Quick Start

Creating a Remote Client to Connect to an SDL Web Core Service

Sep 2017 – SDL Web

Contents

SDL Web Core Service

Information

The Core Service is a Web service that allows applications to interact with the Content Manager. For example, Content Manager clients such as Experience Manager and Content Manager Explorer interact with the Content Manager through the Core Service, and you can use the Core Service to integrate external systems with SDL Web. You also use the Core Service to implement Automatic Activities to control Workflow.

The Core Service is a SOAP Web service based on Windows Communication Foundation. It supports WS-I Basic Profile 1.1 and more advanced WS-* protocols.

This document will show you one method of connecting to an SDL Web Core Service from a remote machine; after which you can perform tasks in the Content Manager as required.

Pre-requisites

The following pre-requisites must be met before you can setup a Core Service client:

- Access to SDL Web 8.5 (remote server)
- Visual Studio

SDL Web Content Manager

1. Find the version of the netTcp client that the CMS is using: Access your Content Manager server (where Tridion is installed) and navigate to:
   %TRIDION_HOME%/bin/client/CoreService

   a. Open the file **Tridion.ContentManager.CoreService.Client.dll.config**

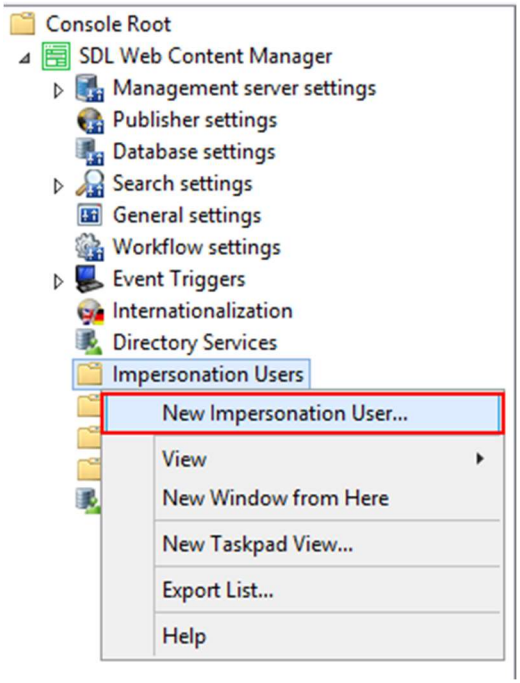
   b. Take a copy or note of the netTcp client, which we will use in this introductory example.

```
<endpoint name="netTcp_201603" address="net.tcp://localhost:2660/CoreService/201603/
netTcp" binding="netTcpBinding" bindingConfiguration="netTcp" contract="
Tridion.ContentManager.CoreService.Client.ISessionAwareCoreService" />
```

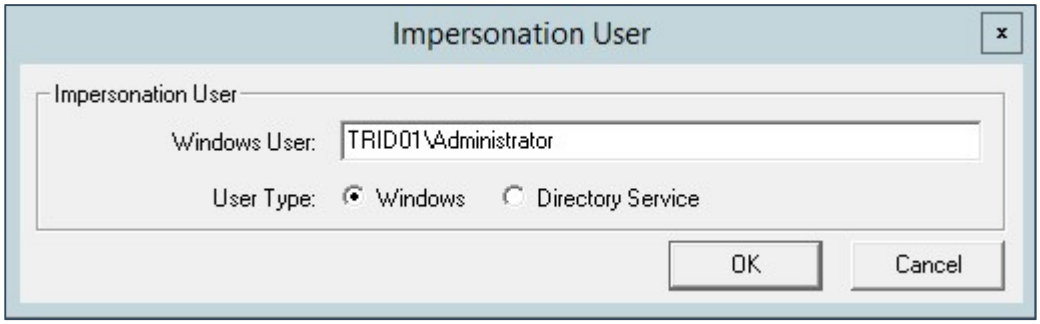
   Note! You will need to mirror this entry later in your Application's Web.config.

2. Create a CMS Impersonation User so that we can use this account to authenticate the Web Application's client connection: From the CMS open the SDL Web Content Manager MMC.

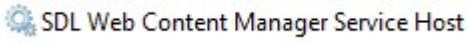
3. From the SDL Web Content Manager MMC right-click **Impersonation Users** and select **New Impersonation User**:



   a. Add the new Impersonation User as shown or select your own User that you want to use:



   b. Click OK.
   c. Close the MMC and restart the CMS **Service Host** service to apply your changes.



4. Ensure that the CMS firewall is set to allow traffic through **port 2660**.

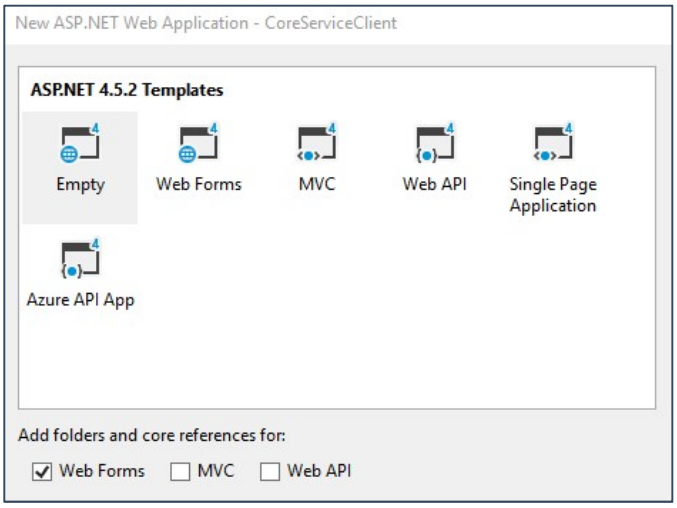
That is it for the CMS side of things!

Creating a Web Application

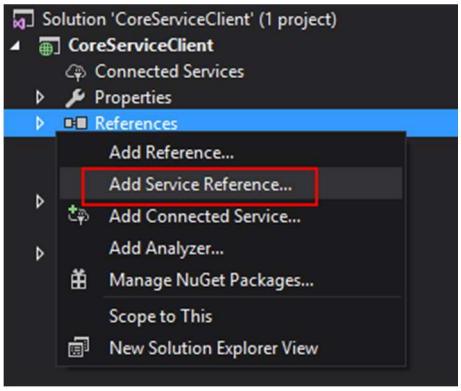
In this example you are going to simply create a basic Web Forms Application that will get the current version of the CMS' API. There is obviously a lot more you can do with the API once you have the client running and you should access the API documentation at http://docs.sdl.com for further details.

1. Open Visual Studio and create a new Project: **File – New – Project**. Select the Template: **Visual C# - Web – ASP.Net Web Application (.Net Framework)**

   a. Give the project a suitable name, such as **CoreServiceClient**.
   b. Select the Empty Template and check the Web forms folders and core references as shown.



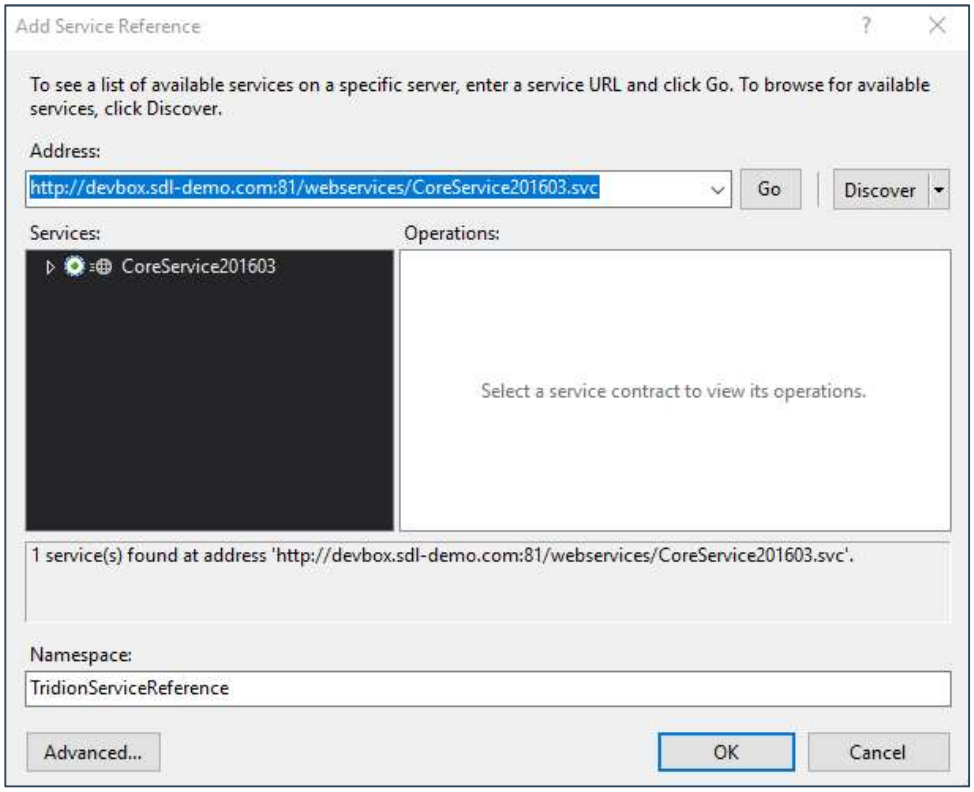
   c. Click OK.
2. In the Solution Explorer of the new Project, right-click the **References** node and select **Add Service Reference**... as shown.



   a. Enter the **Address** or IP to your server along with the route to the Core Service and click **Go**. For example: **http://mycms.com:81/webservices/CoreService201603.svc**

b. You will see the Service in the Services window if the connection is successful.

c. Give the service reference a name, such as **TridionServiceReference** and click **OK**.



d. Your project now has access to the CMS' Core Service.

3. Open the Web.config, which will now contain the added Service Reference and add the following entry to the <bindings> node.

```
<netTcpBinding>
        <binding name="netTcp" transactionFlow="true"
transactionProtocol="OleTransactions" maxReceivedMessageSize="10485760">
          <readerQuotas maxStringContentLength="10485760" maxArrayLength="10485760"/>
        </binding>
        <binding name="streamDownload_netTcp" maxReceivedMessageSize="2147483647"
transferMode="StreamedResponse" sendTimeout="00:10:00"/>
        <binding name="streamUpload_netTcp" maxReceivedMessageSize="2147483647"
transferMode="StreamedRequest" receiveTimeout="00:10:00"/>
      </netTcpBinding>
```

a. Now update the <client> node with the following. Note, this was taken from the binding in the CMS previously.

```
<endpoint name="netTcp_201603" address="net.tcp://devbox.sdl-
demo.com:2660/CoreService/201603/netTcp"
        binding="netTcpBinding" bindingConfiguration="netTcp"
              contract="TridionServiceReference.ISessionAwareCoreService" />
```

Note! Change devbox.sdl-demo.com:2660 to reflect your own CMS domain or IP.

b. In the Web.config's <configuration> node, add the login credentials for the Impersonation User created in the CMS previously.

```
<!-- App Settings -->
<appSettings>
	<add key="coreservice.username" value="Administrator"/>
	<add key="coreservice.password" value="SDLWeb8!"/>
	<add key="coreservice.domain" value="TRID01"/>
</appSettings>
```

Note! Update the credentials to reflect your own define user in the CMS.

c. Save and close the Web.config

Note! The full Web.config is available in Annex A to this document.

4. Now create a Web Form to manage the client connection and make a call through the Core Service API. In your Visual Studio Solution Explorer right-click the Project node and select Add – Web Form.

a. Name the Web Form **Main**.
b. Add the following form control to the Main.aspx file:

```
<body>
<form id="form1" runat="server">
	<div>
	The CMS' API is version:
	<asp:Label ID="apiLbl" runat="server" Text="Only the API knows" />
	</div>
</form>
</body>
```

c. Now switch to the **Main.aspx.cs** class file and add the class code as shown. Note, the full class is at Annex B.

Add the following namespaces:

```
using System;
using System.Net;
using System.ServiceModel;
using CoreServiceClient.TridionServiceReference;
```

Class variables:

```
//CMS Impersonation credentials
 private string username;
 private string password;
 private string domain;

 //tcp communication
 private ISessionAwareCoreService _client;
 private ChannelFactory<ISessionAwareCoreService> _factory;
```

Below the **PageLoad** method add the following:

```
public ISessionAwareCoreService client
 {
     get
     {

         if (_client == null)
         {
//tcp communication
_factory = new C
hannelFactory<ISessionAwareCoreService>("netTcp_201603");
             if (!string.IsNullOrEmpty(username) &&
             !string.IsNullOrEmpty(password) && !string.IsNullOrEmpty(domain))
             {
NetworkCredential networkCredential = new NetworkCredential(username, password,
domain);
_factory.Credentials.Windows.ClientCredential = networkCredential;
             }
             _client = _factory.CreateChannel();
         }
         return _client;
     }
   }
```

In the PageLoad method add the following:

```
//Set credentials from Web.config
username =
System.Configuration.ConfigurationManager.AppSettings.Get("coreservice.username");
password =
System.Configuration.ConfigurationManager.AppSettings.Get("coreservice.password");
domain =
System.Configuration.ConfigurationManager.AppSettings.Get("coreservice.domain");
//Call the API version from the CMS' Core Service API and display it in the Web Form
apiLbl.Text = client.GetApiVersion();
```

Result

Annex A – Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <!-- App Settings -->
  <appSettings>
    <add key="coreservice.username" value="Administrator"/>
    <add key="coreservice.password" value="SDLWeb8!"/>
    <add key="coreservice.domain" value="TRID01"/>
    <add key="settingsGroupID" value="tcm:0-15-65568"/>
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.3.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
        warningLevel="4" compilerOptions="/langversion:6 /nowarn:1659;1699;1701"/>
      <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=1.0.3.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"
        warningLevel="4" compilerOptions="/langversion:14 /nowarn:41008
/define:_MYTYPE=\&quot;Web\&quot; /optionInfer+"/>
    </compilers>
  </system.codedom>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="basicHttp">
          <security mode="TransportCredentialOnly">
            <transport clientCredentialType="Windows" />
          </security>
        </binding>
        <binding name="streamDownload_basicHttp" messageEncoding="Mtom">
          <security mode="TransportCredentialOnly">
            <transport clientCredentialType="Windows" />
          </security>
        </binding>
        <binding name="streamUpload_basicHttp" messageEncoding="Mtom" />
        <binding name="Batch_basicHttp">
          <security mode="TransportCredentialOnly">
            <transport clientCredentialType="Windows" />
          </security>
        </binding>
      </basicHttpBinding>
      <wsHttpBinding>
        <binding name="wsHttp" transactionFlow="true" />
      </wsHttpBinding>
    </bindings>
    <client>
```

```
      <endpoint address="http://devbox.sdl-
demo.com:81/webservices/CoreService201603.svc/basicHttp"
        binding="basicHttpBinding" bindingConfiguration="basicHttp"
        contract="TridionServiceReference.ICoreService" name="basicHttp" />
      <endpoint address="http://devbox.sdl-
demo.com:81/webservices/CoreService201603.svc/streamDownload_basicHttp"
        binding="basicHttpBinding" bindingConfiguration="streamDownload_basicHttp"
        contract="TridionServiceReference.IStreamDownload"
name="streamDownload_basicHttp" />
      <endpoint address="http://devbox.sdl-
demo.com:81/webservices/CoreService201603.svc/streamUpload_basicHttp"
        binding="basicHttpBinding" bindingConfiguration="streamUpload_basicHttp"
        contract="TridionServiceReference.IStreamUpload" name="streamUpload_basicHttp"
/>
      <endpoint address="http://devbox.sdl-
demo.com:81/webservices/CoreService201603.svc/batch_basicHttp"
        binding="basicHttpBinding" bindingConfiguration="Batch_basicHttp"
        contract="TridionServiceReference.ICoreServiceBatch" name="Batch_basicHttp" />
      <endpoint address="http://devbox.sdl-
demo.com:81/webservices/CoreService201603.svc/wsHttp"
        binding="wsHttpBinding" bindingConfiguration="wsHttp"
contract="TridionServiceReference.ISessionAwareCoreService"
        name="wsHttp">
        <identity>
          <dns value="localhost" />
        </identity>
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

Annex B – Main.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Main.aspx.cs"
Inherits="TridionRedirectManager.Main" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            The current API Version is: <asp:Label ID="apiLbl" runat="server"
Text="Not Found" />
        </div>
    </form>
</body>
</html>
```

Annex C – Main.aspx.cs

```
using CoreService.TridionServiceReference;
using System;
using System.Net;
using System.ServiceModel;

namespace CoreService
{
    public partial class Main : System.Web.UI.Page
    {
        //CMS Impersonation credentials
        private string username;
        private string password;
        private string domain;

        //tcp communication
        private ISessionAwareCoreService _client;
        private ChannelFactory<ISessionAwareCoreService> _factory;
        protected void Page_Load(object sender, EventArgs e)
        {
            //Set credentials from Web.config
            username =
System.Configuration.ConfigurationManager.AppSettings.Get("coreservice.username");
            password =
System.Configuration.ConfigurationManager.AppSettings.Get("coreservice.password");
            domain =
System.Configuration.ConfigurationManager.AppSettings.Get("coreservice.domain");
            //Get the API Version from the Core Service
            apiLbl.Text = client.GetApiVersion();
        }

        public ISessionAwareCoreService client
        {
            get
            {

                if (_client == null)
                {
                    //tcp communication
                    _factory = new
ChannelFactory<ISessionAwareCoreService>("netTcp_201603");
                    if (!string.IsNullOrEmpty(username) &&
!string.IsNullOrEmpty(password) && !string.IsNullOrEmpty(domain))
                    {
                        NetworkCredential networkCredential = new
NetworkCredential(username, password, domain);
                        _factory.Credentials.Windows.ClientCredential =
networkCredential;
                    }
                    _client = _factory.CreateChannel();
                }
                return _client;
            }
        }
    }
}
```

Annex D – Troubleshooting

1. When you run the project or build the project you get multiple reference.cs errors suggesting the Service Reference 'TridionServiceReference' cannot be found.

Possible solution:
When you add the Service Reference to the project, run incremental builds as follows:

Create the service reference – build.
Add the netTcp binding to the Web.config – build.
Add the netTcp client to the web.config – build.
Save the Web.config and then continue to write the code to interact with the Core Service.

2. Syntax errors:

If you copied the text in the document then ensure the syntax was pasted as it should be.

SDL (LSE: SDL) is the global innovator in language translation technology, services and content management. For more than 20 years, SDL has transformed business results by enabling nuanced digital experiences with customers across the globe so they can create personalized connections anywhere and on any device. Are you in the know? Find out more at SDL.com.