Scope

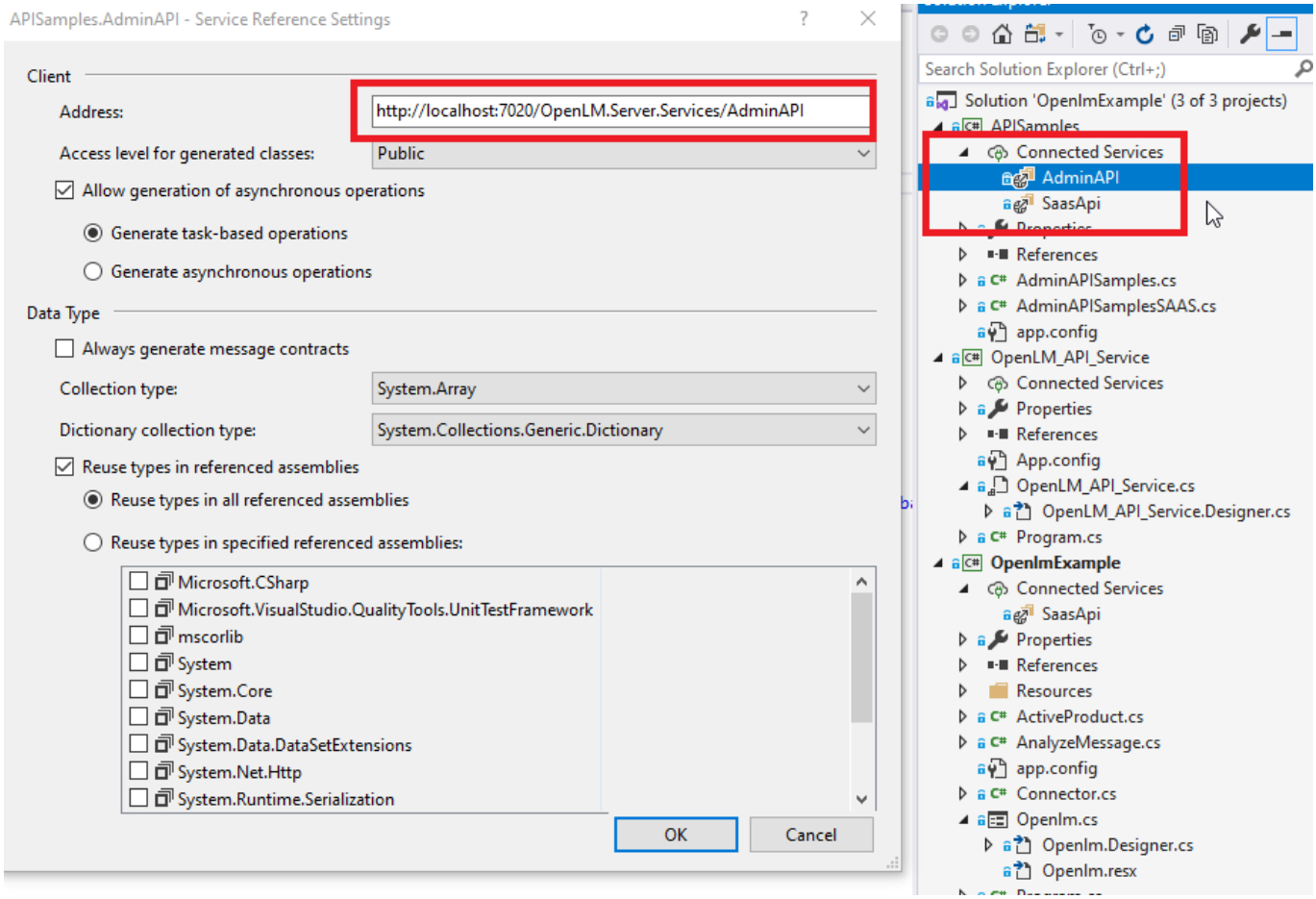
When Trusted Authentication (Windows Authentication) is enabled for OpenLM Server, using the OpenLM SOAP API may throw a **401 – Unauthorized error**. This document describes the steps required to solve this issue and allow a successful connection to the API.

The provided project sample already has these changes implemented.

Steps

1. In Visual Studio, open the Service Reference Settings window for the connected service you have set up (e.g. AdminAPI).

2. In the Address field, enter **http://*{OpenLM_Server_Host}*:7020/OpenLM.Server.Services/AdminAPI.** **{OpenLM_Server_Host}** should be replaced with the OpenLM Server hostname/IP. Click **OK.**



3. After adding the reference, NTLM security needs to be configured in **app.config** (if it has not been configured already)

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

<configuration>

<system.serviceModel>

<bindings>

<basicHttpBinding>

<binding name="basicHttpBindingConfiguration_IAdminAPI">

 <security mode="TransportCredentialOnly">
```

```
 <transport clientCredentialType="Ntlm"
proxyCredentialType="Windows"/>
 </security>
 </binding>
<binding name="BasicHttpBinding_Saas"/>
<binding name="gzip"/>
</basicHttpBinding>
<wsHttpBinding>
<binding name="mtop">
<security mode="None"/>
</binding>
</wsHttpBinding>
</bindings>
<client>
<endpoint
address="http://localhost:7020/OpenLM.Server.Services/AdminAPI"
binding="basicHttpBinding"
bindingConfiguration="basicHttpBindingConfiguration_IAdminAPI"
contract="AdminAPI.IAdminAPI"
name="basicHttpBindingConfiguration_IAdminAPI"/>
<endpoint
address="http://saas.openlm.com/SaasService/Service.svc/soap"
binding="basicHttpBinding"
```

```
bindingConfiguration="BasicHttpBinding_Saas" contract="SaasApi.Saas"
name="BasicHttpBinding_Saas"/>

</client>

</system.serviceModel>

<startup><supportedRuntime version="v4.0"
sku=".NETFramework,Version=v4.7.2"/></startup></configuration>
```

Examples

In the APISamples project, in the AdminAPISamples.cs file, all test methods use the CreateBaseInfo method that uses SessionID, which is the authentication result.

```
[...]
private string GetSessionID(AdminAPIClient client)

{

LoginFormSettingsResponse response = client.GetLoginFormSettings(new
LoginFormSettingsRequest());

if (response.UserAuthenticationRequired)

{

return AdminAPIAuthentication(client, response.ShowWinAuth);

}

else//No authentication required

{
```

```
return string.Empty;

}

}

private static string AdminAPIAuthentication(AdminAPIClient client,
bool useWindowsAuthentication)

{

UserAuthenticationRequest userAuthenticationRequest = new
UserAuthenticationRequest { TrustedAuthentication =
useWindowsAuthentication };//Windows authentication

if (!useWindowsAuthentication)//OpenLM server authentication

{

userAuthenticationRequest.UserName = UserName;

userAuthenticationRequest.Password = Password;

}/* If OpenLM server authentication is required uncheck this code

else

{

userAuthenticationRequest.UserName = UserName;

userAuthenticationRequest.Password = Password;

userAuthenticationRequest.TrustedAuthentication = false;

}*/

UserAuthenticationResponse userAuthenticationResponse =
```

```
client.PerformUserAuthentication(userAuthenticationRequest);

if (userAuthenticationResponse.Error != null)

{

throw new Exception(userAuthenticationResponse.Error.Message);

}

return userAuthenticationResponse.SessionID;

}
```

The second example is the OpenLM_API_Service project which shows how to run a service. The same authentication method is used.

From **OpenLM_API_Service\OpenLM_API_Service.cs**

```
protected override void OnStart(string[] args)

{

GetSessionID(new AdminAPIClient());

}

protected override void OnStop()

{

}

private string GetSessionID(AdminAPIClient client)
```

```
{

LoginFormSettingsResponse response = client.GetLoginFormSettings(new
AdminAPI.LoginFormSettingsRequest());

if (response.UserAuthenticationRequired)

{

return AdminAPIAuthentication(client, response.ShowWinAuth);

}

else//No authentication required

{

WriteServiceResulToTextFile("No authentication required");

return string.Empty;

}

}

private static void WriteServiceResulToTextFile(string result)

{

File.WriteAllText(ConfigurationManager.AppSettings.Get("ResultFile"),
$"{DateTime.Now.ToString()} {result}");

}

private static string AdminAPIAuthentication(AdminAPIClient client,
bool useWindowsAuthentication)

{
```

```
UserAuthenticationRequest userAuthenticationRequest = new
UserAuthenticationRequest { TrustedAuthentication =
useWindowsAuthentication };//Windows authentication

if (!useWindowsAuthentication)//OpenLM server authentication

{

WriteServiceResulToTextFile("OpenLM server authentication required");

}

UserAuthenticationResponse userAuthenticationResponse =
client.PerformUserAuthentication(userAuthenticationRequest);

if (userAuthenticationResponse.Error != null)

{

WriteServiceResulToTextFile($"OpenLM server windows authentication
error {userAuthenticationResponse.Error.Message}");

throw new Exception(userAuthenticationResponse.Error.Message);

}

WriteServiceResulToTextFile($"OpenLM server windows authentication
successful, created session id:
{userAuthenticationResponse.SessionID}");

return userAuthenticationResponse.SessionID;

}
```

The Windows service writes the result to a result file as defined in **app.config**

```
<appSettings>
```

```
<add key="ResultFile" value="C:\\Temp\\OpenLM_API_Service.txt"/>

</appSettings>
```

The result file contains the result of the authentication request: either success or failure with an explanation.