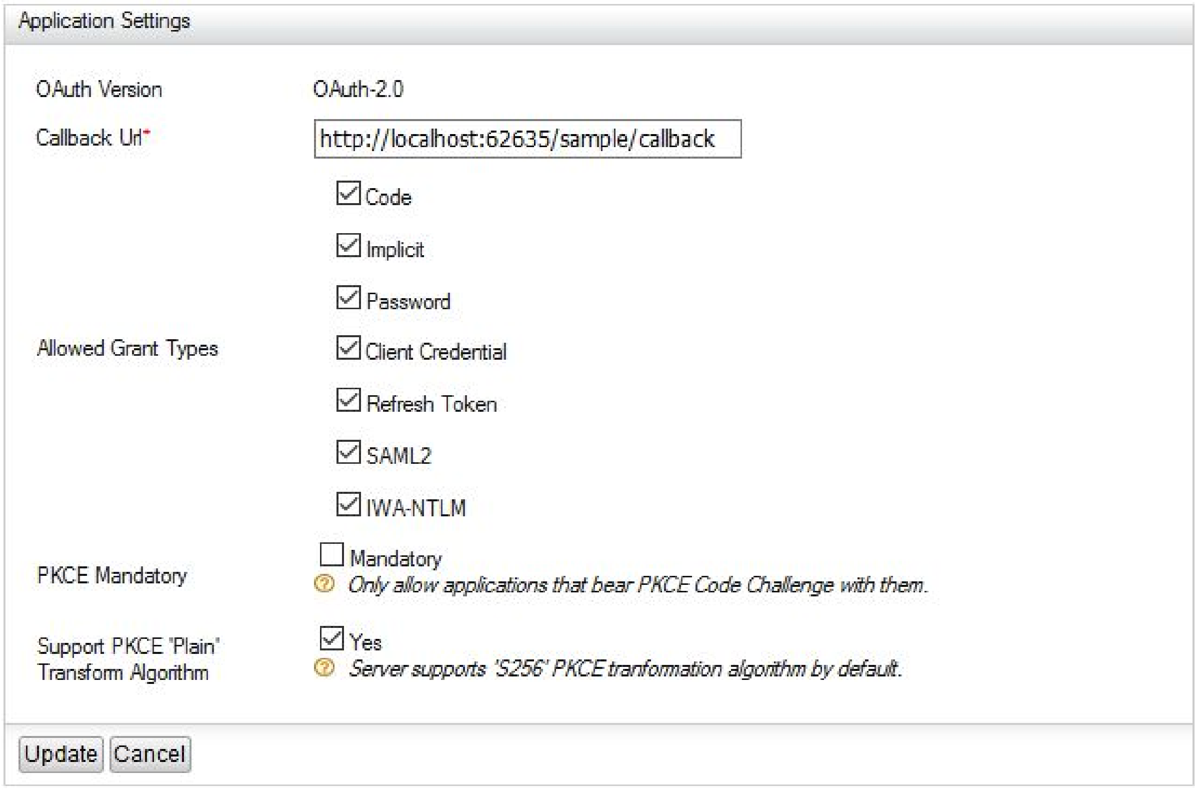
Service Provider Name:* trip-guider

A unique name for the service provider

Description:

A meaningful description about the service provider

2. Then navigate to Inbound Authentication Configuration -> OAuth / OpenID Connect Configuration.



Application Settings

OAuth Version: OAuth-2.0

Callback Url*: http://localhost:62635/sample/callback

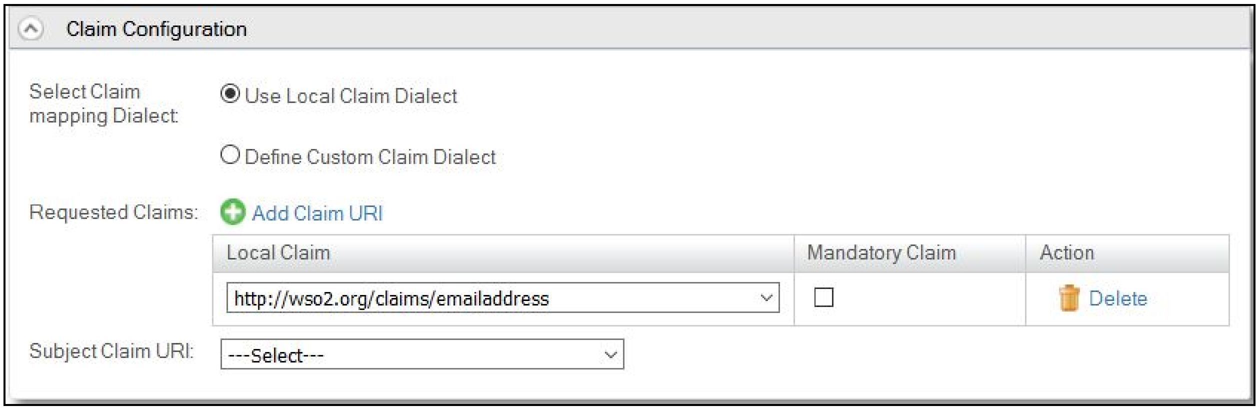
Allowed Grant Types:
☑ Code
☑ Implicit
☑ Password
☑ Client Credential
☑ Refresh Token
☑ SAML2
☑ IWA-NTLM

PKCE Mandatory: □ Mandatory
Only allow applications that bear PKCE Code Challenge with them.

Support PKCE 'Plain' Transform Algorithm: ☑ Yes
Server supports 'S256' PKCE tranformation algorithm by default.

3. You can do above configurations and click the update button.

4. Then navigate to Claim Configuration and add the claims you want under requested Claims. Here I have added email address. You can have any claim configuration you prefer.



Claim Configuration

Select Claim mapping Dialect: ● Use Local Claim Dialect
○ Define Custom Claim Dialect

Requested Claims: Add Claim URI

Local Claim	Mandatory Claim	Action
http://wso2.org/claims/emailaddress	□	Delete

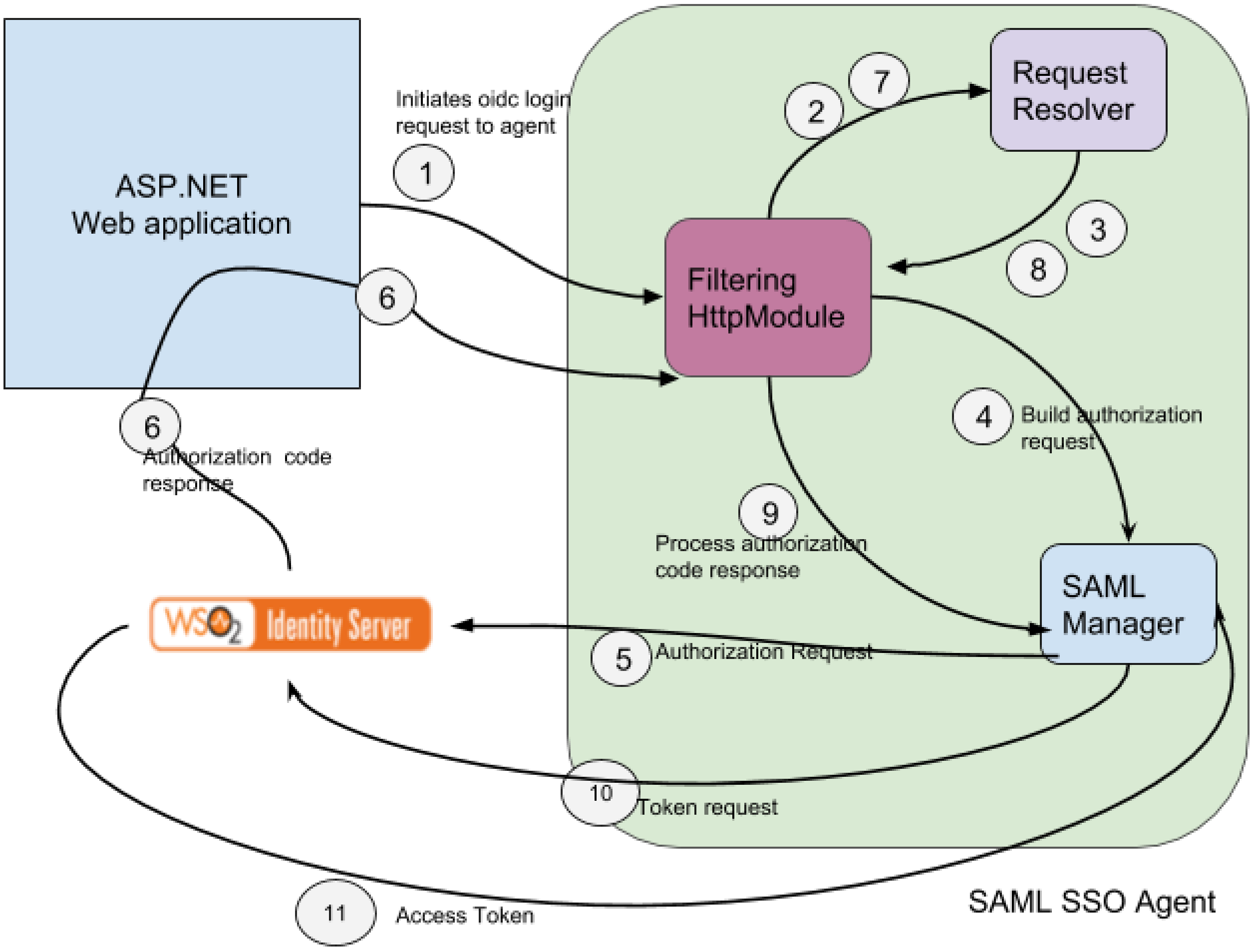
Subject Claim URI: ---Select---

Then do the above Step 1, 2, 3 again to create an SP for **Music Store** app. Service Provider Name in step 1 should be **music-store.** Assertion Consumer URL in Step 2 should be **http://localhost:60662/music-store/acs.**Rest of the thing can be done in the same manner as it was done for Trip Guider app. Then you might have to update attribute consuming service index properties in web config files of the samples.

5. Now you can press F5 in visual studio and try the app. Sample screenshots taken during the trying out of samples are show below

I. After pressing F5 Trip Guder and Music Store gets started in you default browser as shown below.
II. Then, press the login button in trip guider.
III. You get logged in successfully and you can see the claims you have received. (You receive claims which were configured using claim configuration while configuring the Service Provider only.)
IV. Press the login button in Music Store app and you should seamlessly get logged in without any prompt for login.(Occurrence of Single Sign on.)
V. Now you can try to logout of any of these two. I will try logging out from Music Store app.
VI. Then you can see that you have automatically logged out from trip Guider app too.(Occurrence of Single Logout)

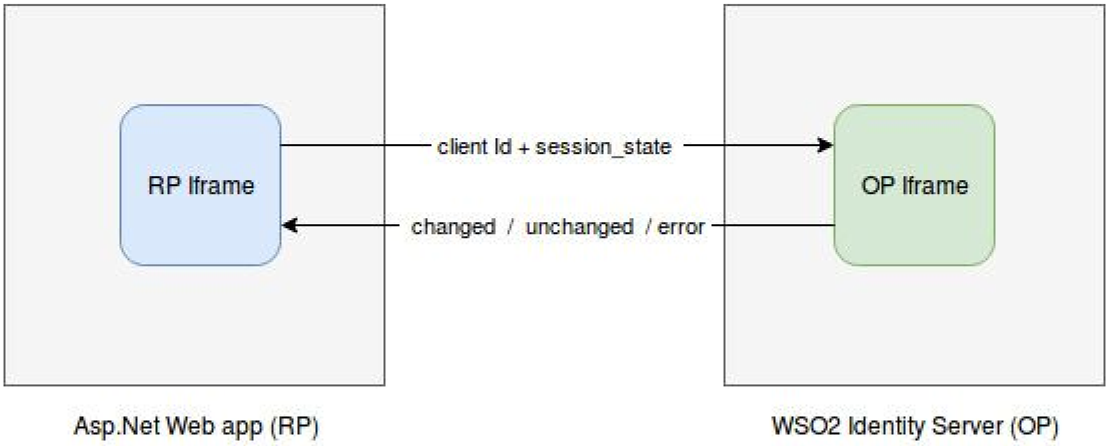
★ Architectural Diagram of the OIDC SSO Agent



2, 3, 7, 8 are related to resolving of the current request.

★ OIDC Agent supports Single Logout

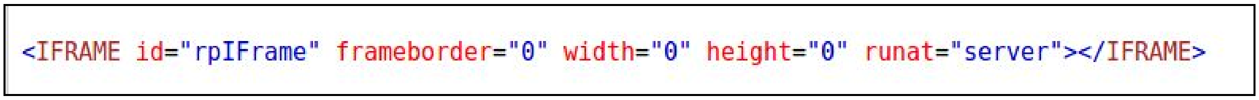
This agent uses the polling mechanism to achieve Single logout functionality. To elaborate more on this feature, there is a hidden RP(relying Party) iframe which keeps polling the OP(OpenId Provider) iframe periodically with a given time interval. As per the description found in OpenID Connect Specification, this mechanism can check the login status by polling at OP minimizing the network traffic. At each time when the OP is polled, it will return a value (could be either changed, unchanged or error). As supposed by the spec, when the return value is equals to “**changed**”, a passive request should be sent to OP and if the response is a login error it should be handled as SLO.


RP Iframe
client Id + session_state
changed / unchanged / error
OP Iframe
Asp.Net Web app (RP)
WSO2 Identity Server (OP)

★ How to enable Single Logout ?

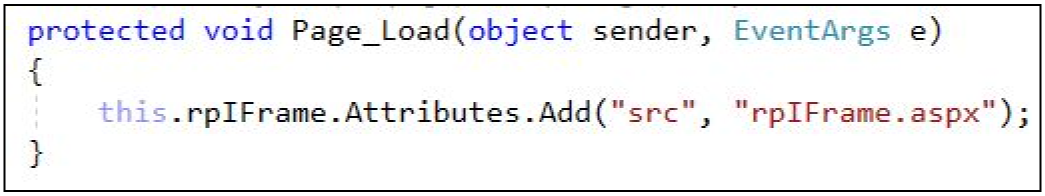
It is really simple to enable Single Logout for your asp.net web application. All you have to do is following simple steps below.

1. Add the **RpIFrame.asax** to your project. You can get it from here
2. Add the following line to the asp.net pages where you want to get logged out if SLO has occurred. For example, suppose you have a page that displays secure content and you want to redirect that page to Default.aspx once logout occur. Then, simply add the following line to your desired .aspx page.



```
<IFRAME id="rpIFrame" frameborder="0" width="0" height="0" runat="server"></IFRAME>
```

3. Then, in the code behind file of the .aspx file for which you have done the step 1, add the following code block there. This will wire up the src for the IFRAME.



```
protected void Page_Load(object sender, EventArgs e)
{
    this.rpIFrame.Attributes.Add("src", "rpIFrame.aspx");
}
```

4. Add “***PostLogoutRedirectUri***” property to the Web.config file of your ASP.NET web application with your desired location if you have not added that yet. Next, Add “**EnableSLO**” property and the value should be “**true**”.

Tips - *Important in all scenarios*

1. If your asp.net web app uses Newtonsoft.Json as a reference, Please make sure that you have updated Newtonsoft.Json the reference to latest version.Otherwise it will cause errors during runtime.
   (Note: by default you get this Newtonsoft.Json as a reference in your asp.net webapp when you create your project.)