

SAML SSO Agent for .NET Framework

★ Introduction

Suppose you have a 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 SAML 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 SAML authentication mechanism.

Moreover, you can use almost any scenario of SAML with SAML SSO agent seamlessly. For instance, you can use enable request signing and encryption assertions just by enabling a property in web.config file. And also you can enable signature, request /response validations in the same manner. And, you have the flexibility of choosing the form of HTTP-binding that matches your requirements too. Thus, you can enable SAML authentication in any form that you prefer.

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

If you plan to use SAML 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 steps

1. Add the agent.dll reference to your Asp.NET web application. (You can get this from the git repo)
2. Configure the mandatory properties in your ASP.NET web application's web.config file. Following image shows how it looks like after adding those properties to your web.config file.

Property	Description	Default Value
EnableSAML2SSOLogin	Enable SAML authentication	false
SAML2SSOURL	SSO URL	samlssso
SPEntityId	Service Provider Identifier	<i>null</i>
AssertionConsumerURL	Assertion Consumer URL	<i>null</i>
EnableResponseSigning	Enable signing responses with	false

	a X509 Certificate	
HTTPBinding	Protocol binding used throughout SAML flow	urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
EnableRequestSigning	Enable signing requests with a X509 Certificate	false
AttributeConsumingServiceIndex	Attribute consuming service index generated by Identity Server during SAML config. For the Service Provider	<i>null</i>
EnableSLO	Enable single logout for SPs	false
SLOURL	Single Logout URL	samllogout
PostLogoutRedirectUrl	Redirection URL after logout	<i>null</i>
CertFriendlyName	Friendly Name of the Certificate which is used.	wso2carbon

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

```

<appSettings>
  <add key="EnableSAML2SSOLogin" value="true" />
  <add key="SAML2SSOURL" value="samlss" />
  <add key="SPEntityId" value="trip-guider" />
  <add key="AssertionConsumerURL" value="http://localhost:49763/trip-guider/acs" />
  <add key="HTTPBinding" value="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" />
  <add key="EnableRequestSigning" value="true" />
  <add key="EnableResponseSigning" value="true" />
  <add key="AttributeConsumingServiceIndex" value="1561305277" />
  <add key="EnableSLO" value="true" />
  <add key="SLOURL" value="samllogout" />
  <add key="PostLogoutRedirectUrl" value="http://localhost:49763/trip-guider/Default" />
  <add key="CertFriendlyName" value="wso2carbon" />
</appSettings>

```

- Next, if you want to sign and validate your saml Requests /Responses, 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 SAML authentication mechanism.[*Note: The above mentioned **FilteringHttpModule** class is extended from IHttpModule. Click [here](#) for more information on IHttpModules.]*

```
<system.web>
  <httpModules>
    <add name="FilteringModule"
      type="org.wso2.carbon.identity.agent.FilteringHttpModule, Agent" />
    <add
      name="ApplicationInsightsWebTracking"
      type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule, Microsoft.AI.Web" />
  </httpModules>
</system.web>
```

```
<system.webServer>
  <validation validateIntegratedModeConfiguration="false" />
  <modules>
    <remove name="FilteringModule" />
    <add name="FilteringModule"
      type="org.wso2.carbon.identity.agent.FilteringHttpModule, Agent" />
    <remove name="ApplicationInsightsWebTracking" />
  </modules>
</system.webServer>
```

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 saml 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 "samlssso". And in the places that you have logout controls, it should be "samllogout".

*[Note: "samlssso" and "samllogout" are values that were configured under Step No: 2 for the properties **SAML2SSOURL** and **SLOURL** respectively. However, "samlssso" and "samllogout" are the default values for those two properties.]*

```
<a href="samlssso">Log In</a>
```

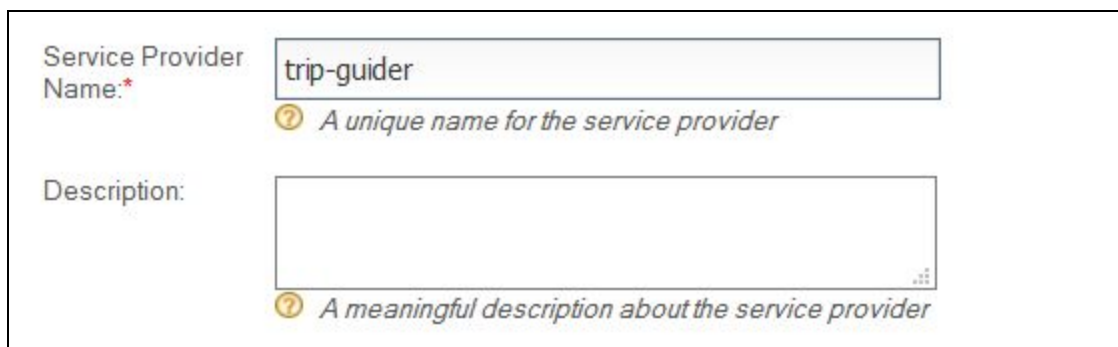
```
<a href="samllogout">Log Out</a>
```

Upon successful completion of the 6 steps above, you ASP.NET web application is enabled with SAML 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 SAML SSO authentication for both of them. Below diagrams show the configurations done for those two while testing.

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



The screenshot shows a configuration form for a Service Provider in WSO2 Identity Server. It contains two main fields: 'Service Provider Name' and 'Description'. The 'Service Provider Name' field is filled with 'trip-guider' and has a tooltip that reads 'A unique name for the service provider'. The 'Description' field is empty and has a tooltip that reads 'A meaningful description about the service provider'.

Service Provider Name:	<input type="text" value="trip-guider"/>
	<i>? A unique name for the service provider</i>
Description:	<input type="text"/>
	<i>? A meaningful description about the service provider</i>

2. Then navigate to Inbound Authentication Configuration -> SAML Web SSO Configuration.

Edit Service Provider

Issuer *

trip-guider

Assertion Consumer URLs *

Add

http://localhost:49763/trip-guider/acs
Delete

Default Assertion Consumer URL *

http://localhost:49763/trip-guider/acs

NameID format

urn:oasis:names:tc:SAML:1.1:nameid-form

Certificate Alias

wso2carbon

Response Signing Algorithm *

http://www.w3.org/2000/09/xmldsig#rsa-sha1

Response Digest Algorithm *

http://www.w3.org/2000/09/xmldsig#sha1

☒ Enable Response Signing

☒ Enable Signature Validation in Authentication Requests and Logout Requests

☒ Enable Assertion Encryption

☒ Enable Single Logout

SLO Response URL

Single logout response accepting endpoint

SLO Request URL

Single logout request accepting endpoint

☒ Enable Attribute Profile

☒ Include Attributes in the Response Always

3. You can configure the rest of the check boxes if you prefer 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	<input type="checkbox"/>	Delete

Subject Claim URI:

---Select---

5. Then, you have to navigate to the location: **Identity providers > Resident > Inbound Authentication Config > SAML2 Web SSO Config** of the management console and configure **Identity Provider Entity Id** field to a valid URI value as show in the image given below.

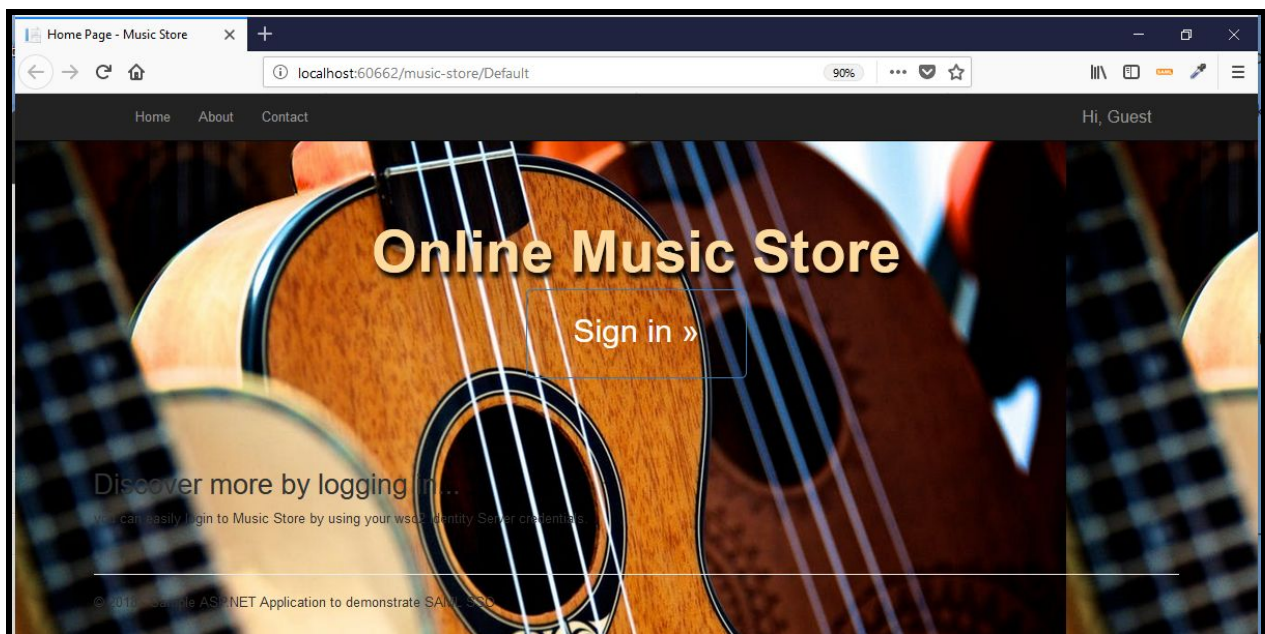
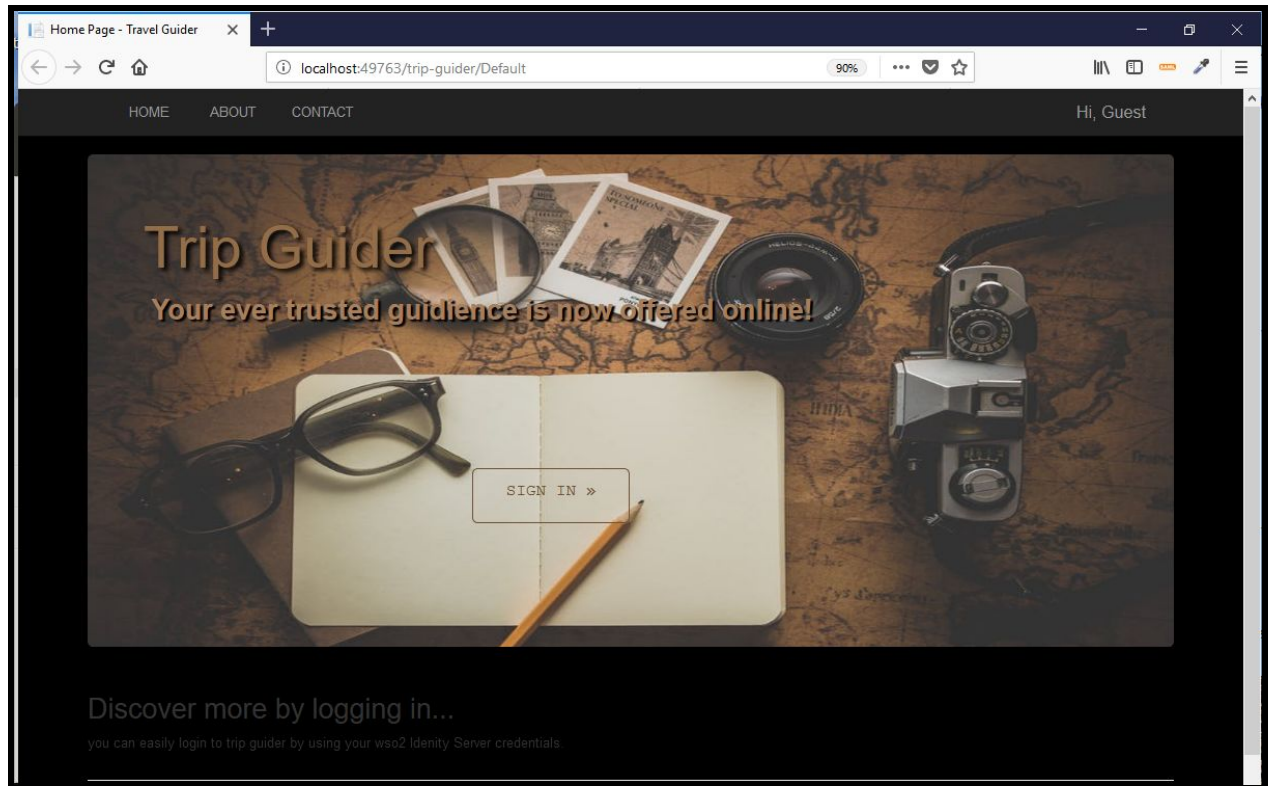
The screenshot shows the 'Inbound Authentication Configuration' window. Under the 'SAML2 Web SSO Configuration' tab, the 'Identity Provider Entity Id' field is set to 'wso2:org:IS'. Below this field is a hint: '? Enter identity provider's entity identifier value'. The 'Destination URLs' section contains a table with one entry: 'https://localhost:9443/samlso', with an 'Add' button to the right and a 'Delete' button (trash icon) below it. The 'SSO URL' is 'https://localhost:9443/samlso' and the 'Logout Url' is 'https://localhost:9443/samlso'. At the bottom, there is a 'Download SAML Metadata' button.

*[Note: Do this configuration, otherwise you will get an XML Exception since the default SAML NameIdentifier format used is **urn:oasis:names:tc:SAML:2.0:nameid-format:entity** and the default value for Identity Provider Entity ID is **localhost**]*

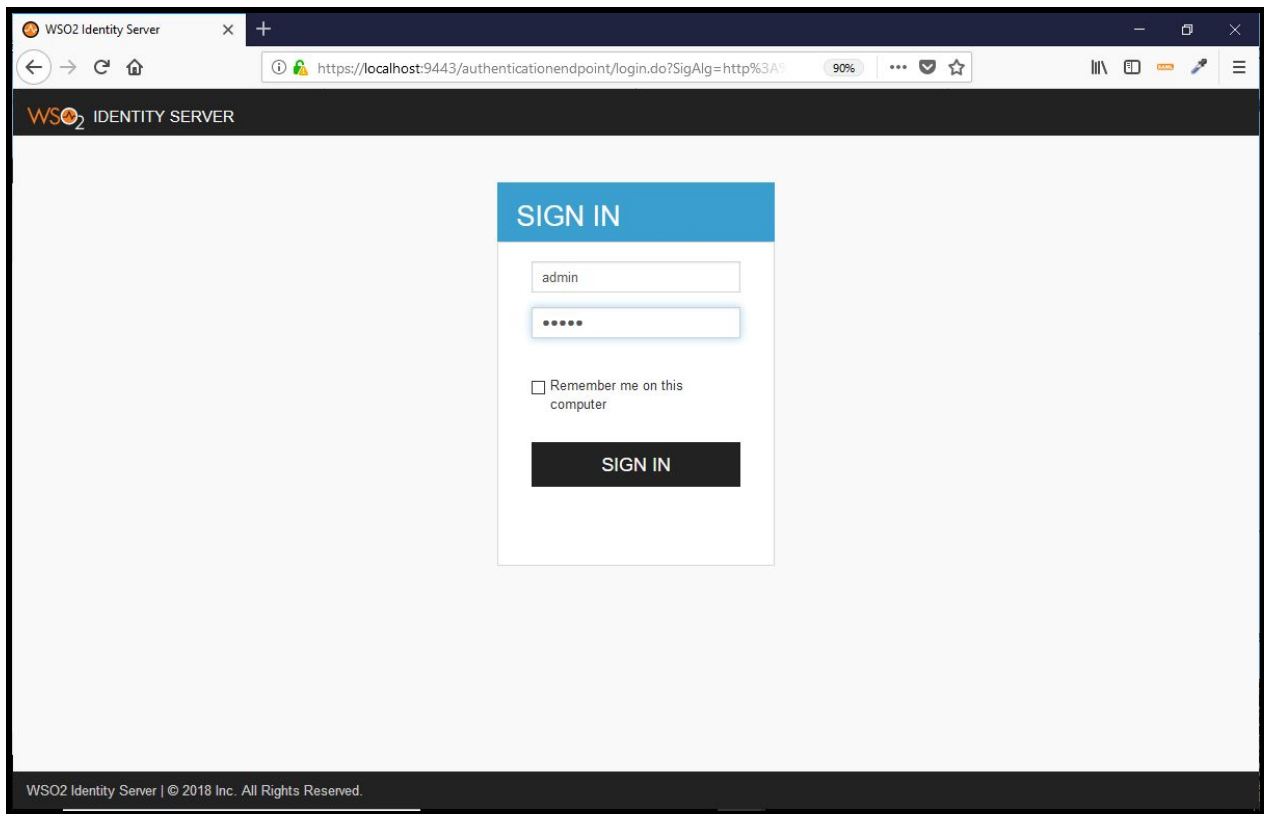
- Then do the above Step 1, 2, 3, 4 again to create an SP for **Music Store** app. Service Provider Name in step 1 should be **music-store**. Assertion Consumer URL in Step 2 should be <http://localhost:60662/music-store/acs>. Rest of the thing can be done in the same manner as it was done for Trip Guide app. Then you might have to update attribute consuming service index properties in web config files of the samples.

5. Now you can press Ctrl + F5 in visual studio and try out the apps. Sample screenshots taken while running those application are shown below.

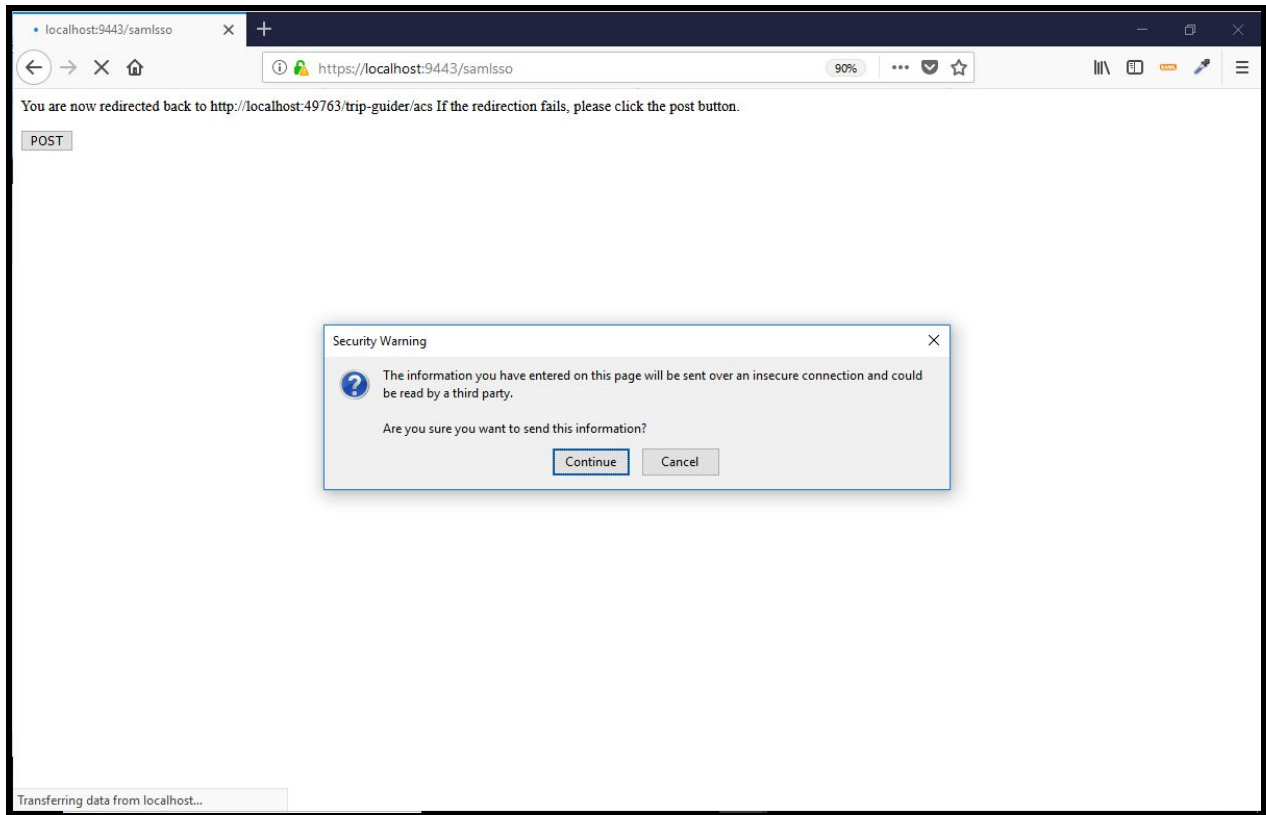
- I. After pressing Ctrl + F5 Trip Guder and Music Store gets started in you default browser as shown below.



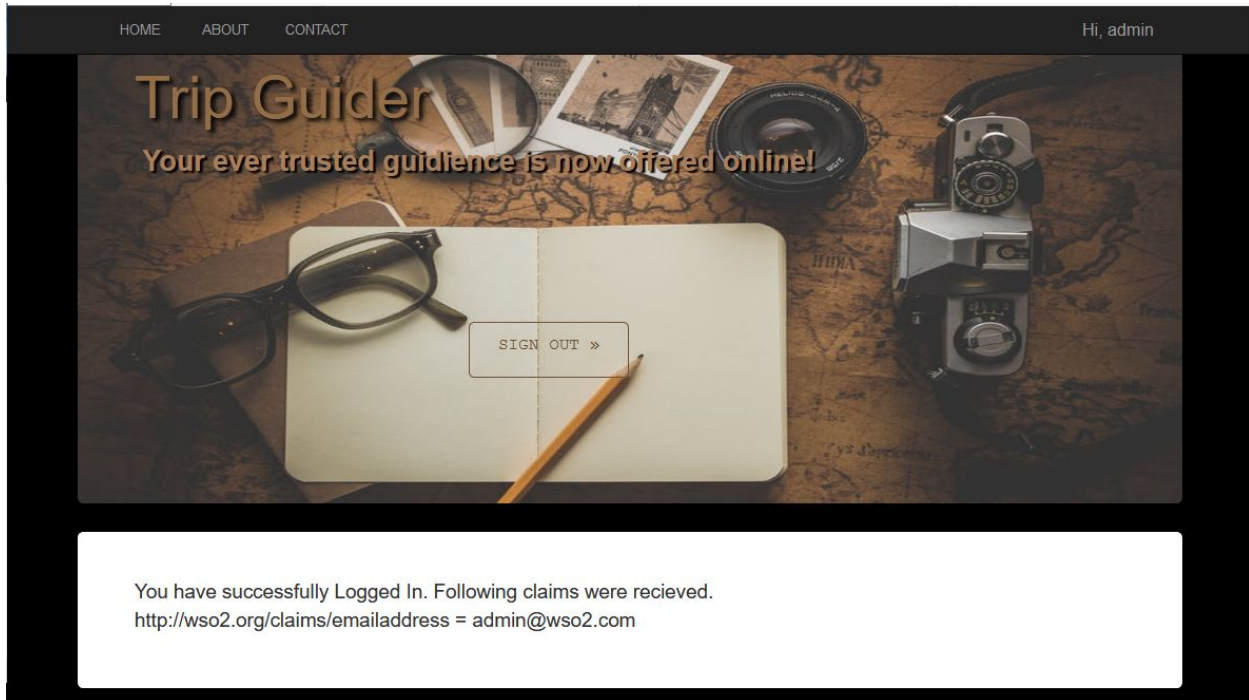
- II. Press the **Sign In** button in trip guider. Then you will be redirected to Wso2 Identity Server. You can test login using default admin credentials.



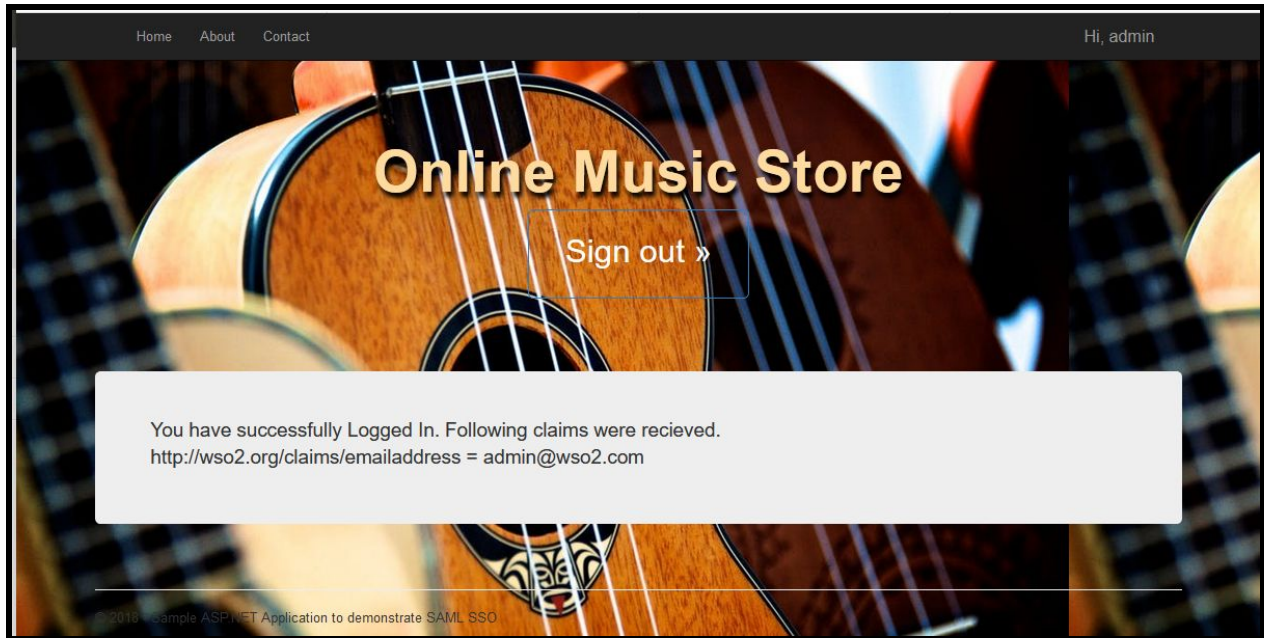
- III. You will get the below alert if you are using Http-POST binding. You can press the continue button and if the redirection fails you can click the POST button Provided.



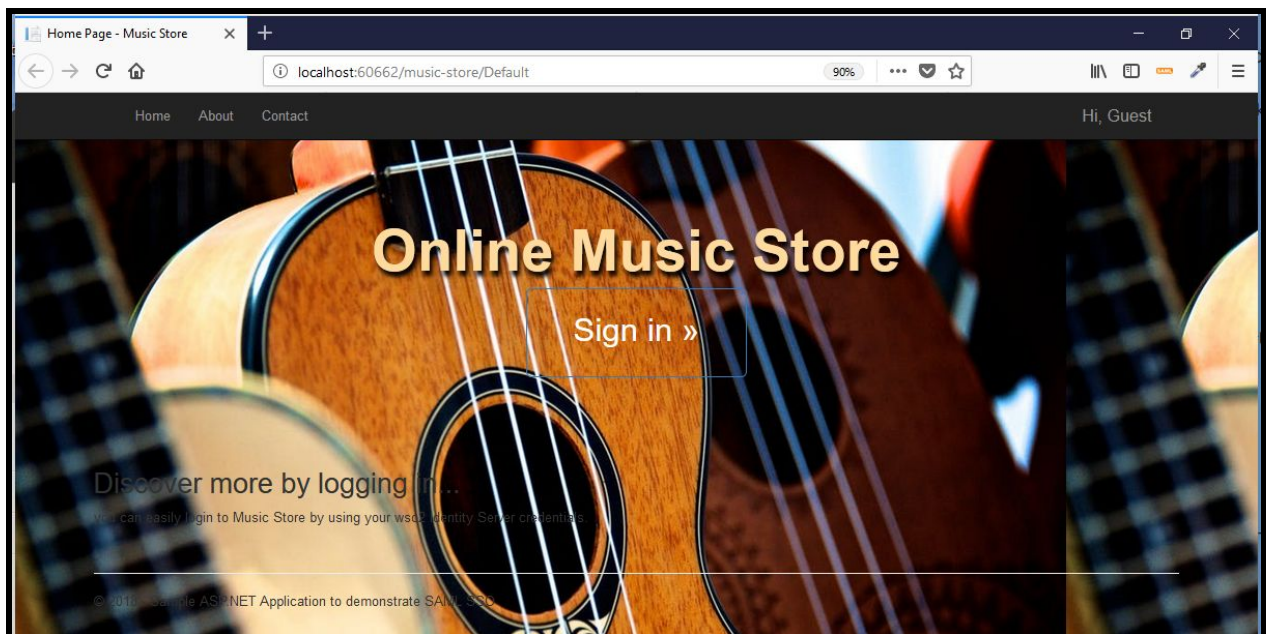
- IV. Then, You are logged in successfully and you can see the claims you have received. And you can see the Name ID of the subject at the top right corner of the window. *[Note: You receive claims which were configured using claim configuration while configuring the Service Provider only.]*



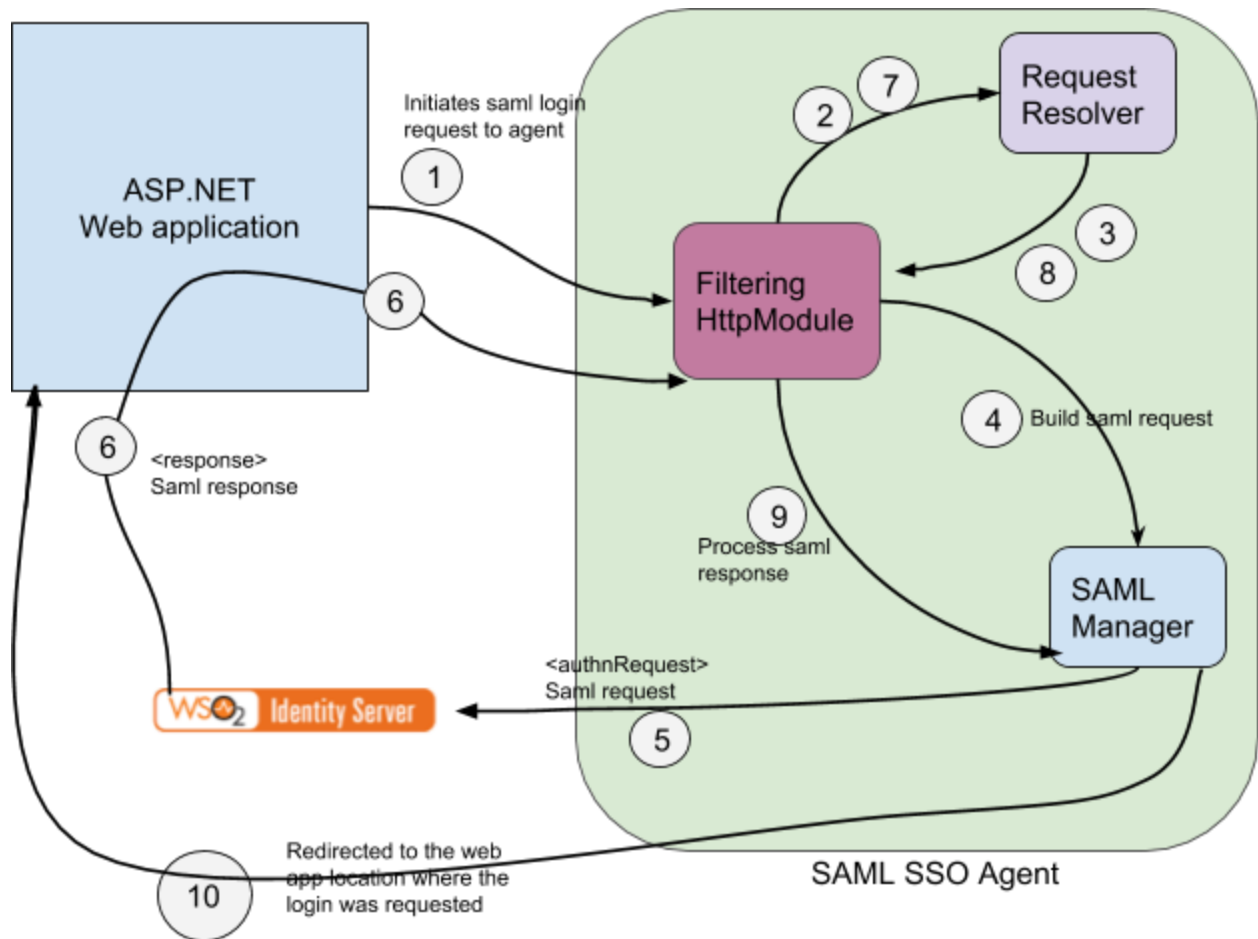
- V. Press the login button in Music Store app and you should seamlessly get logged in without any prompt for login.(Occurrence of Single Sign on.)



- VI. Now you can try to logout of any of these two. I will try logging out from Trip Guide app. Go ahead and click the logout button in Trip guide app. Then you can see that you have automatically logged out from Music Store app too.(Occurrence of Single Logout - back channel mode)



★ Architectural Diagram of the SAML SSO Agent



Note: No: 2, 7, 3, 8 are related to resolving of the current request.

★ Tips

1. You might have to install proper version of Java Cryptography Extension (JCE) unlimited strength jurisdiction policy files considering the java version. Otherwise it will cause in throwing Illegal key size exception in Identity Server when you enable assertion encryption.

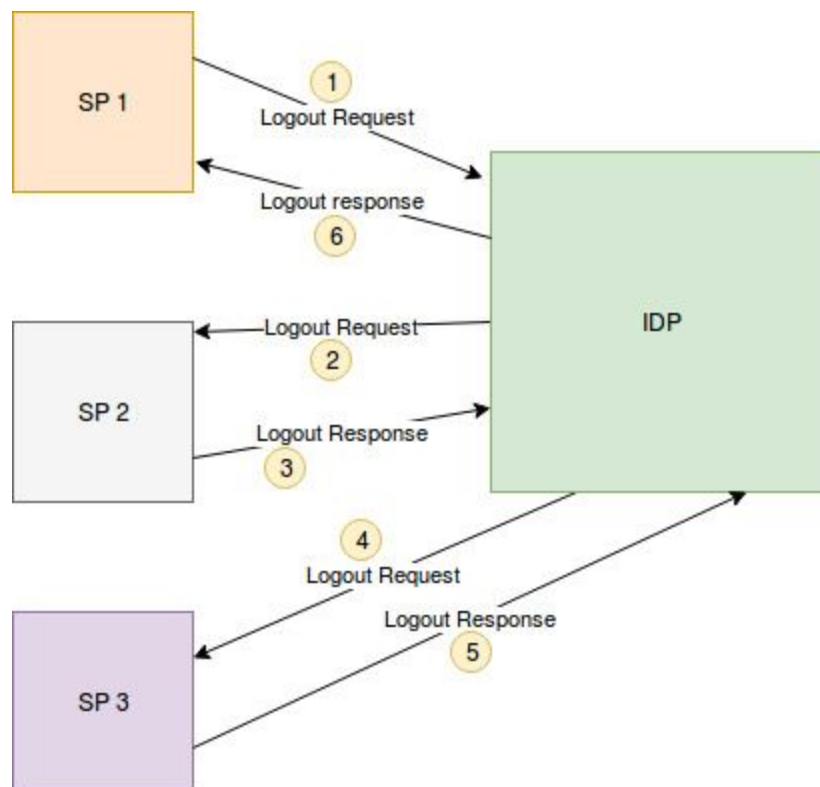
You can download JCE-8 from [here](#). Or else you can find the JCE for your java version from the official oracle java downloads.

[Note: Following content of the document is related to Single logout and just ignore is if you do not want to incorporate SLO for you apps.]

★ SAML Agent supports Single Logout

There are two possible modes of operation when it comes to SAML single logout. Further, there is a Front channel model and a back channel model to implement SAML single logout. SAML SSO agent supports Single Logout with compliance to SAML Back-Channel Single Logout model.

Following is a brief description on how saml SLO (back channel) model works. Suppose there are two service providers registered with wso2 Identity Server namely SP1 and SP2. SP1 and SP2 get logged in from a given browser and thus, Single Sign On has occurred. After some time user decides to logout from SP1. Then the SP1 Sends logout request to IDP. At the receipt of the Single Logout Request, IDP propperpergates logout request to other service providers that are participating SAML session (In our case it is SP2). At the receipt of SAML Logout request Sp2 Send a SAML Logout response with status 200 OK. Then after receiving logout response with 200 OK from participants (i.e. SP2), IDP sends logout response to the service provider who initiated logout (i.e. SP1). Thus single logout process gets completed. Below diagram depicts the SLO flow.



SAML Since there are plenty of tutorials online related to SAML SLO, I am not going to elaborate more on how SAML SLO works.

Getting back to our SAML SSO Agent, here we have a **SLOManagerIFrame.aspx** in our agent. This is used to achieve a polling mechanism from the client side to check if SLO has occurred. This SLOManagerIFrame communicates with Web methods found in **CheckSession.asmx** to achieve this goal of checking if SLO has occurred.

★ How to enable Single Logout with SAML SSO Agent ?

You can follow the steps given below to incorporate SAML Single logout to your asp.net web application.

1. Add the SLOManagerIFrame.aspx to your project. You can get it from a sample provided..
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 occurs. 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", "SLOManagerIFrame.aspx");
}
```

4. Add "**PostLogoutRedirectUrl**" 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**".

5. Then, enable Single logout for the service providers in IDP management console by checking "Enable single logout" checkbox.

<input checked="" type="checkbox"/> Enable Single Logout	
<i>SLO Response URL</i>	<input type="text"/>  <i>Single logout response accepting endpoint</i>
<i>SLO Request URL</i>	<input type="text"/>  <i>Single logout request accepting endpoint</i>

Management Console -> Service Providers -> List -> (choose SP) -> Inbound Authentication Configuration -> SAML SSO Configuration

After successful completion of above four steps, your app is not capable of performing SAML Single logout.

LOGS

NLOG logging framework is incorporated in agent to perform logging whenever in need. These logs are written to a file in a folder called **log** in your asp.net web application directory. Further, the configuration is already done for the file target and debugger target.