# Module 2 Lab A – Querying the discovery Service

Time 20 Minutes

##### Objective: Use the discovery service to get the endpoints for each organisations WebApplication, OrganizationService and OrganizationDataServices

**Step 1: Use the Settings area to get the URL for the discovery service**



|  |  |
| --- | --- |
| 1 | Using internet explorer navigate to http:[yourDynamicsDomain].crm11.dynamics.com.  Login and then navigate to the settings area and click on Customisations. |

|  |  |
| --- | --- |
| 2 | Click on developer resources and then copy note down (copy) the URL of the discovery service. We will be using this endpoint to retrieve information about other endpoints that will allow us to interact with CRM data and Metadata later on. |

**Step 2: Generating a client credential**

|  |  |
| --- | --- |
| 1 | We need to create and register a client credential using the device registration tool that is supplied with the SDK.  Open up Visual Studio 2015 (run as Administrator) and open the following project DeviceRegistration.csproj located C:\Program Files\Microsoft SDKs\SDK\Tools\DeviceRegistration |
| 2 | Open up the CommandLineParameters.cs and look at the command line parameters that can be used when the console application is running.  The /operation parameter takes the value register in order to register a client credential alternatively you can use the “show” switch to display an existing device credentials.  if you don’t want a randomly generated device username and password then there are parameters that you can set to assign name and password that you choose |
| 3 | Open up the properties for the project and set the command line parameters as follows /operation:show    You should see the following output    If the device were already registered the username password would have been revealed.  Change the Command line arguments the ones listed below. Feel free to change the Name and Password values if you want.  /operation:Register |
| 4 | Save the changes and the hit ctrl+F5 to run without debugging and record the name and password that is listed in the output window. We will use these credentials later on. If you want to see the registered credentials with the app change the command line parameter operation to Show instead of register. |
|  |  |

**Step 3: Creating a Windows Application to Query the Discovery Service**

|  |  |
| --- | --- |
| 1 | Open up Visual Studio 2015 (run as Administrator) and create a Windows Form Application called Dynamics365Disco in the C:\QACRMDEV\Labs\Module2 folder |
| 2 | Resize the form and place a Listbox and button on the form as sown below. To save time don’t bother renaming the controls |
| 3 | Right click on the project references and browse to and add a reference to Microsoft.Xrm.Sdk.dll located in C:\Program Files\Microsoft SDKs\SDK\Bin  Note that this is one of the assemblies that was update via NuGet.  In addition, by selecting the assemblies tab add a reference to System.ServiceModel |
| 4 | On the designer double click button1 to open up the editor in order to write code that will respond to the click event.  private void button1\_Click(object sender, EventArgs e)  {    }  Before adding any code to the click event add the following namespaces to the list at the top.    using Microsoft.Xrm.Sdk.Discovery;  using Microsoft.Xrm.Sdk.Client;  using System.ServiceModel.Description;  using System.Net; |

|  |  |
| --- | --- |
| 5 | Declare two private fields of type ClientCredentials named userCred and deviceCred in the Form1 class . These will store the User and Device credentials we will be using to access the discovery service. Set the security protocol default to TLS 1.2 as shown in the listing below.  namespace DiscoDemo  {  public partial class Form1 : Form  {  private ClientCredentials userCred; // stores user credentials  private ClientCredentials deviceCred; // stores device credentials  public Form1()  {  ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;  InitializeComponent();  } |
|  |  |

|  |  |
| --- | --- |
| 6 | In the Buttons’s click event add code to initialise the 2 client credentials variables using the username password combinations that are correct for your account. |
| 7 | Below the code that you have just added to the click event handler but still within the button1\_click add the create an instance of a DiscoveryServiceProxy class, passing it parameter values that are valid for your Dynamics 365 tenancy  string discoUrl = "https://[url of discovery service.svc]";  using ( var serviceProxy = new DiscoveryServiceProxy(new Uri(discoUrl),  null,  userCred,  deviceCred))  {  }  Replace the url of discovery service placeholder with the actual url you recorded earlier in the lab. It should end with .svc . |
| 8 | Within the using statement add a call to a function called getInfo passing in the serviceProxy as a parameter and returning an OrganizationServiceProxy object  OrganizationServiceProxy orgProxy = getInfo(serviceProxy);  The getInfo function now needs to be written.  The visual studio editor can generate a stub function for you to complete (hint place your cursor over the function and hit ctrl+.) |

|  |  |
| --- | --- |
| 9 | **Replace** the call to throw a new NotImplementedException by copying and pasting the following into the getInfo method  // Retrieve details about all organizations discoverable via the  // Discovery service.  RetrieveOrganizationsRequest orgsRequest =  new RetrieveOrganizationsRequest()  {  AccessType = EndpointAccessType.Default,  Release = OrganizationRelease.Current  };  RetrieveOrganizationsResponse organizations =  (RetrieveOrganizationsResponse)serviceProxy.Execute(orgsRequest);  foreach (KeyValuePair<EndpointType, string> endpoint in organizations.Details[0].Endpoints) // get the endpoints for the first organisation  {  listBox1.Items.Add(string.Format(" Name: {0}", endpoint.Key));  listBox1.Items.Add(string.Format(" URL: {0}", endpoint.Value));  if (endpoint.Key == EndpointType.OrganizationService)  {  OrganizationServiceProxy organizationProxy = new OrganizationServiceProxy(new Uri(endpoint.Value), null, userCred, deviceCred);  // organizationProxy.EnableProxyTypes(); // enables support for early bound proxy types  return organizationProxy;  }  }  return null;  Note that the foreach loop is iterating over the collection of endpoints and adding the endpoint Name (Key) and the Url (Value) to the listbox |
| 10 | Run the app and click on the button. Hopefully you’ll see something similar to the screenshot below |
| 11 | At the conclusion of this lab you will have used the discovery service api to query for available organisational endpoints. |