Microsoft Dynamics 365/CRM 2016
Extending

Lab Manual

Lab 2a: Modifying Navigation	3
Lab 2b: Working with Command Buttons	28
Lab 3: Form Scripting	44
Lab 4: Using the Organization Service	66
Lab 5: Querying Dynamics 365 Data	84
Lab 6: Creating a Custom Workflow Activity	103
Lab 7: Creating a Plug-in	115

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should

Contents

Lab 02A – Modifying Navigation 3
Overview 3
What you will learn 3
What you need to complete this lab 3
Exercise 1 – XRMToolBox 4
Task 1: Running XrmToolBox 4
Exercise 2 – Add new Area, Group and Subarea(s) 7
Task 1: Create the Event Listing Entity 7
Task 2: Add Event Info Area 12
Task 3: Add Groups 15
Task 4: Add Subareas to the Events Info Group 18
Task 5: Add a Microsoft Events SubArea to the Public Sites Group 23
Task 6: Order Area and Add Title 24
Lab 02B – Working with Command Buttons 28
Overview 28
What you will learn 28
Working with Command Buttons 29
Task 1: Get and Import Ribbon Workbench 29
Task 2: Prepare Your Solution 31
Task 3: Hide Command Button 33
Task 4: Move Command Button 37
Task 5: Add Command Button 39
Lab 03 – Form Scripting 44
Overview 44
What you will learn 44
What you need to complete this lab 44
Exercise 1 – Add Fields and Vendor Section 45
Task 1: Edit the Account Form 45
Exercise 2 – Setup Solution and Connect to CRM 51
Task 1: Create a new CRM Package Project 51
Task 2: Register Form Event Handler 56

Task 3: Test Your work. 63
Lab 04 – Using the Organization Service 66
Overview 66
What you will learn 66
What you need to complete this lab 66
Exercise 1 – Connect to Dynamics 365 67
Task 1: Connect to Dynamics 365 67
Exercise 2 – Use Late Binding 75
Task 1: Create Record Using Late Bound 75
Exercise 3 – Using Early Bound Entities 79
Task 1: Generate Type Classes 79
Task 2: Create a Case Using Early Bound 81
Lab 05 – Querying Dynamics 365 Data 84
Overview 84
What you will learn 84
What you need to complete this lab 84
Exercise 1 – Query Expression 85
Task 1: Write a basic Query Expression 85
Exercise 2 – Fetch Expression 94
Task 1: FetchXML 94
Lab 06: Creating a Custom Workflow Activity 103
Lab 07: Creating a Plug-in 115

Lab 02A – Modifying Navigation

Overview

In this lab you will be modifying how users navigate a Dynamics 365 instance by customizing the SiteMap.

There are different ways that you can edit the SiteMap. You could manually edit the SiteMap XML to reflect the changes desired, but oftentimes a 3rd party tool can make that process much easier. To demonstrate this process, you will be using a community tool to assist you in modifying the SiteMap XML.

What you will learn

- How to modify CRM Navigation
- How to add a new Area to CRM
- How to add new Groups to an Area
- How to add new Sub Areas
- How to add a link to an external site to the navigation

What you need to complete this lab

- Access to the XRMToolbox community tool.

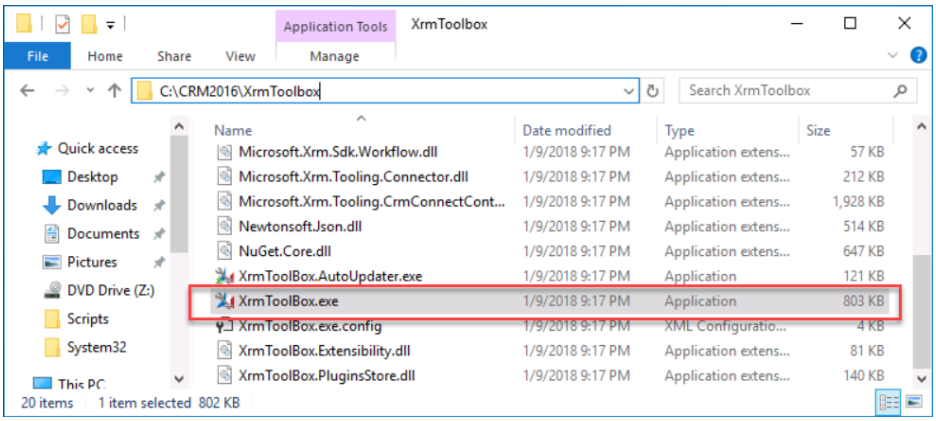
Exercise 1 – XRMToolBox

In this exercise, you will run a community-built tool called XRMToolBox that will help edit the SiteMap.

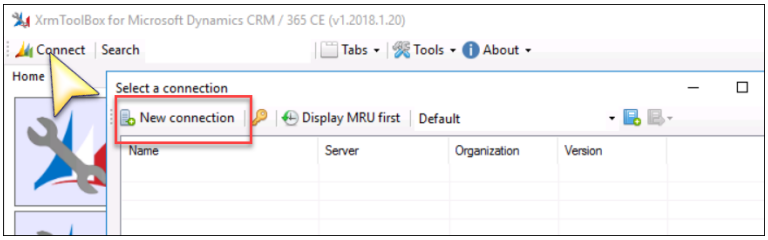
Task 1: Running XrmToolBox

In this task you will run XRMtoolbox.

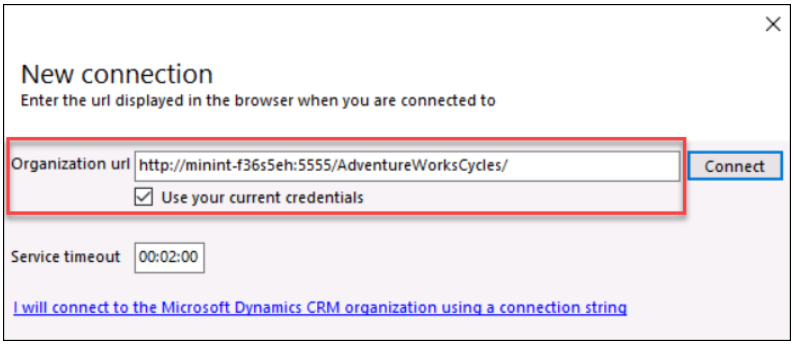
1) In the **C:\CRM2016\XrmToolbox** folder, locate and double click the **XrmToolBox.exe** application file.



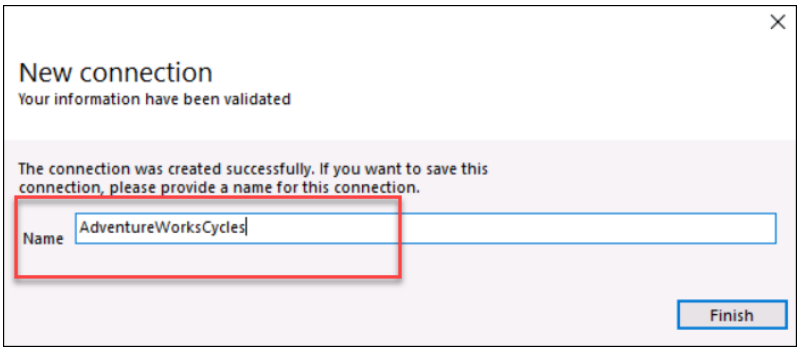
2) Click **Connect** and click **New connection**.



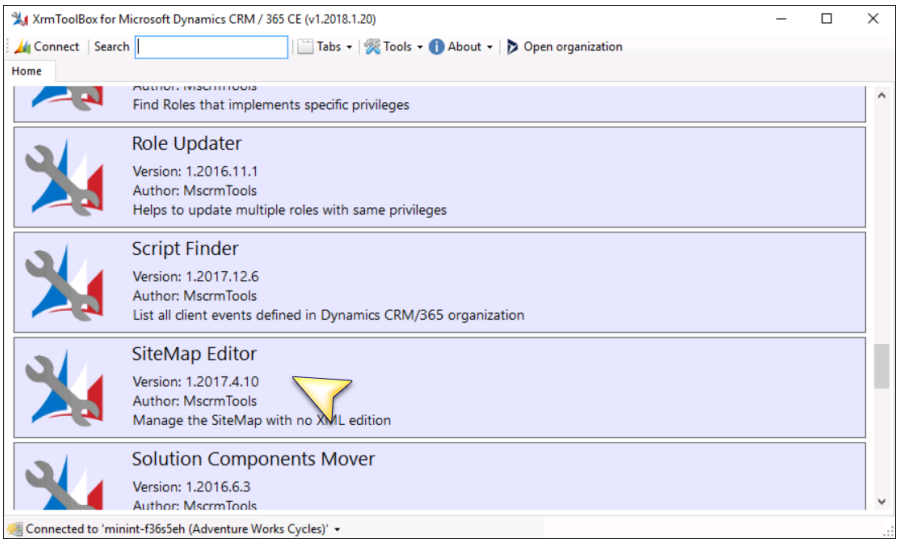
3) Enter **http://minint-f36s5eh:5555/AdventureWorksCycles** in the **Organization url** textbox.
4) Select the **Use your current credential** checkbox.



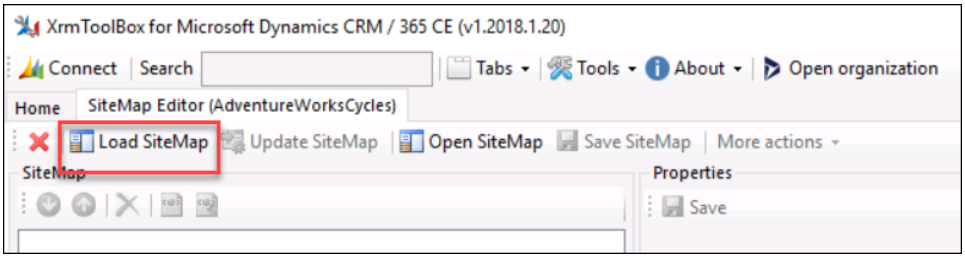
5) Click **Connect**.
6) In the **New connection** dialog, type **AdventureWorksCycles** in the **Name** textbox and click **Finish**.



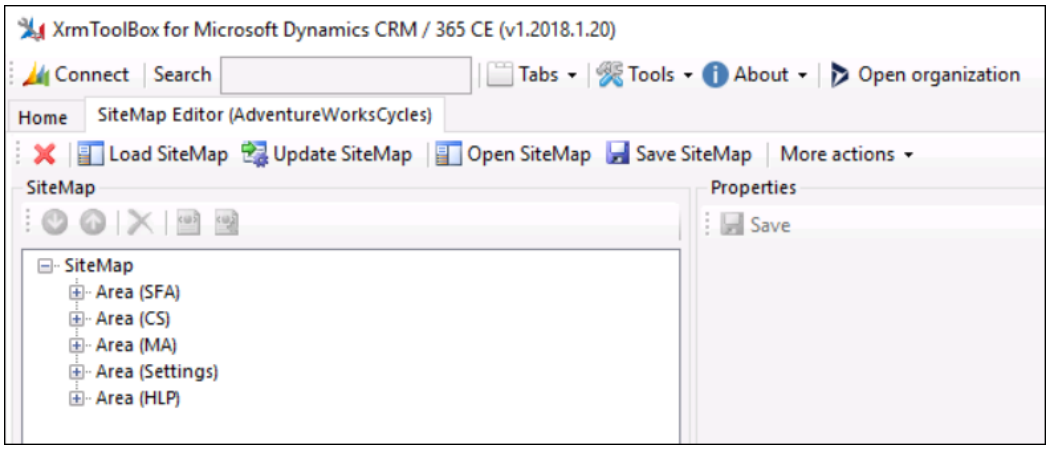
7) After you connect to your CRM organization locate and click SiteMap Editor.



8) Click Load SiteMap.



9) The tool will now load your existing SiteMap.


XrmToolBox for Microsoft Dynamics CRM / 365 CE (v1.2018.1.20)
Connect
Search
Tabs
Tools
About
Open organization
Home
SiteMap Editor (AdventureWorksCycles)
Load SiteMap
Update SiteMap
Open SiteMap
Save SiteMap
More actions
SiteMap
Properties
Save
SiteMap
Area (SFA)
Area (CS)
Area (MA)
Area (Settings)
Area (HLP)

Exercise 2 – Add new Area, Group and Subarea(s)

In this exercise you will be customizing CRM navigation by creating a new Area labeled Events. From this Area you should be able to view and create Event Listings and Accounts. You should also be able to access the external Microsoft Events page from your deployment.

Once completed, you would like to the Events area to reflect the structure that is outlined below:

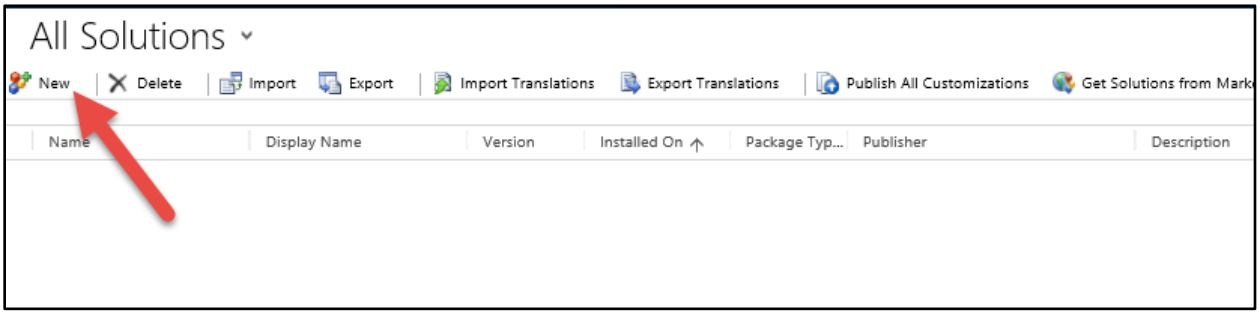
Site Map

- Sales
- Events *(New Area)*
    - *Event Info (New Group)*
        - Accounts
        - Event Listing (New Custom Entity)
    - Public Sites *(New Group)*
        - Microsoft Events (Link to https://msevents.microsoft.com)
- Service
- Marketing

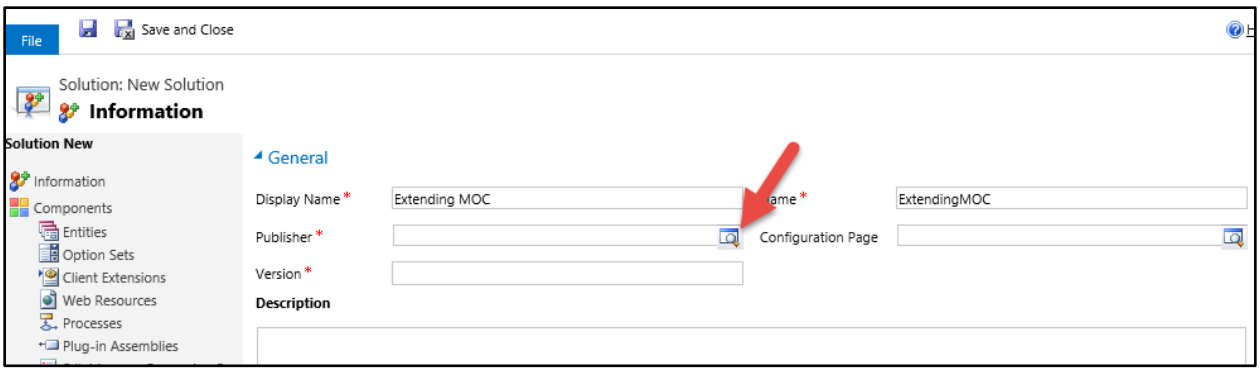
Task 1: Create the Event Listing Entity

In this task you will create an entity to represent Event Listings. Because we are not using this for anything other than adding it to navigation we will not be adding any custom fields to it.

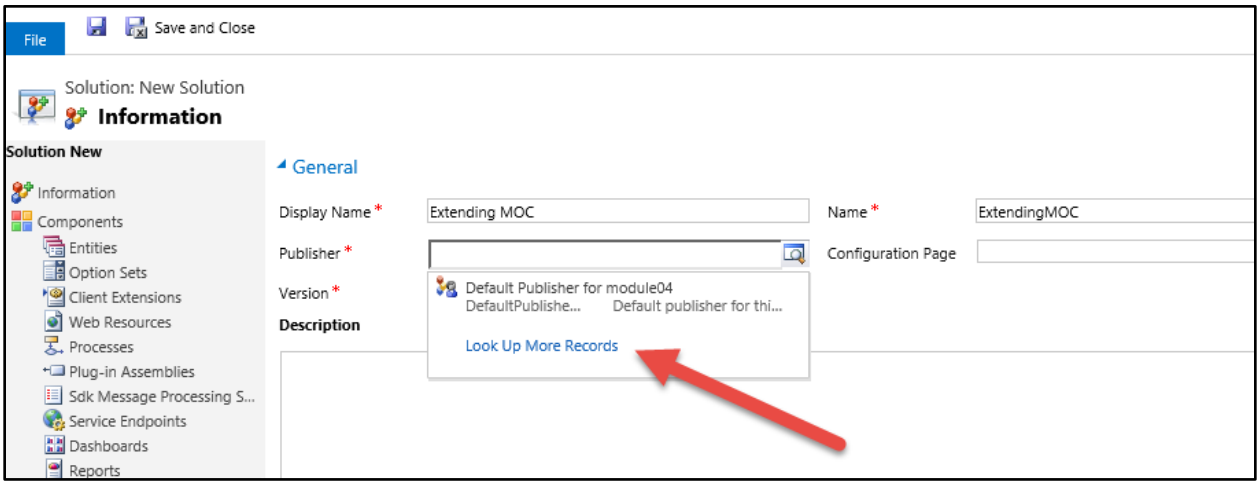
1) Go to your Dynamics 365 Organization, navigate to Settings | Solutions and click New.



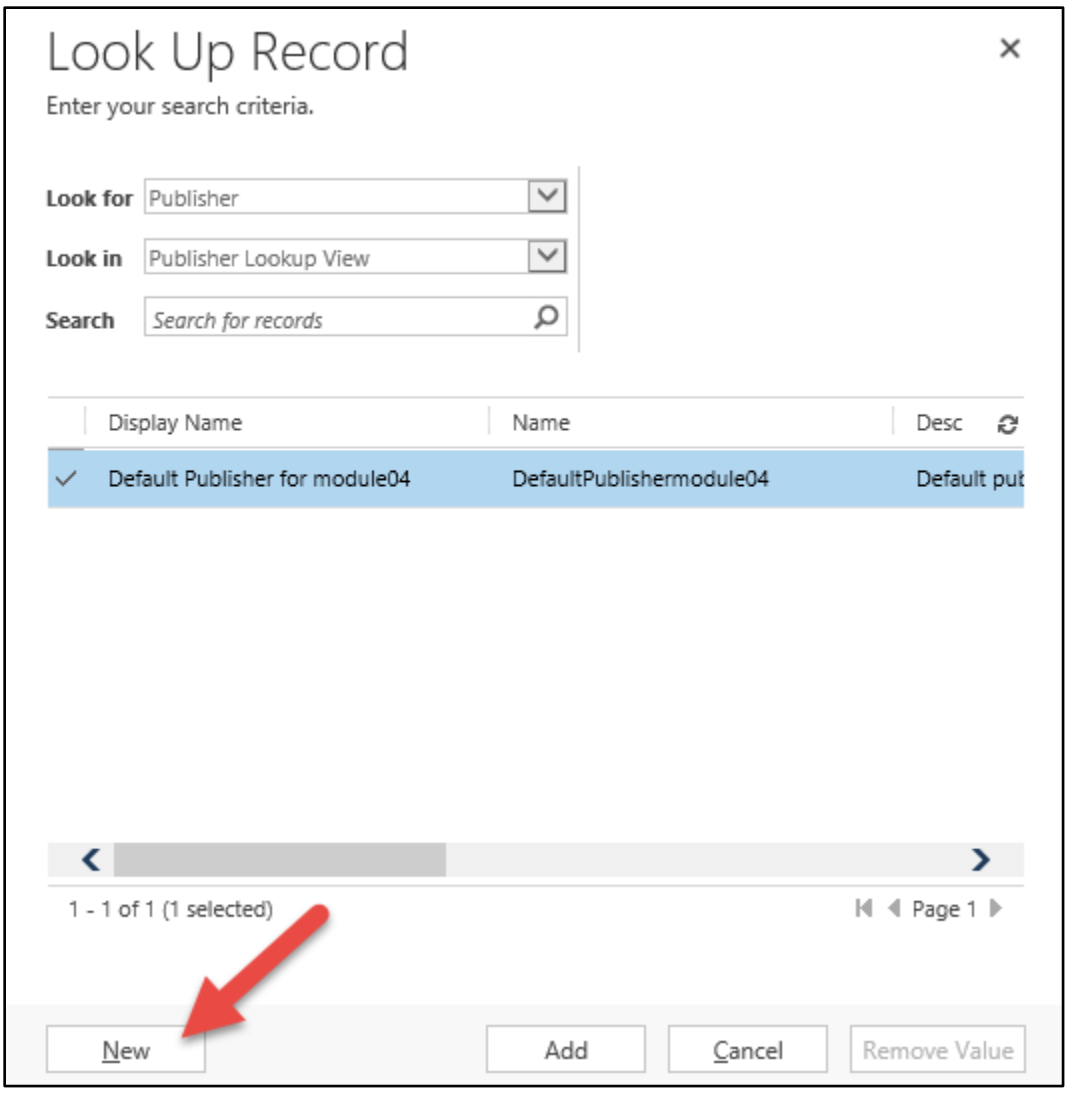
2) Enter **Extending MOC** for **Display name** and click the **Publisher** lookup.



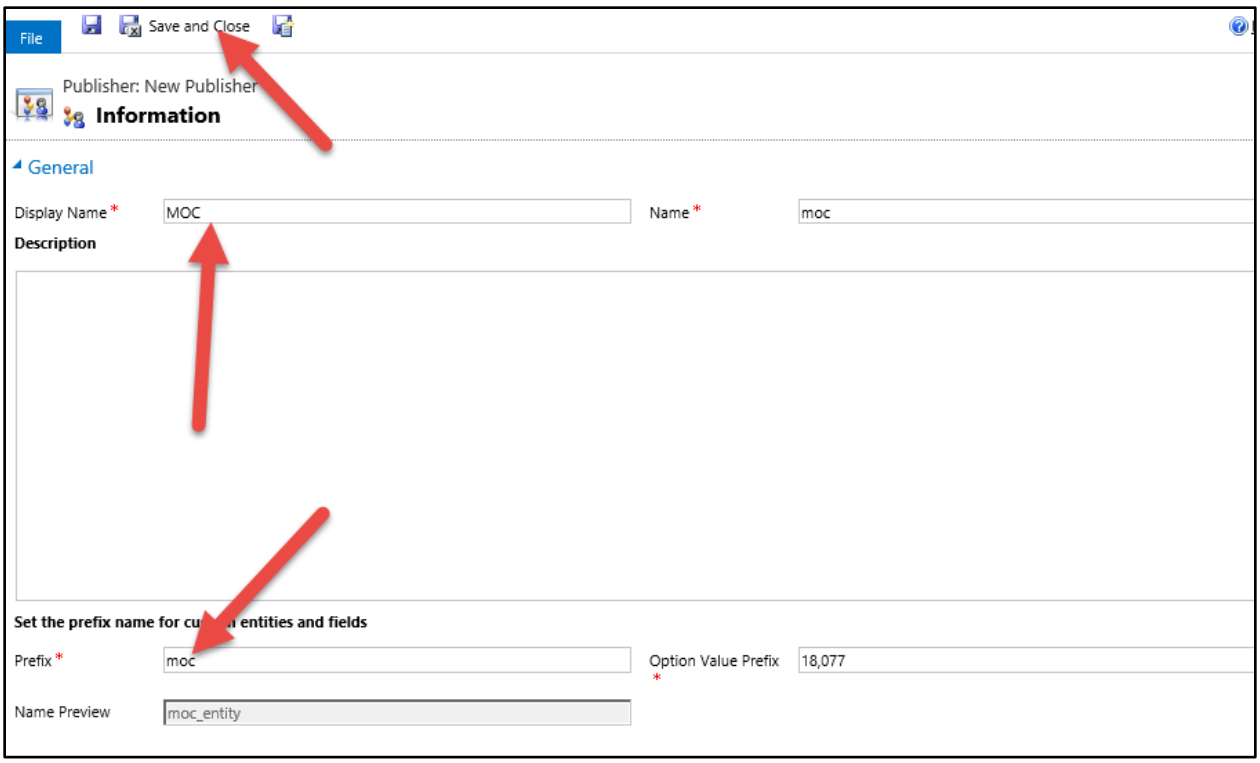
3) Click **Look Up More Records**.



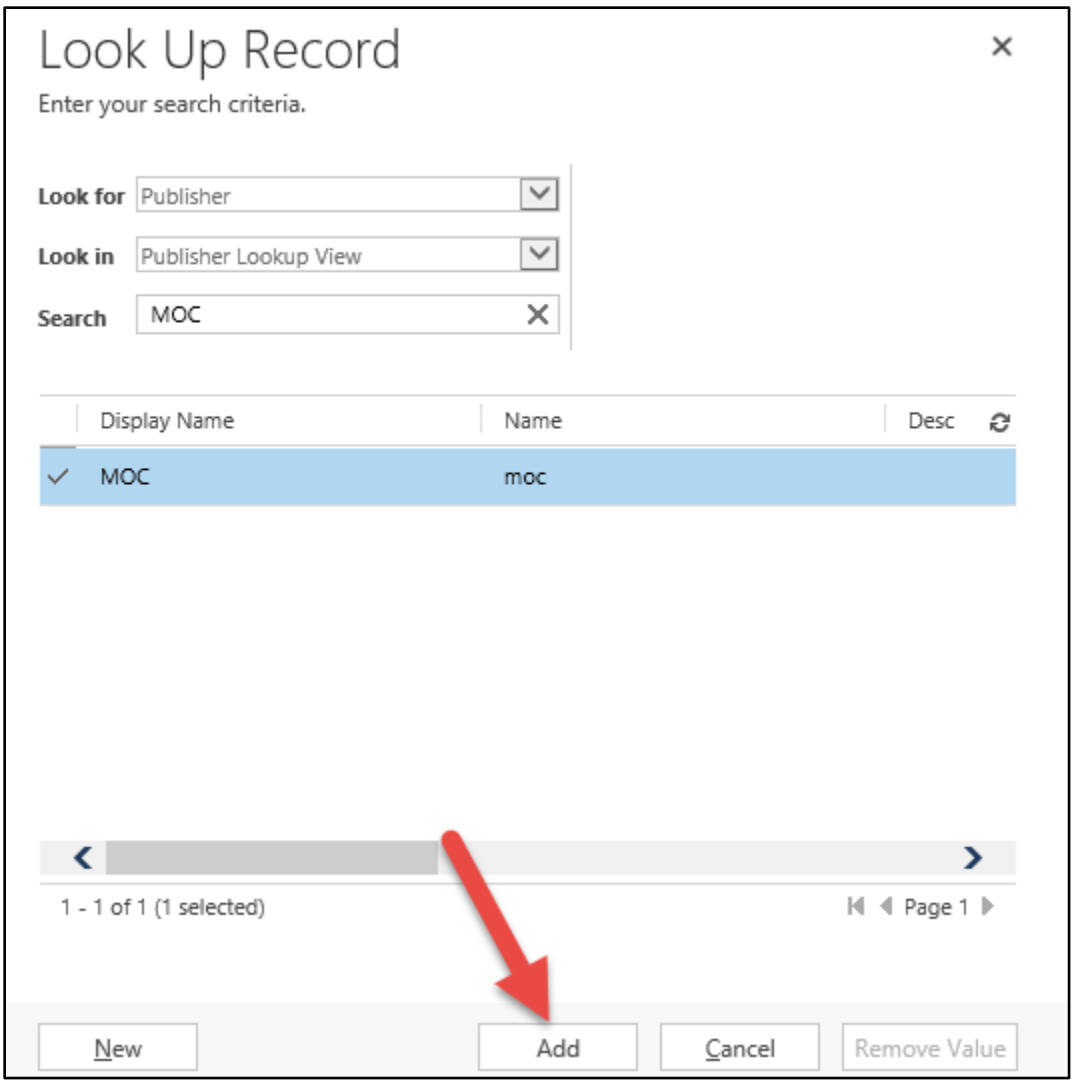
4) Click **New**.



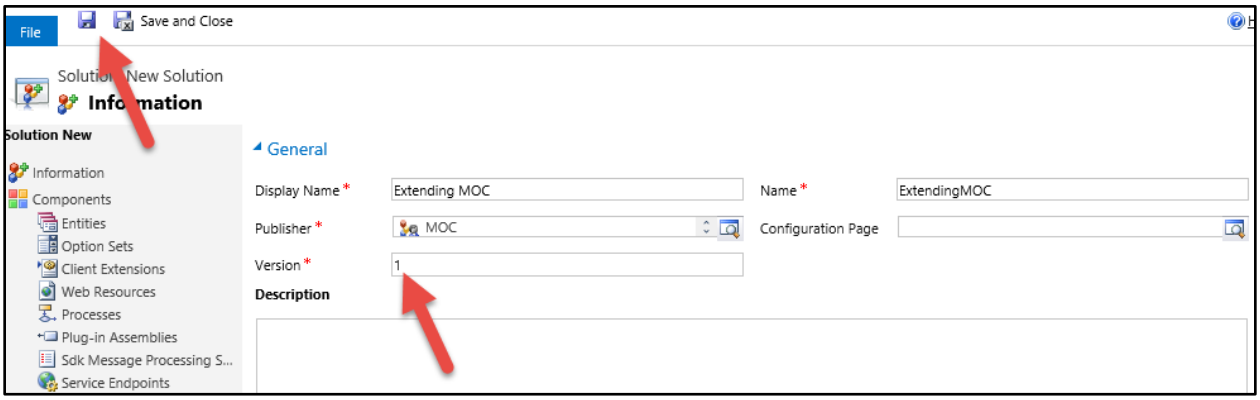
5) Enter **MOC** for **Display Name**, **moc** for **Prefix** and click **Save and Close**.


File
Save and Close
Publisher: New Publisher
Information
General
Display Name *
MOC
Name *
moc
Description
Set the prefix name for cu entities and fields
Prefix *
moc
Option Value Prefix *
18,077
Name Preview
moc_entity

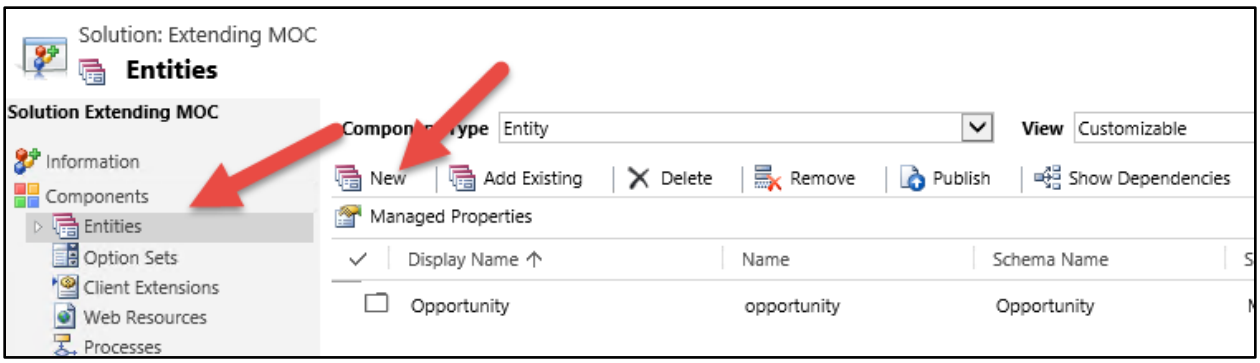
6) Click **Add**.



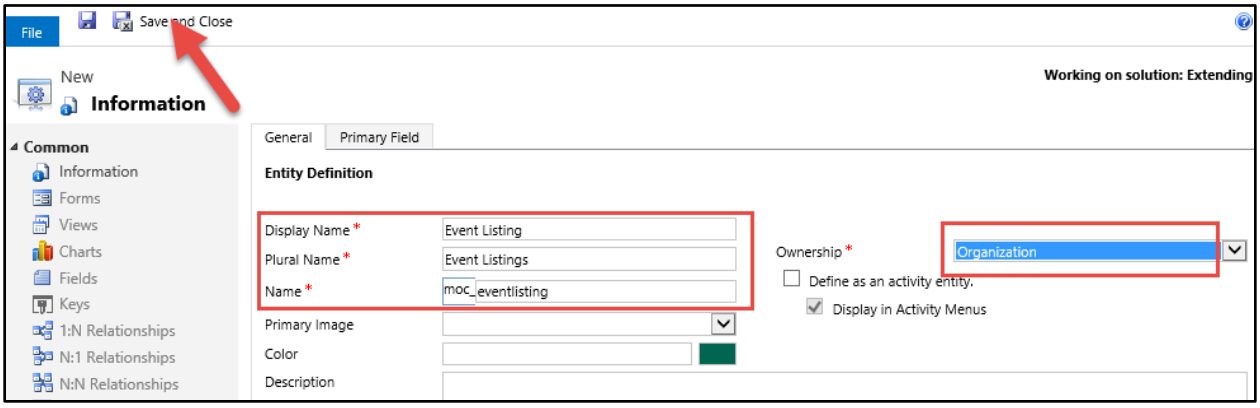
7) Enter **1** for **version** and click **Save**.



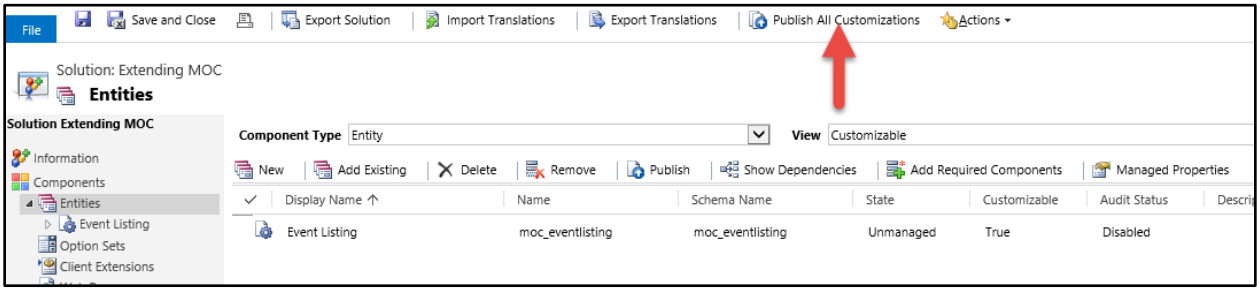
8) Select **Entities** and click **New**.



9) Enter **Event Listing** for **Display Name**, **Event Listings** for **Plural Name**, **moc_eventlisting** for **Name**, select **Organization** for **Ownership**, and click **Save and Close**. Leave other items, including data type, to the defaults already defined.



10) Click **Publish All Customizations**.

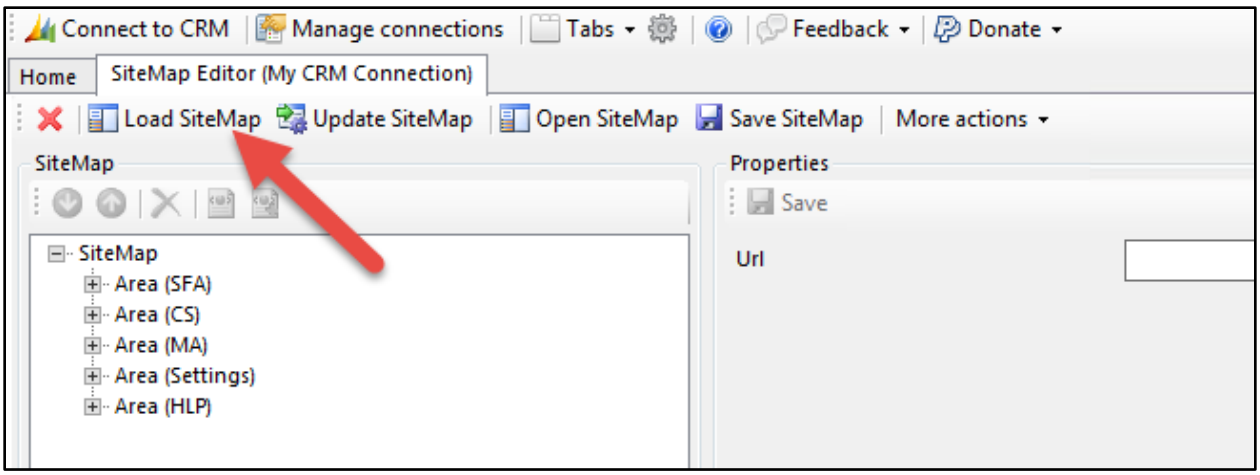


11) Close the Solution Explorer. As a reminder, we are only making enough of this entity to learn about the site map, there is not a need at this time to build out more of the entity details.

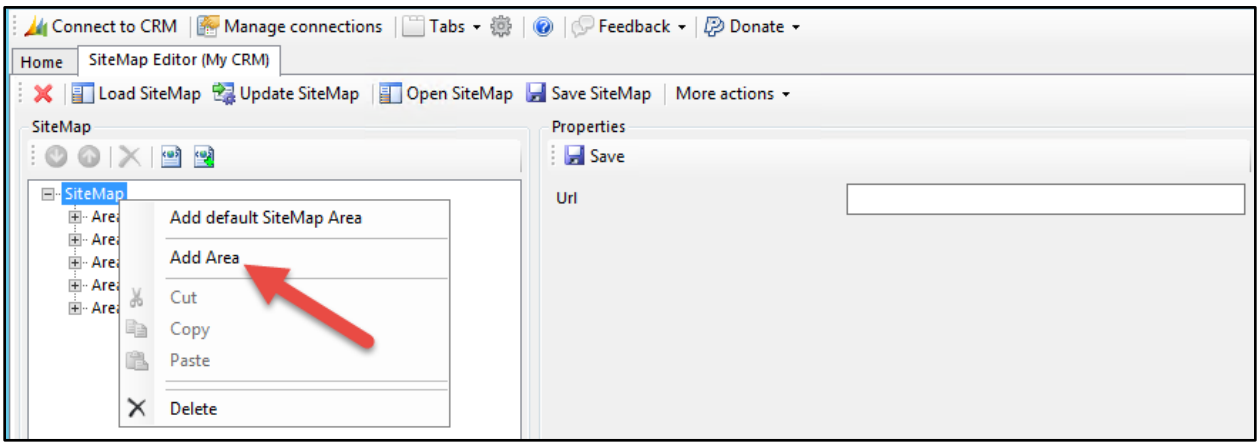
Task 2: Add Event Info Area

In this task you will use the SiteMap Editor to add a new SiteMap Area called Events to Dynamics 365.

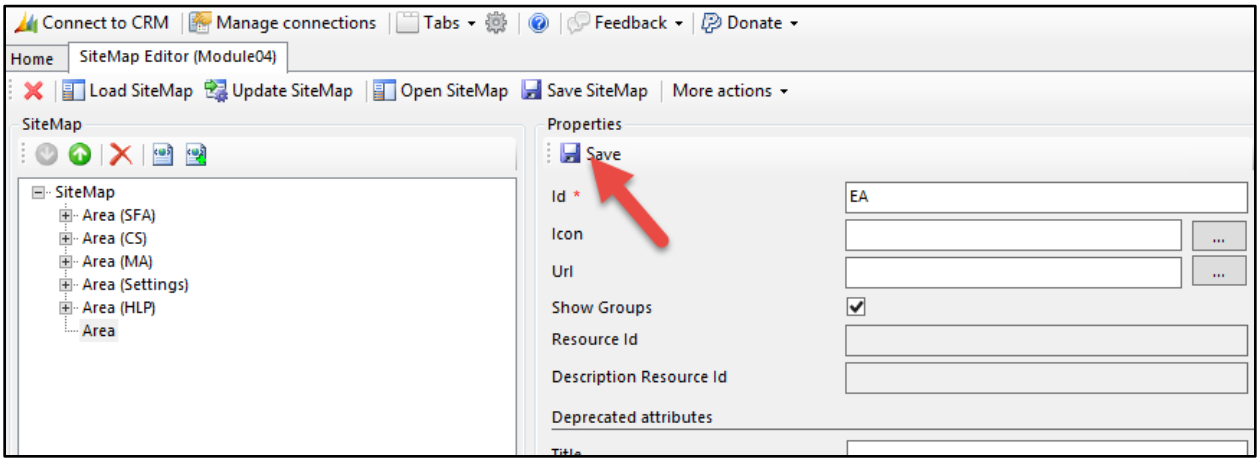
1) Go to the XrmToolBox, and click **Load SiteMap**. (This action will refresh XrmToolBox.)



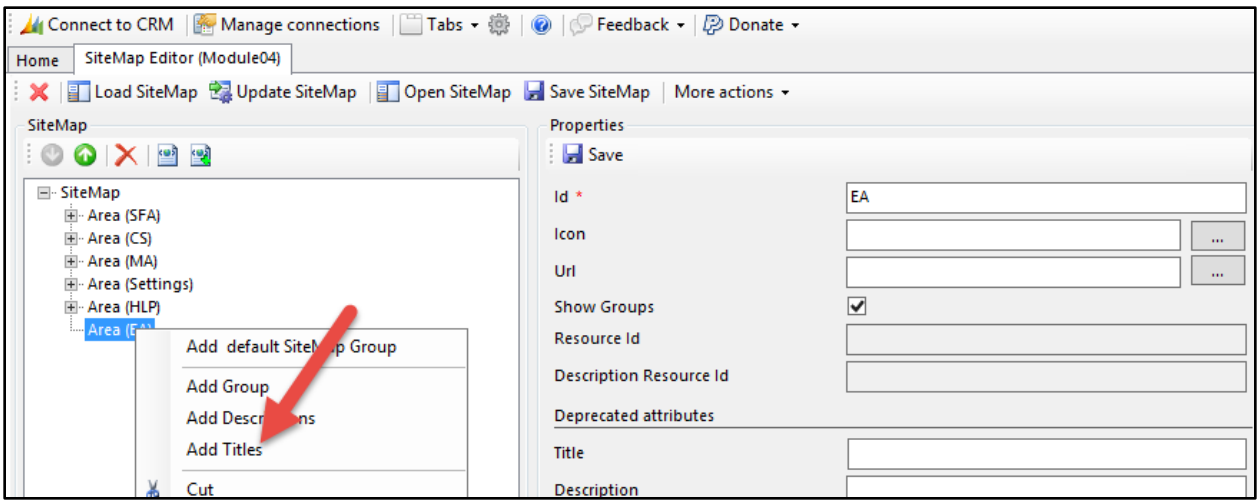
2) Right-click **SiteMap** and click **Add Area**.



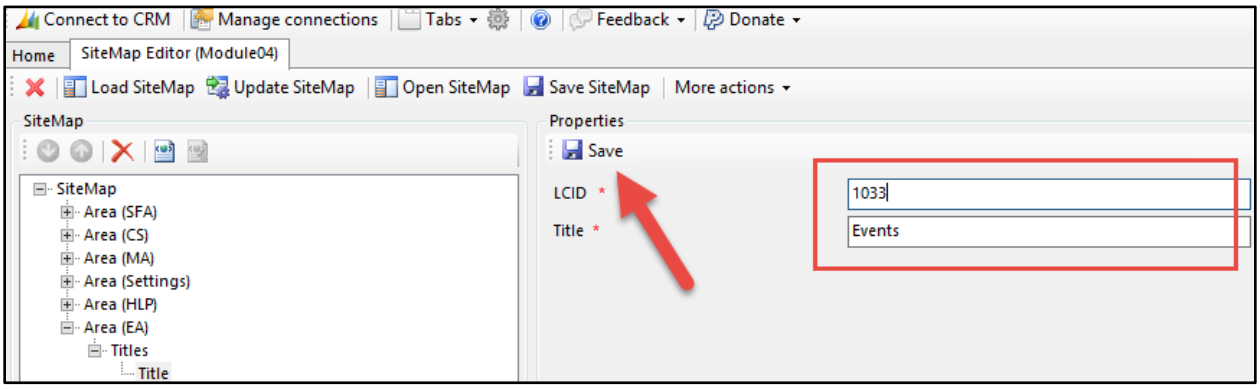
3) Enter **EA** for **Id**, check **Show Groups** checkbox, and click **Save**.



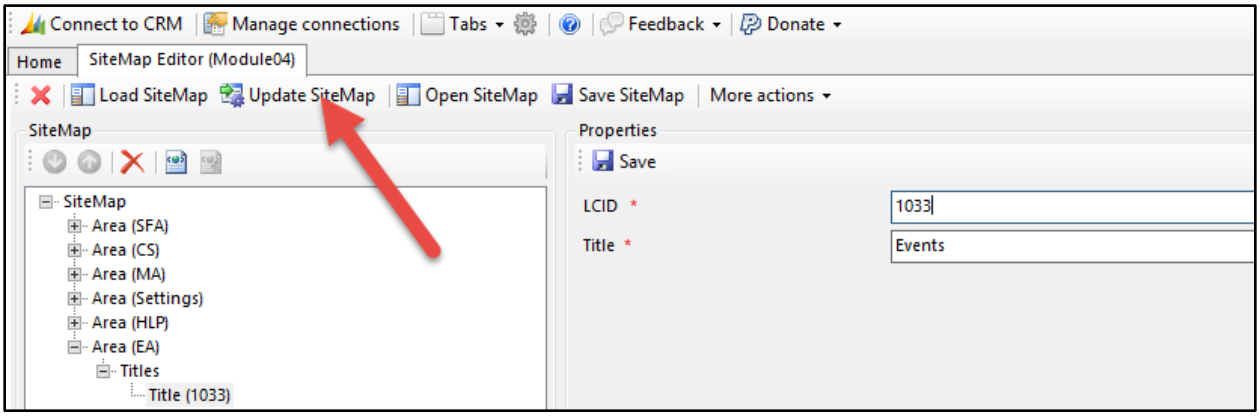
4) Right click Area (EA) and click **Add Titles**.



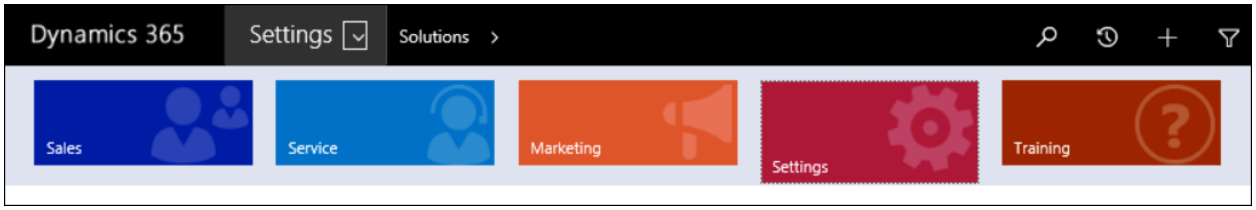
5) Enter **1033** for **LCID** (because we are only using English for this example), **Events** for **Title**, and click **Save**.



6) Click **Update SiteMap** and wait for update to complete.



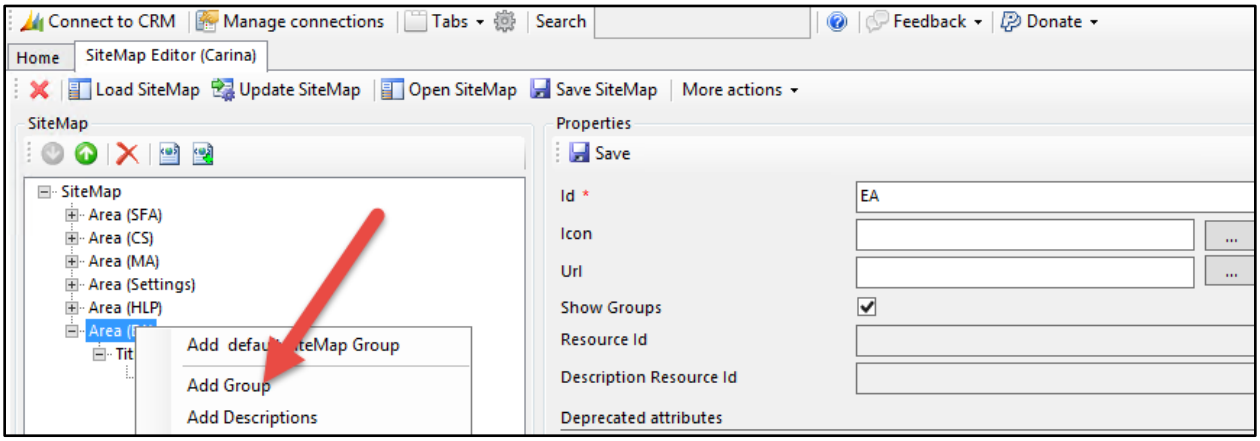
7) Go to your Dynamics 365 organization. Refresh the page, and click on the menu button. Notice that the Dynamics 365 SiteMap will not show any the new Events Area yet because it does not have Groups and SubAreas defined.



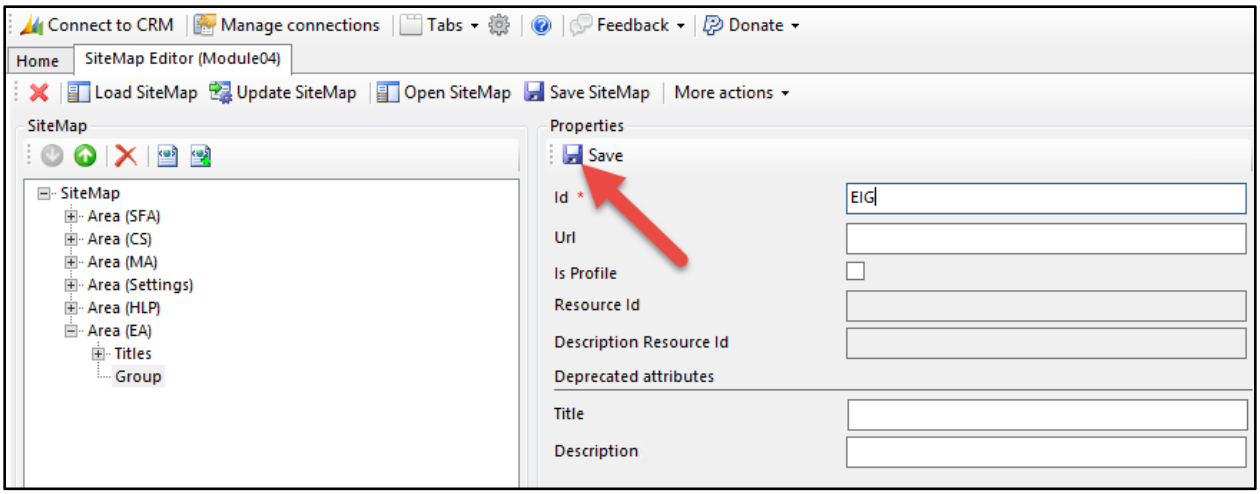
Task 3: Add Groups

In this task you will add two new Groups called Event Info and Public Sites to the Events Area.

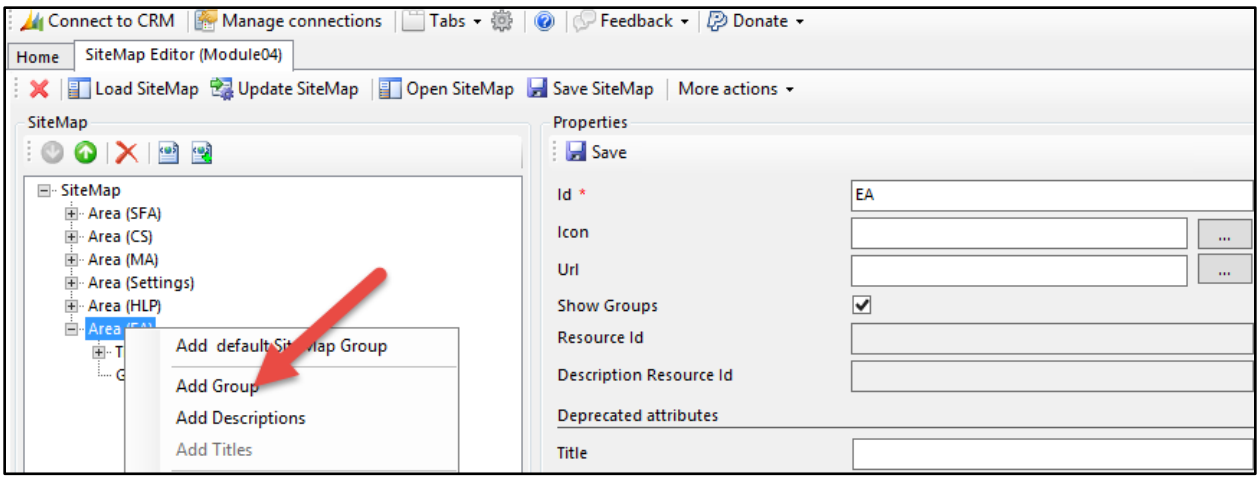
1) Go back to the XrmToolBox, right click the **EA** area, and click **Add Group**.



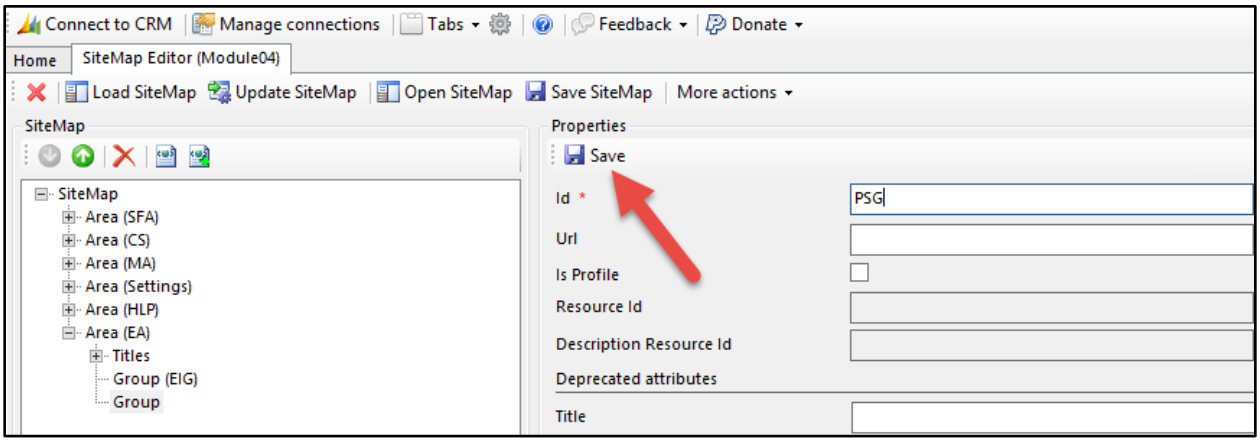
2) Enter **EIG** for **Id** and click **Save**.



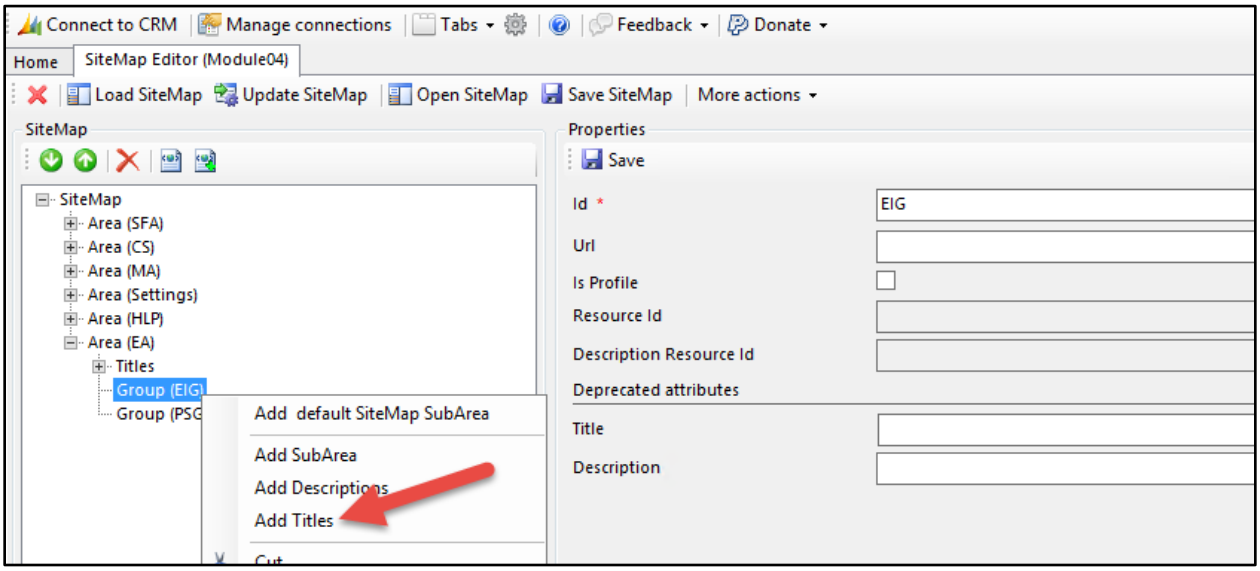
3) Right click on **EA** area and click **Add Group**.



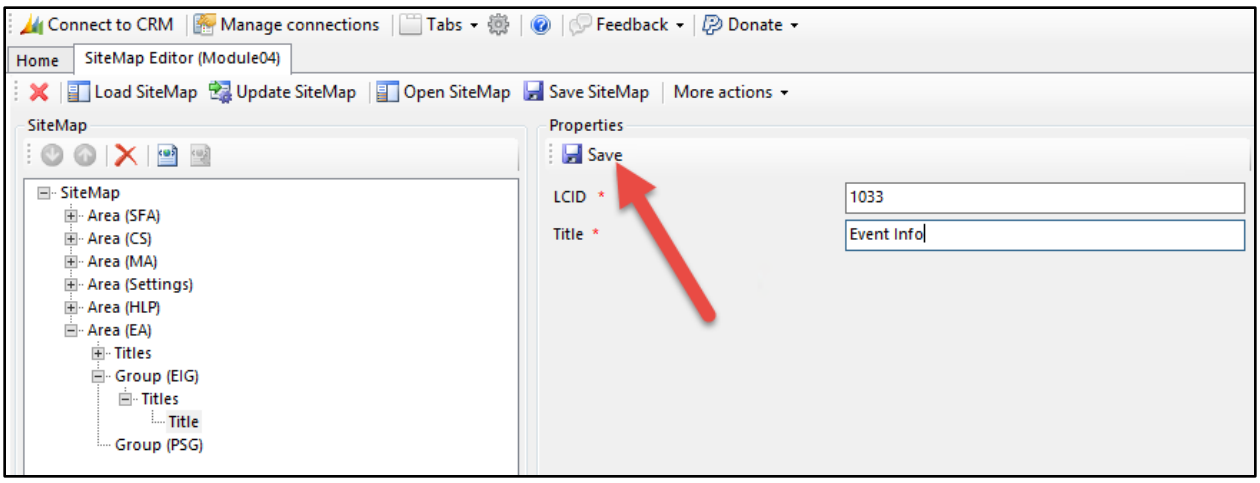
4) Enter **PSG** for **Id** and click **Save**.



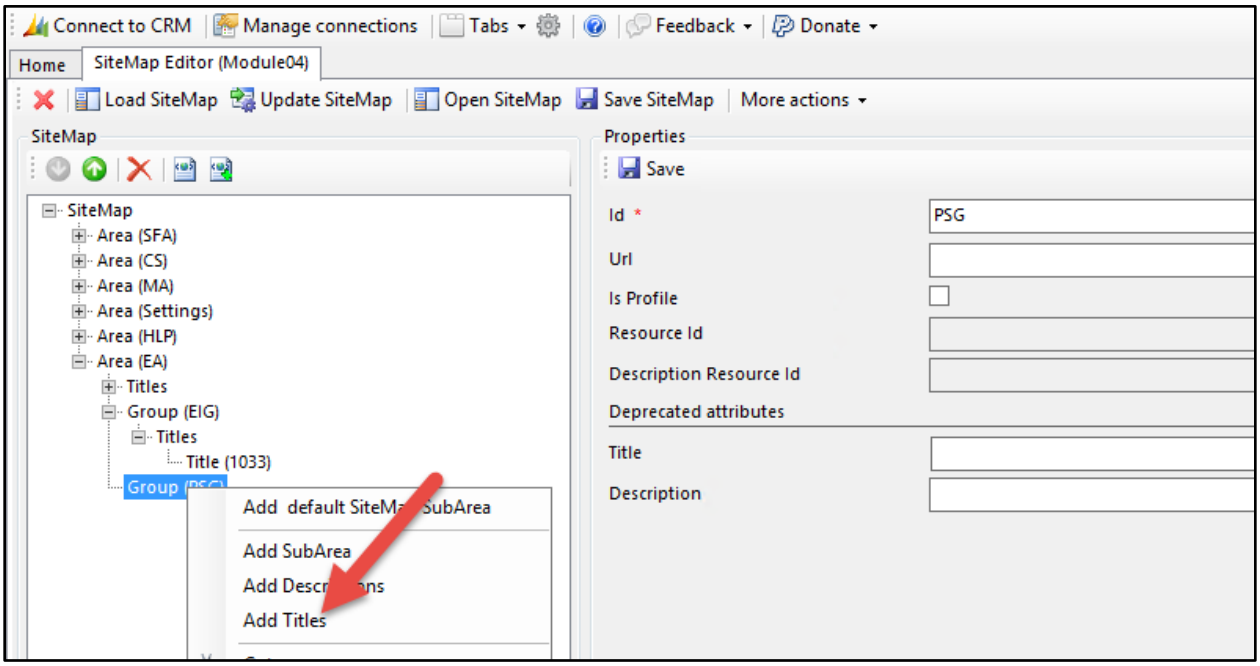
5) Right click on **EIG** and click **Add Titles**.



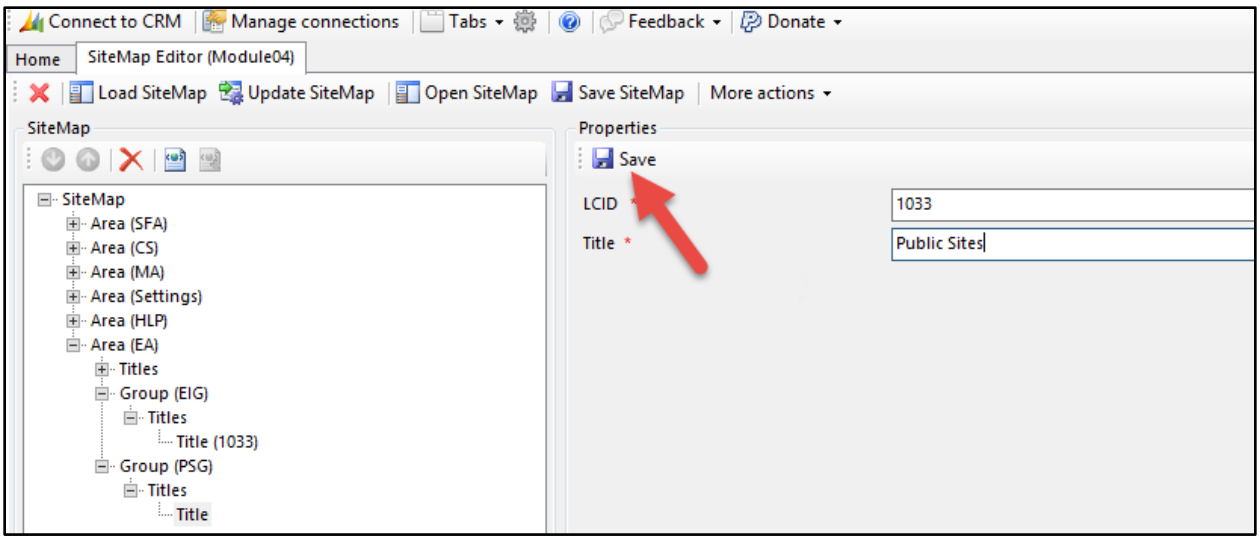
6) Enter **1033** for **LCID**, **Event Info** for **Title** and click **Save**.



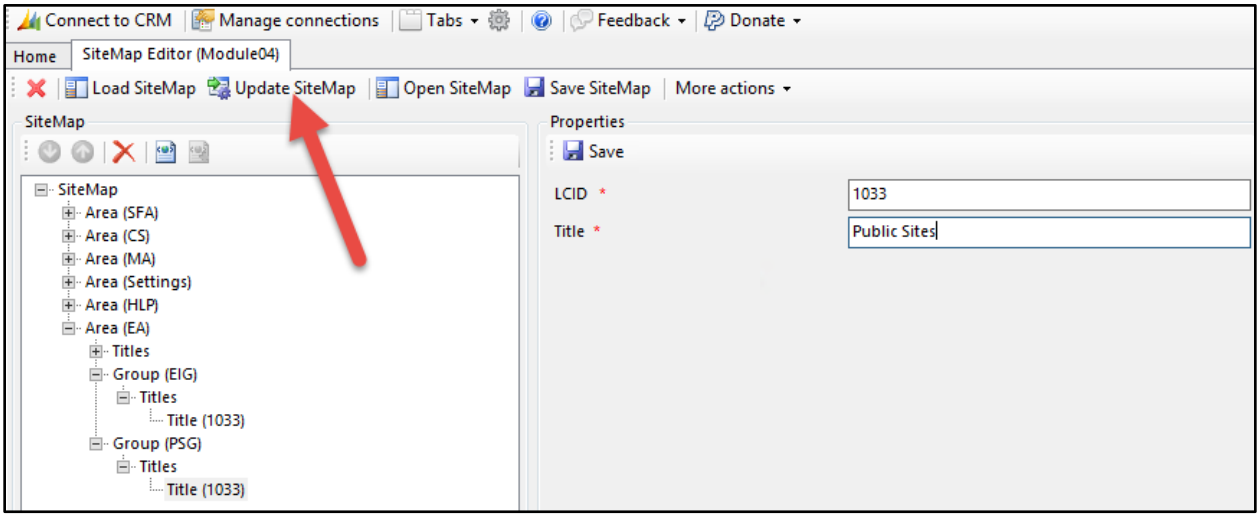
7) Right click on **PSG** and click **Add Titles**.



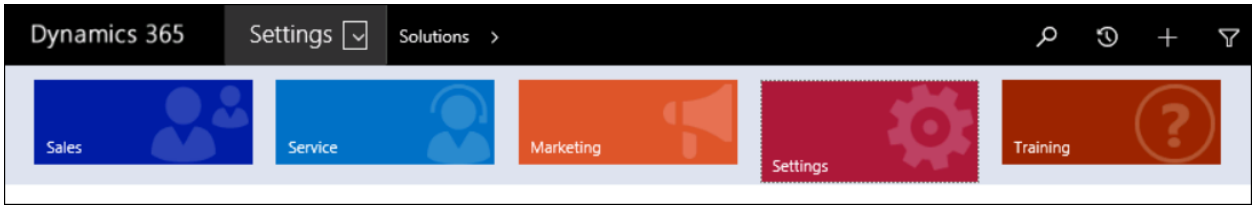
8) Enter **1033** for **LCID**, **Public Sites** for **Title**, and click **Save**.



9) Click **Update SiteMap** and wait for update to complete.



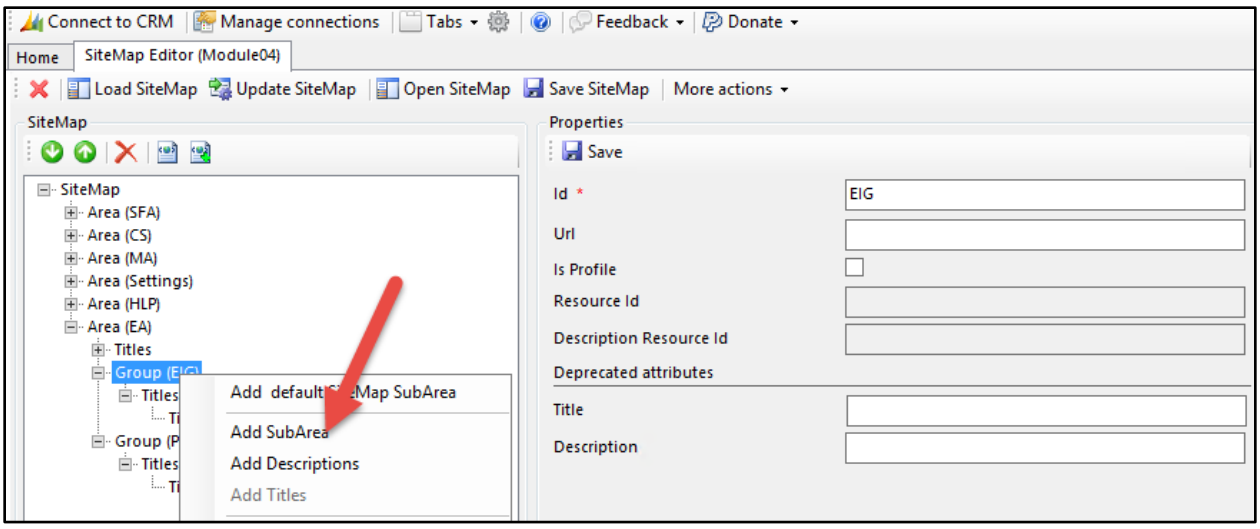
10) Go back into your Dynamics 365 organization. Refresh the page and click on the menu button. The Dynamics 365 SiteMap still does not display the Events Area because none of the Groups defined contain any SubAreas.



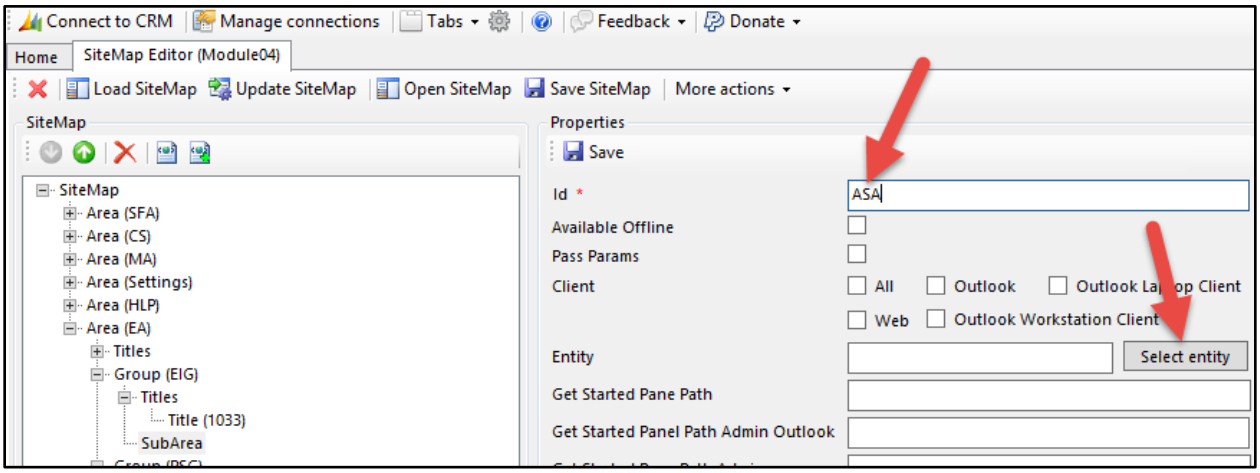
Task 4: Add Subareas to the Events Info Group

In this task, you add two SubAreas called Accounts and Event Listings to the Event Info group.

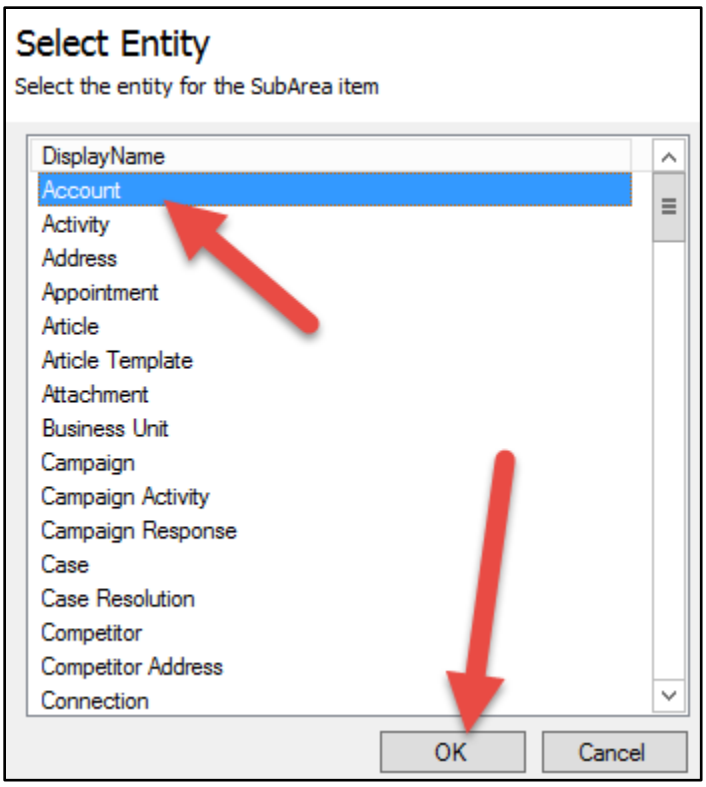
1) Go to XrmToolBox, right click the **EIG** group, and click **Add SubArea**.



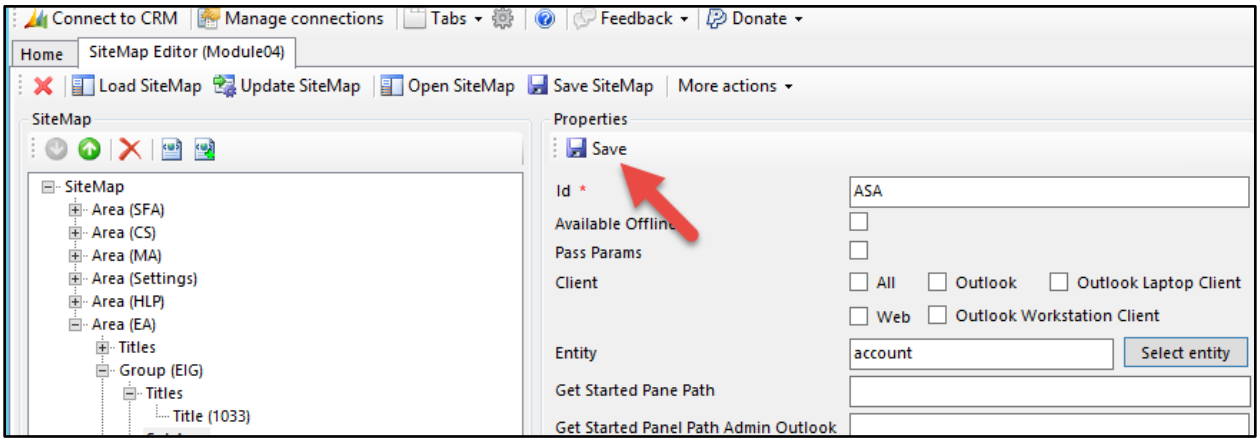
2) Enter **ASA** for **Id**, and click **Select Entity**.



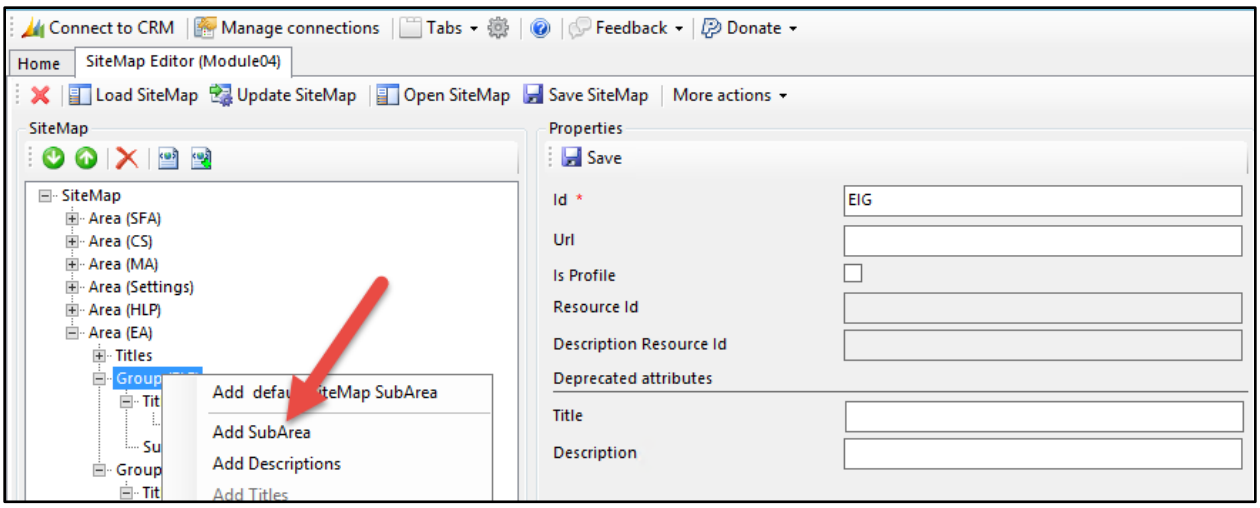
3) Select **Account** and click **OK**.



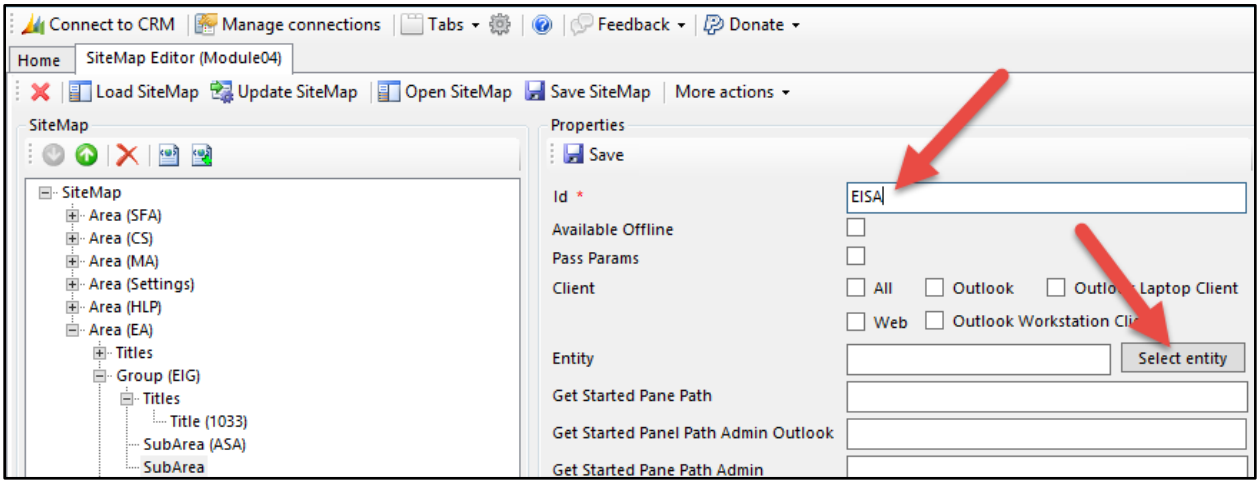
4) Click **Save**.



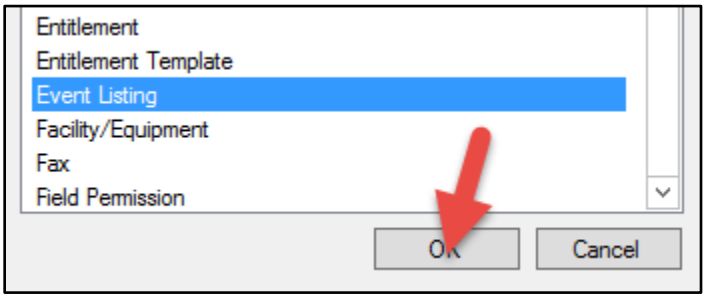
5) Right click on **EIG** group and click **Add SubArea**.



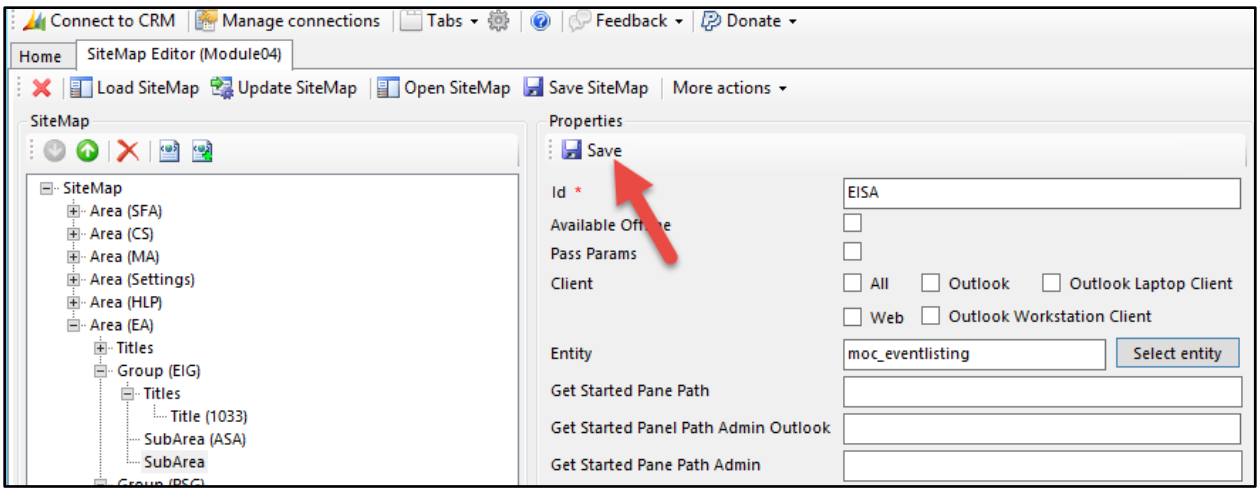
6) Enter **EISA** for **Id** and click **Select Entity**.



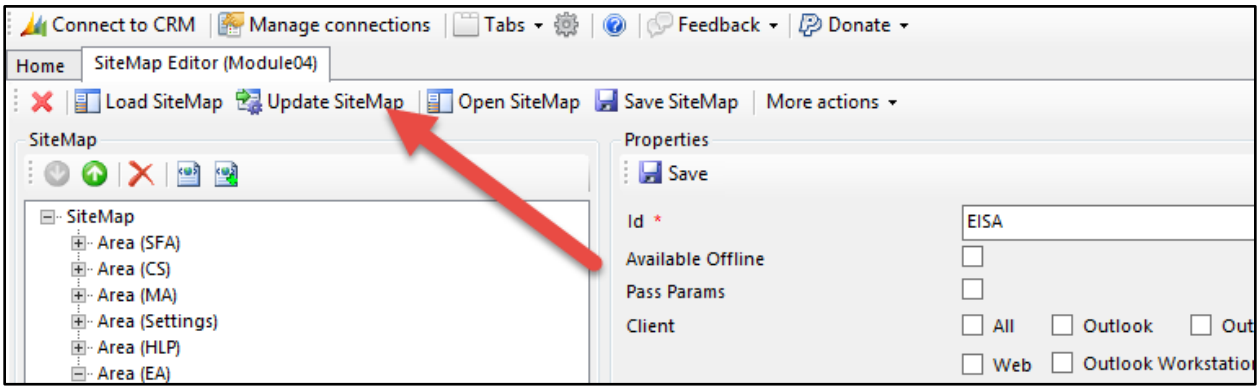
7) Select **Event Listing** and click **OK**.



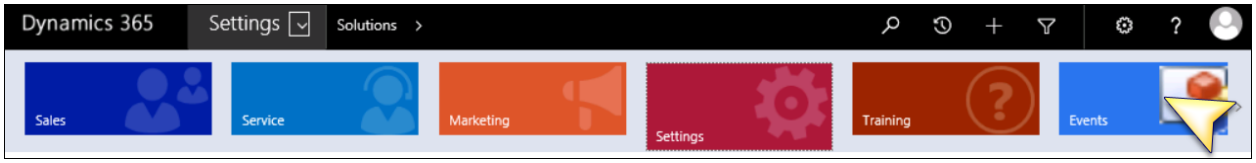
8) Click **Save**.



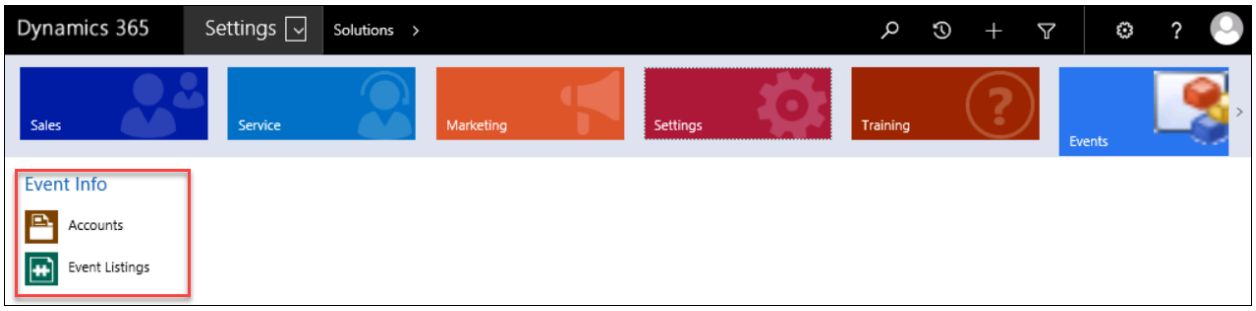
9) Click **Update SiteMap** and wait for update to complete.



10) In your Dynamics 365 organization, refresh the page and click on the menu button. Dynamics 365 SiteMap will now display the Events Area because at least one Group has a SubArea defined.



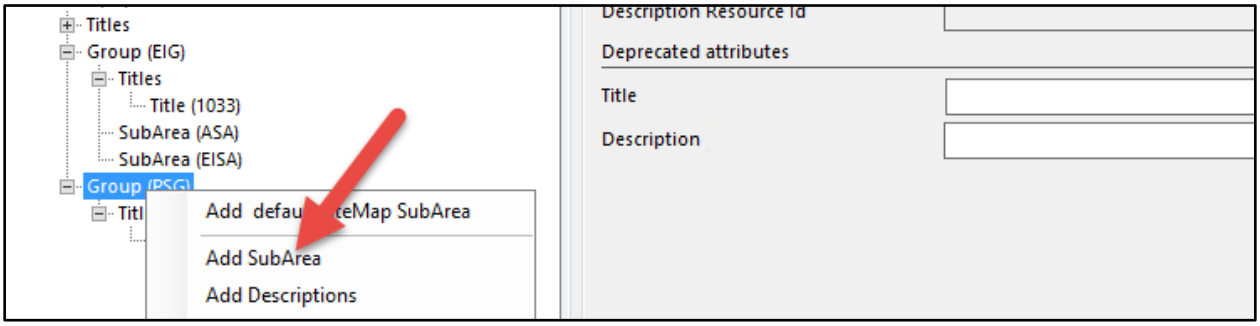
11) Select Events. Notice that only the Event Info Group (with Accounts and Event Listings SubAreas) is displayed because it is the only group that contains at least one defined SubArea.



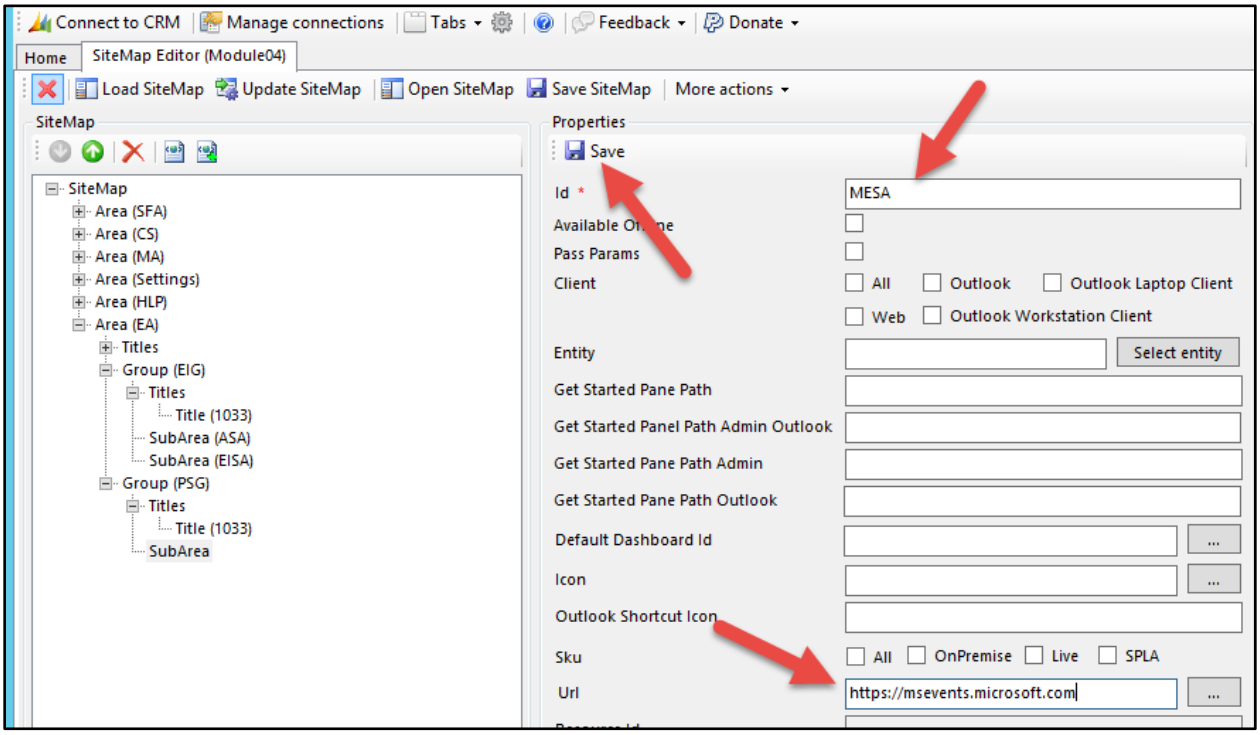
Task 5: Add a Microsoft Events SubArea to the Public Sites Group

In this task you will add SubArea to the Public Sites Group.

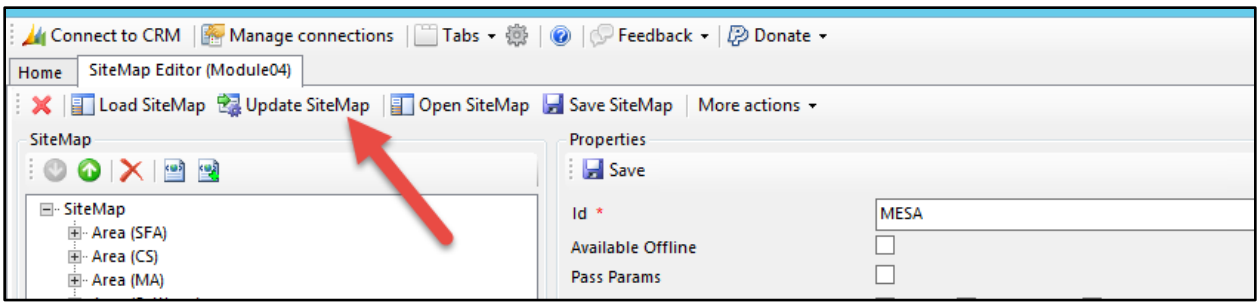
1) Right click on the **PSG** group and click **Add SubArea**.



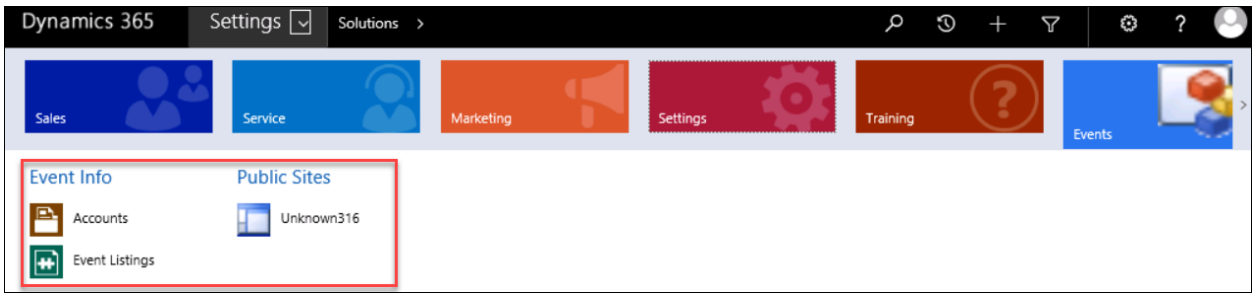
2) Enter **MESA** for Id, enter https://msevents.microsoft.com for **Url** and click **Save**.



3) Click **Update SiteMap** and wait for update to complete.



4) Back in your Dynamics 365 organization refresh the page, click on the menu button, and click Events. Both the Event Info and Public Sites groups are listed with as well as the defined SubAreas for each on.

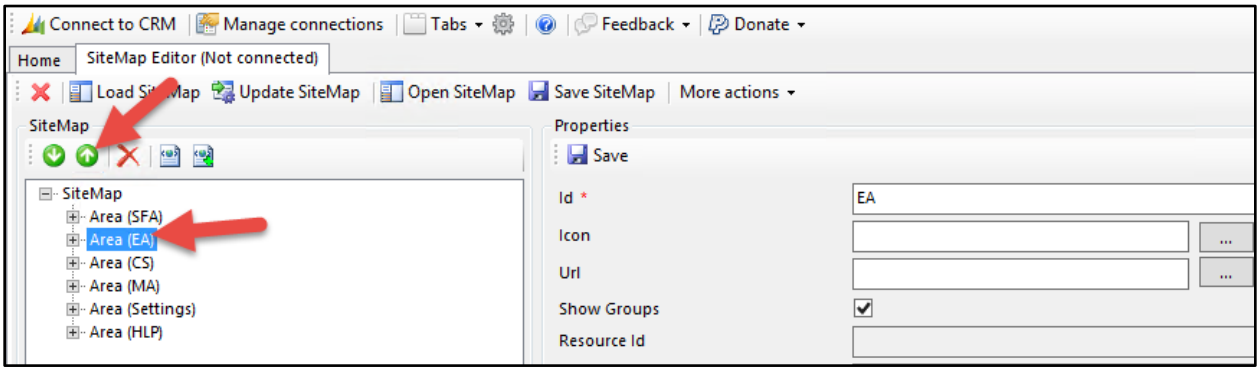


5) Notice that the auto generated title for the Microsoft Events SubArea is not user friendly, we will take care of that in the next Task.

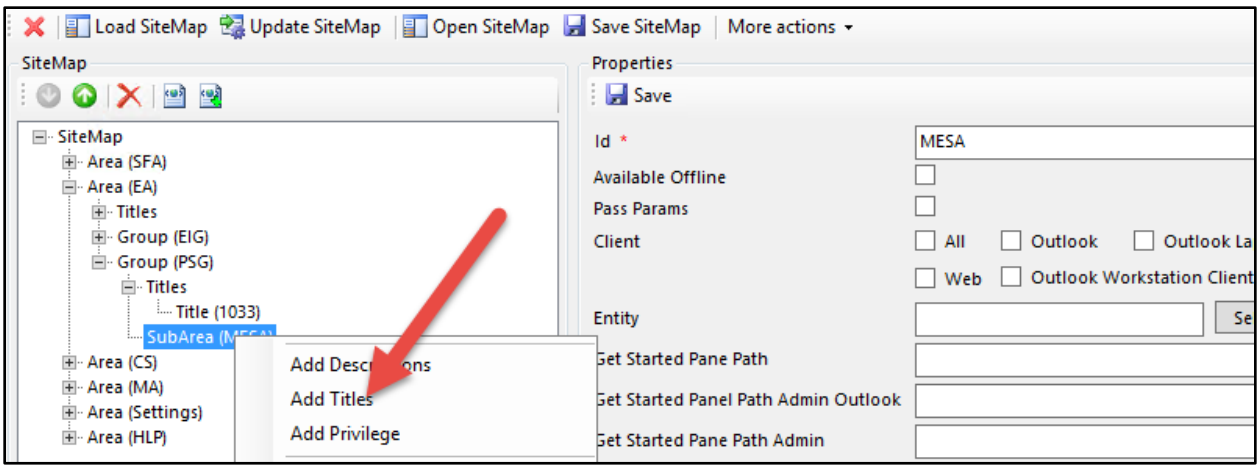
Task 6: Order Area and Add Title

In this task you add reorder SiteMap Areas and add Title to the external link.

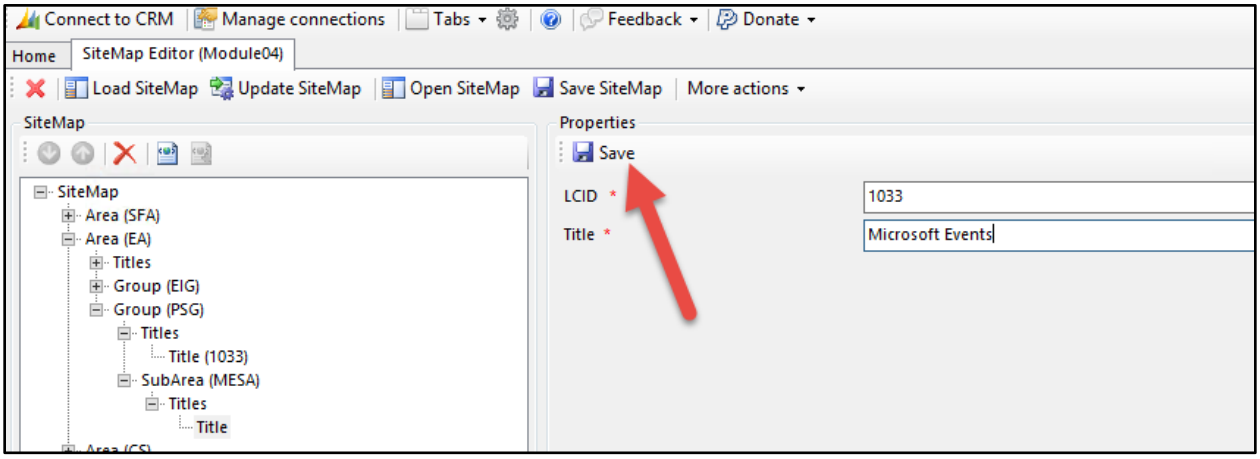
1) Go to the XrmToolBox, select the **EA** area. Using the arrow keys move the EA area right below the **SFA** area.



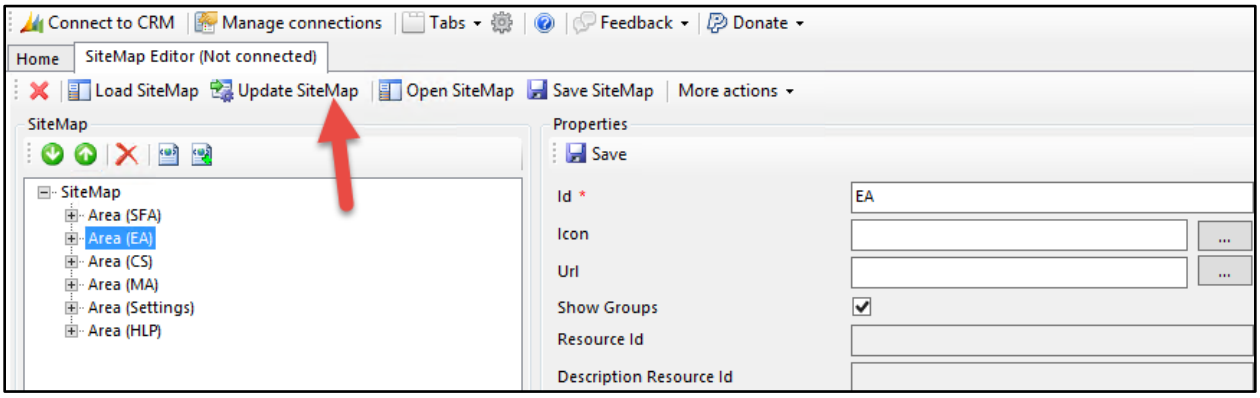
2) Expand **EA** area, expand **PSG** group, right click on **MESA** sub-area, and click **Add Titles**.



3) Enter **1033** for **LCID**, Enter **Microsoft Events** for the **Title**, and click **Save**.



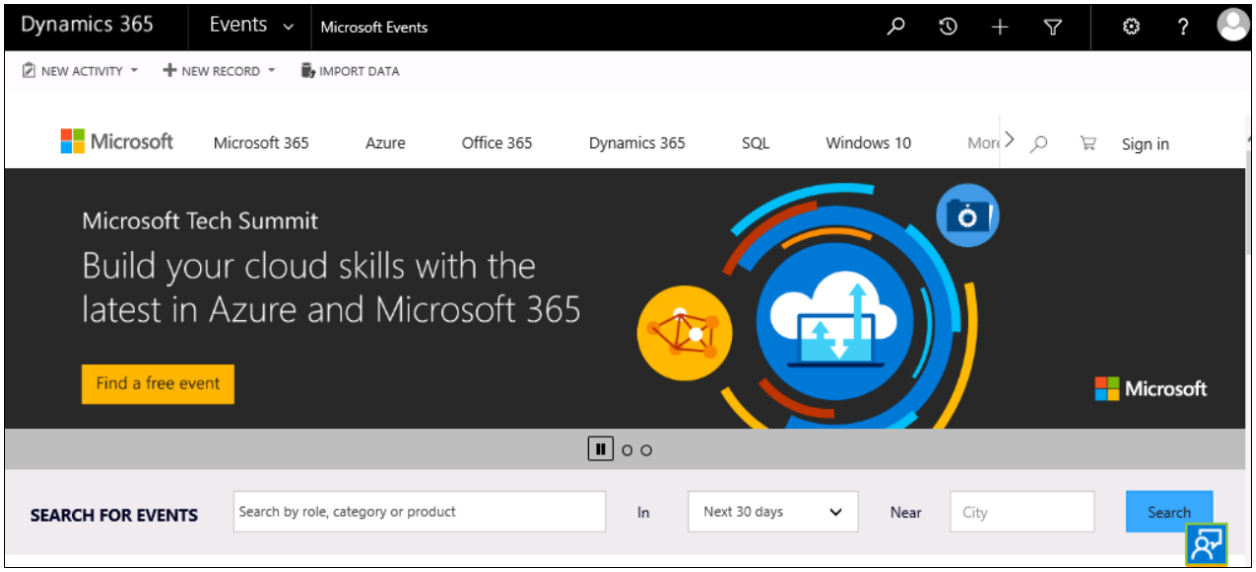
4) Click **Update SiteMap** and wait for update to complete.



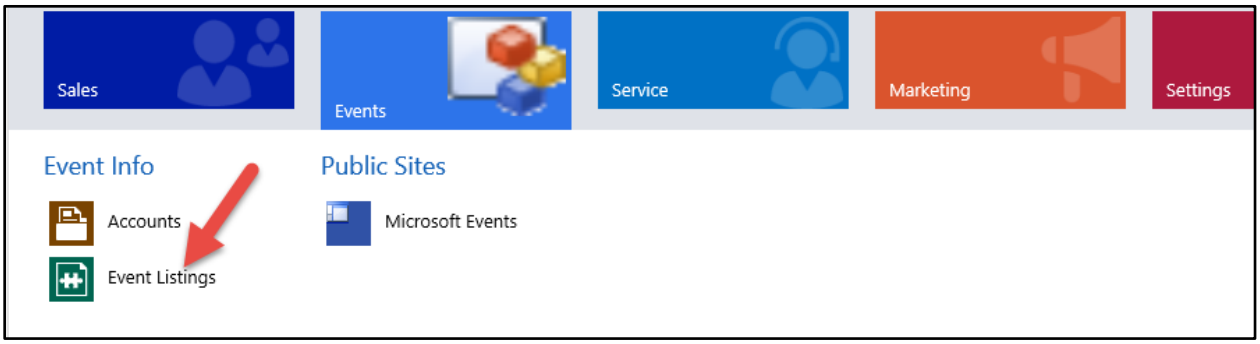
5) Go to your Dynamics 365 organization, refresh the page, and click on the menu button. Events should now appear second in the menu.

6) Click **Events**.
7) Click **Microsoft Events**.

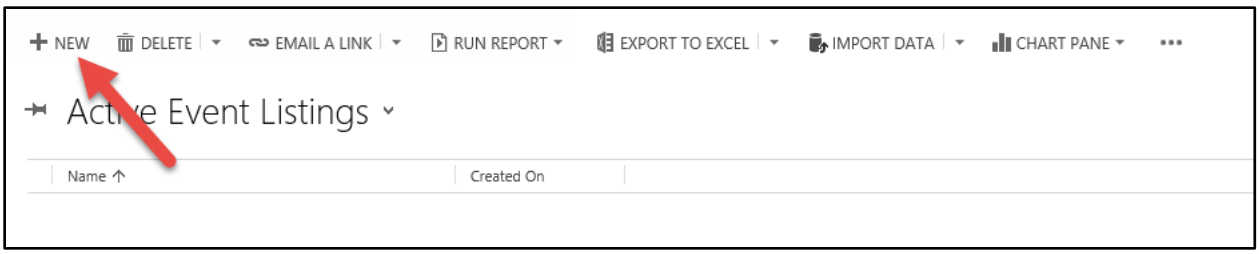
8) The external web page will be loaded.



9) Click the **Main menu** button, select **Events** and then click **Event Listings**.

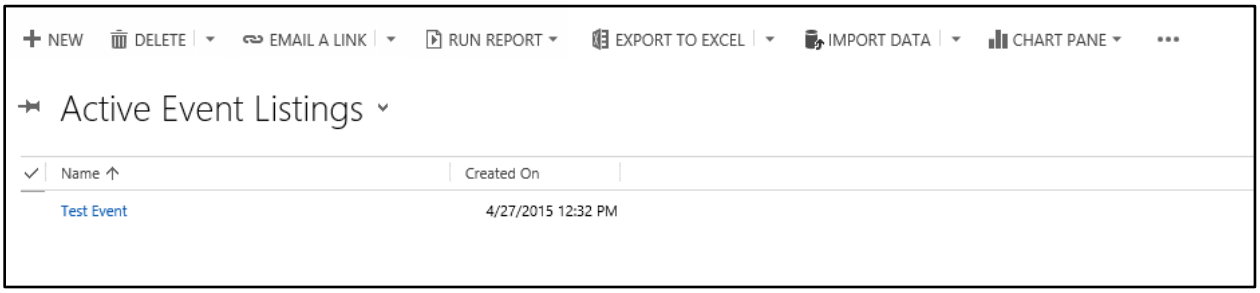


10) Event Listings will be loaded. Click **New**.



11) Enter **Test Event** for **Name** and click **Save & Close**.

12) The new Event Listing record will now be shown.



Lab 02B – Working with Command Buttons

Overview

In this lab you will be customizing Dynamics 365 command bar by Adding, Modifying, and Removing command buttons.

There are different ways that the Command Bar can be customized. You could manually edit the Customizations.XML file to reflect the changes desired, but oftentimes a 3rd party tool can make that process much easier. To demonstrate this process, you will be using a community tool to assist you in modifying the Dynamics 365 command bars.

What you will learn

- How to Add a Command Button
- How to Hide an existing Command Button
- How to Modify an existing Command Button

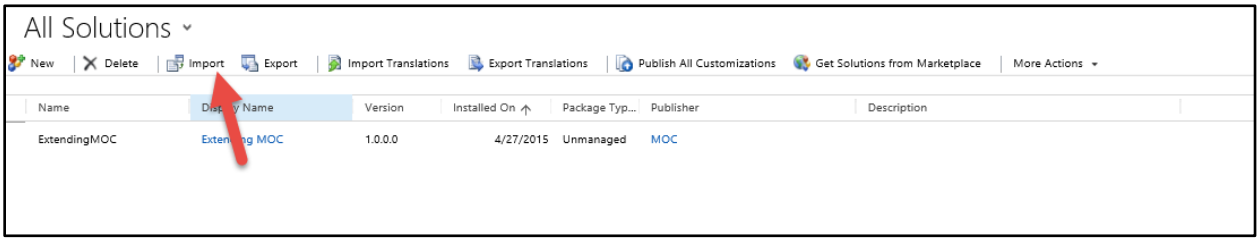
Working with Command Buttons

In this exercise you download and import Ribbon Workbench for Dynamics. Afterwards, you will use it to add, hide, and move buttons.

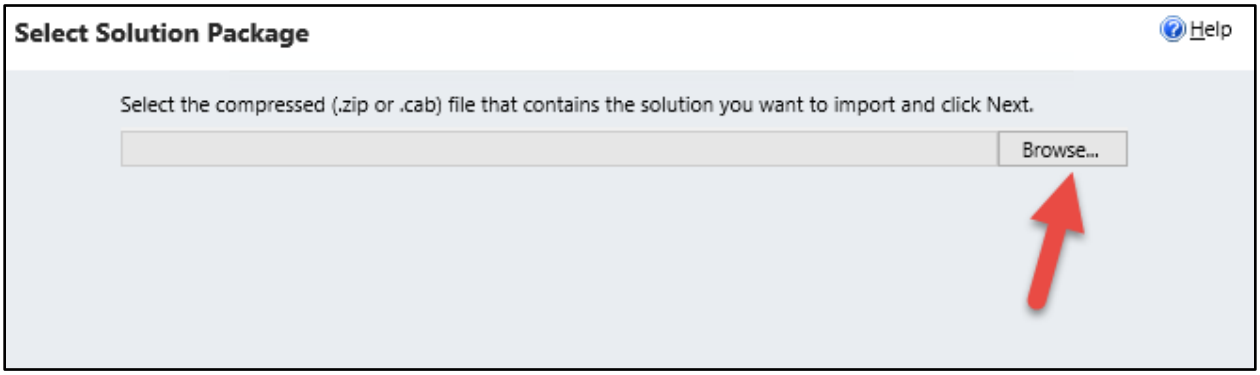
Task 1: Get and Import Ribbon Workbench

In this task you will get the Ribbon Workbench and import it to your lab environment.

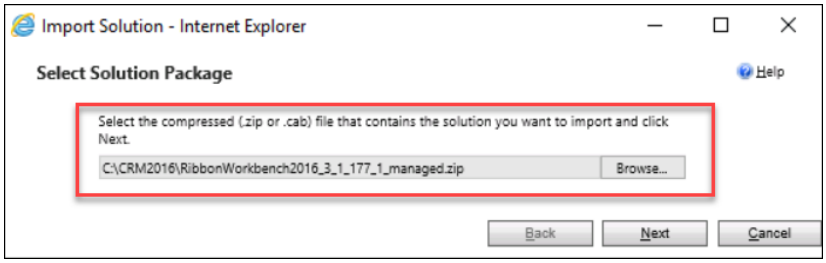
1) In your Dynamics 365 organization, navigate to **Settings | Solutions** and click **Import**.



2) Click **Browse**.

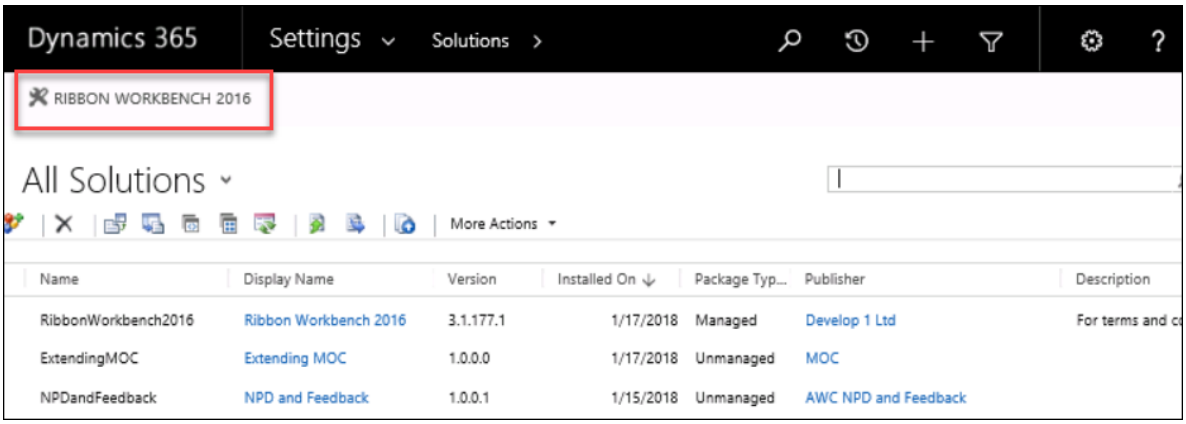


3) Navigate to the **C:\CRM2016** folder and double-click the **RibbonWorkbench2016_3_1_177_1_managed.zip** file.



4) Click **Next**.
5) Click **Next** again and then click **Import**.
6) Click **Close**, after import is completed.

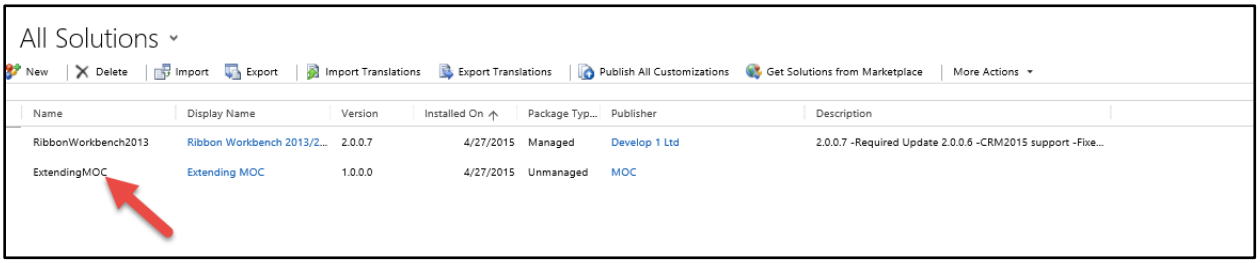
7) Refresh the browser and you will see new **Ribbon Workbench 2016** button.



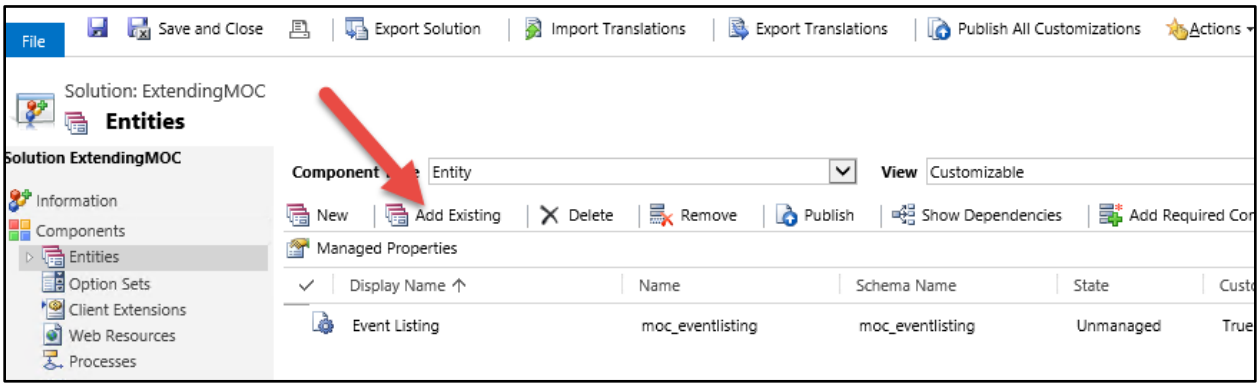
Task 2: Prepare Your Solution

In this task you add the **Account** entity to the **ExtendingMOC** solution. Ribbon Workbench only is able to edit entities that are in the solution you choose. You always want to use a solution with the least number of entities with Ribbon Workbench to minimize impact on others working in the same CRM.

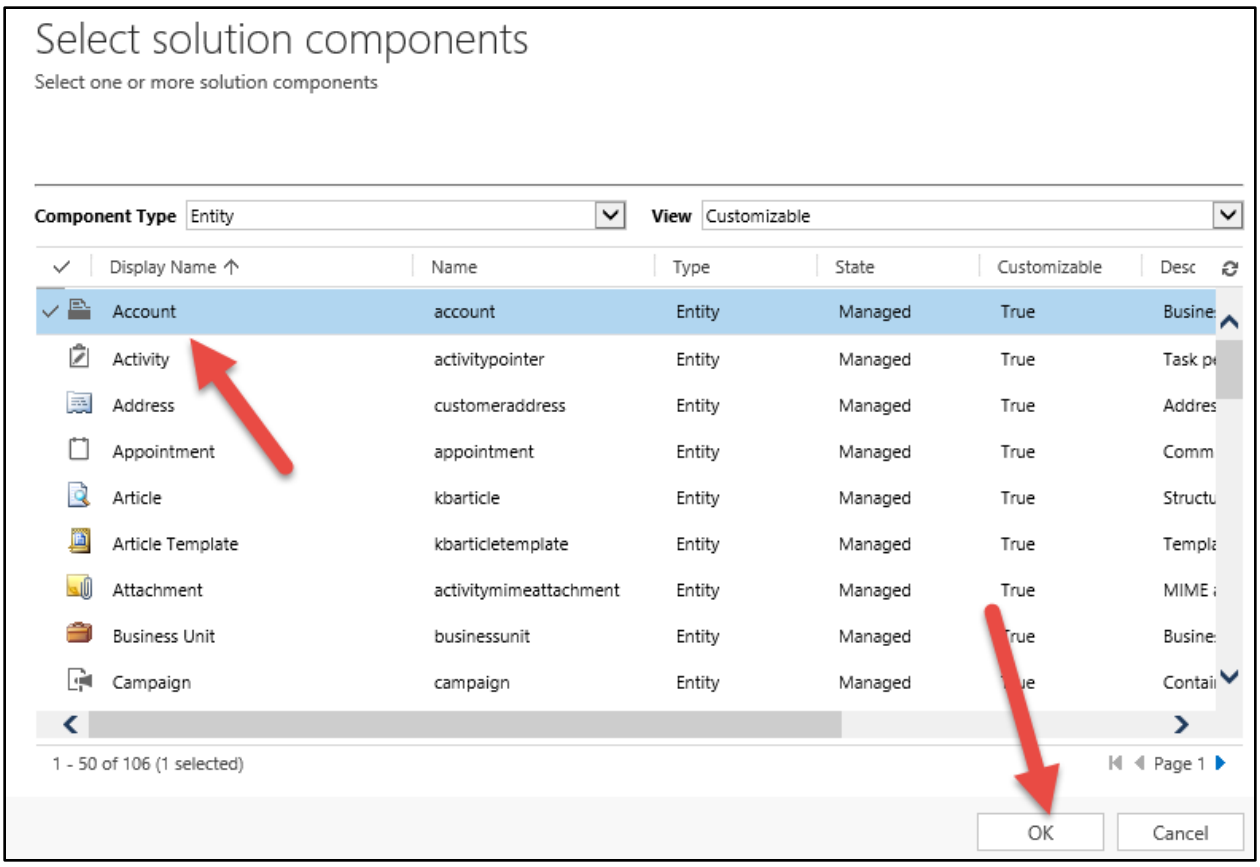
1) Navigate to **Settings | Solutions** and open the **ExtendingMOC** solution.



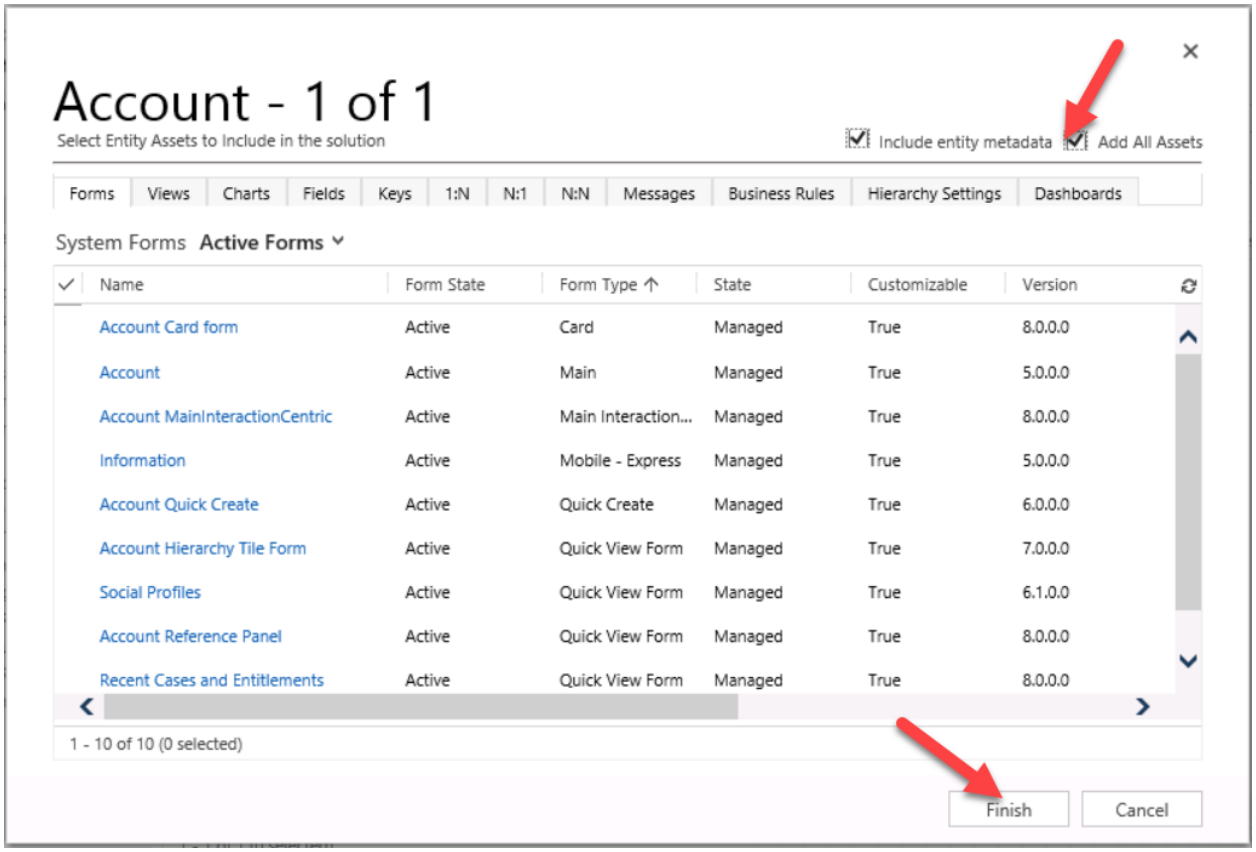
2) Select **Entities** and click **Add Existing**.



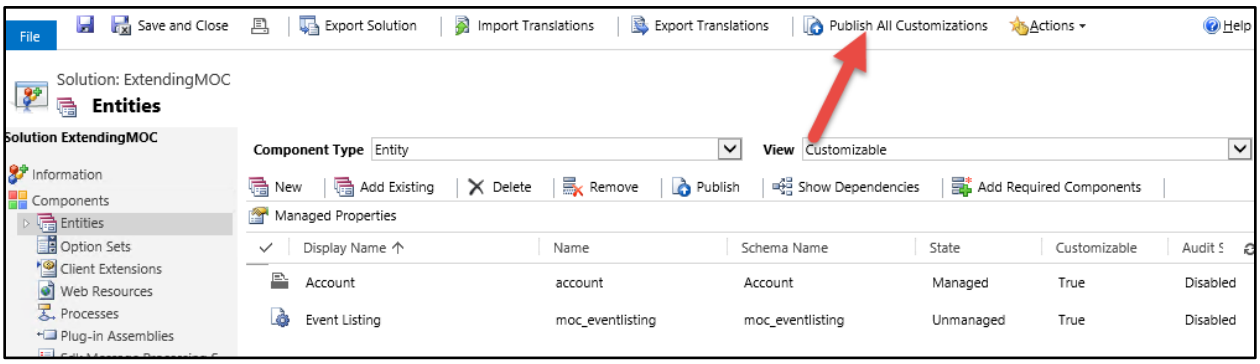
3) Select **Account** and click **OK**.



4) Click **Add All Assets** and then click **Finish**.



5) Click **Publish All Customizations**.

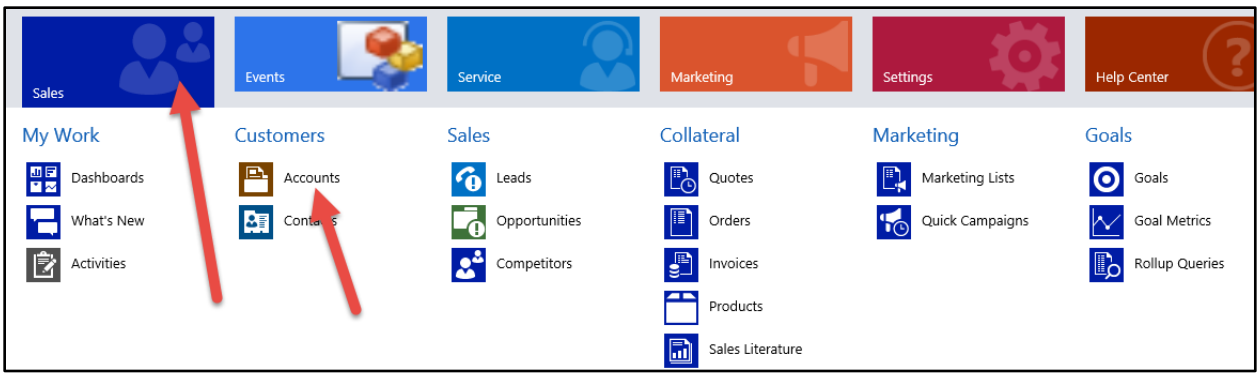


6) Close the solution explorer after publish completes.

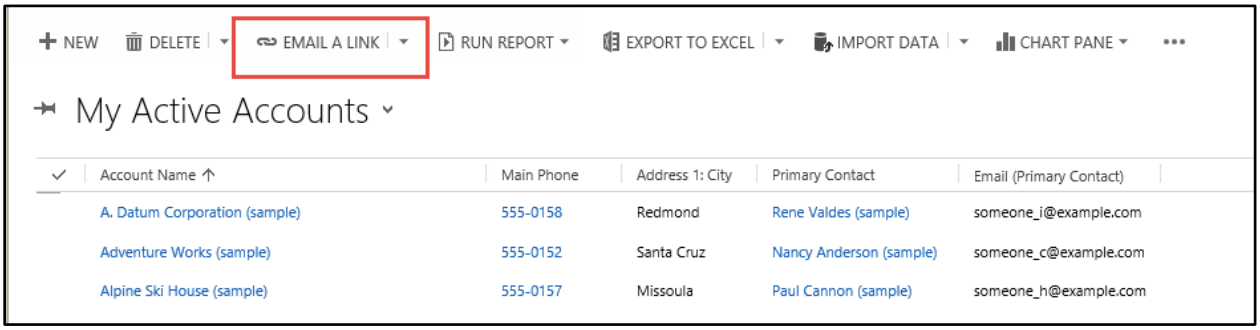
Task 3: Hide Command Button

In this task you will hide the Email a Link command button for the Account Entity. Before you hide the Email Link command button, take a moment to see how it is displayed in the application.

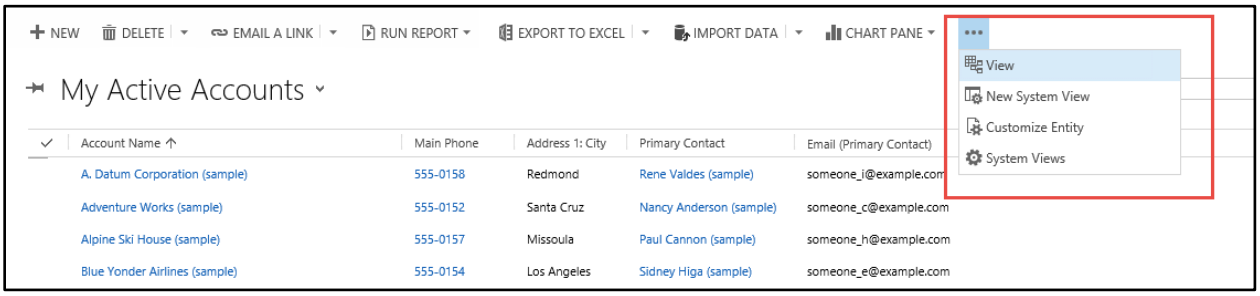
1) Navigate to **Sales | Accounts**.



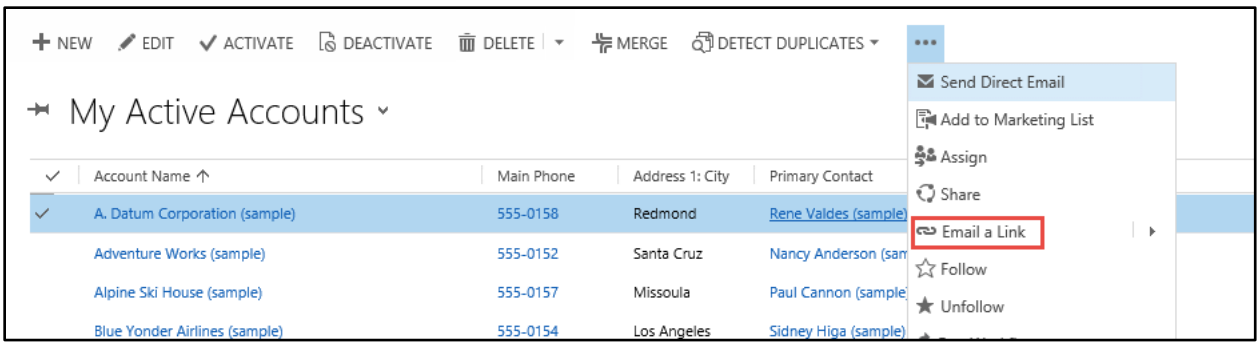
2) Notice that included in the command bar buttons is **Email a Link** button. This is the button that you will be hiding.



3) Click on the **More Commands** button […], notice that you will not see the **Email a Link** button.

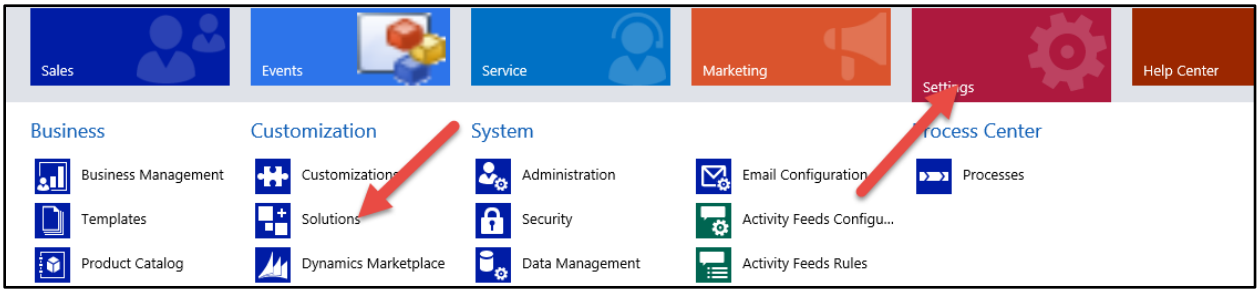


4) Select an Account record and click on the **More Commands** button. When an Account record is selected, the **Email a Link** button is moved under **More Commands**. You will also hide this button.

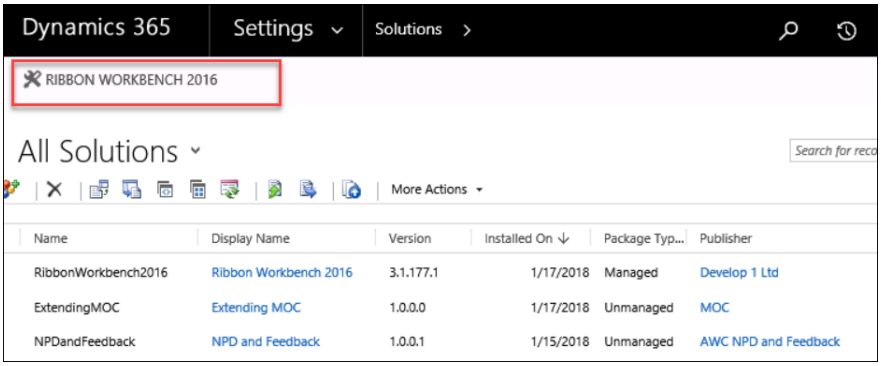


Let's hide the **Email a Link** Command Button.

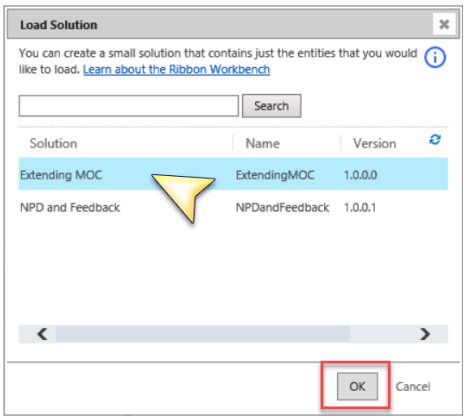
5) Navigate to **Setting | Solutions**.



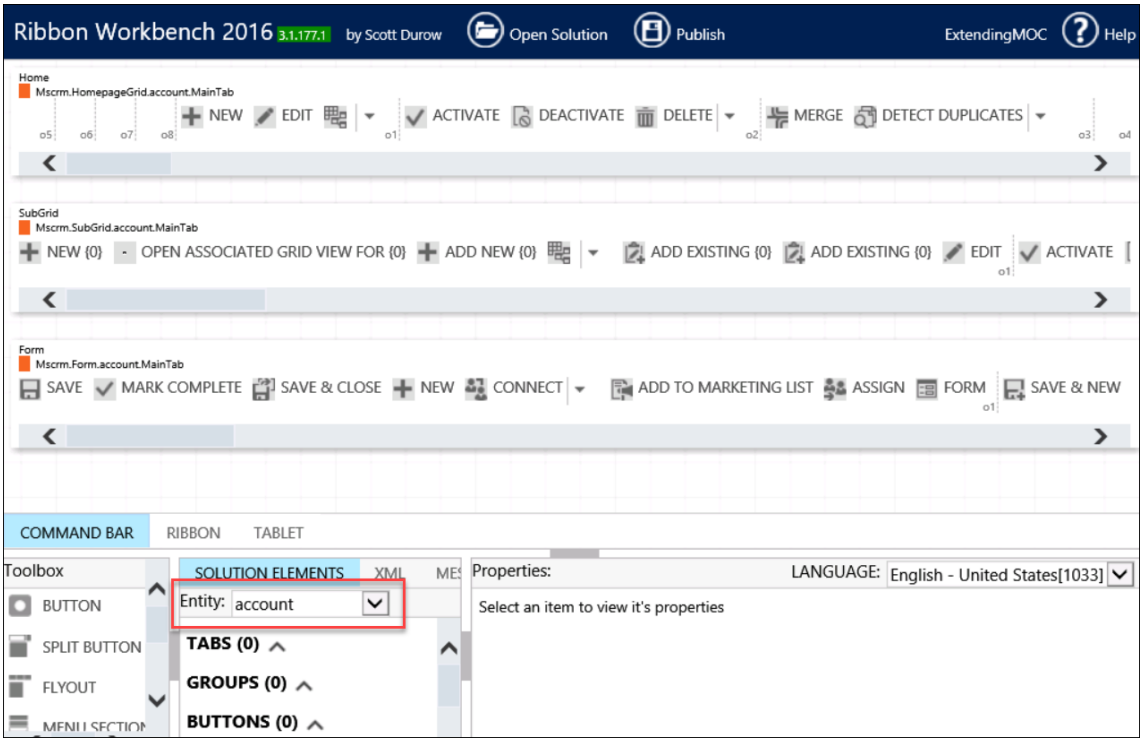
6) Click **Ribbon Workbench 2016**.



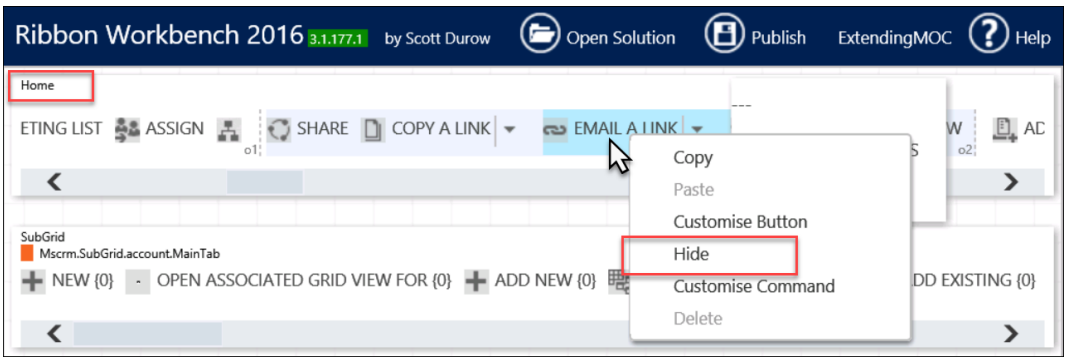
7) Select the **ExtendingMOC** solution and click **OK**.



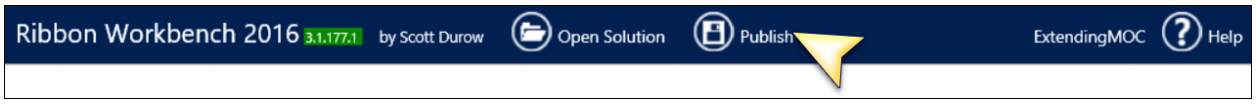
8) Make sure that the **Account** entity is selected in the **Entity** list.



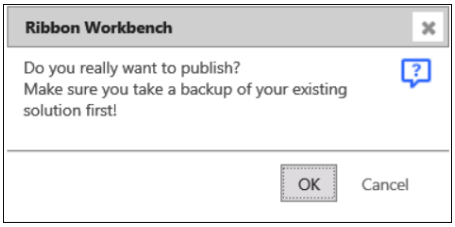
9) Under **Home**, scroll to the right, locate and right-click the **Email a Link** button and click **Hide**.



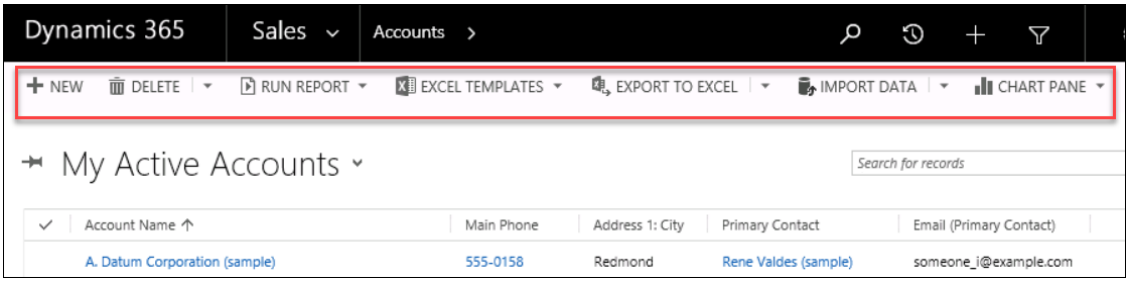
10) Click **Publish**.



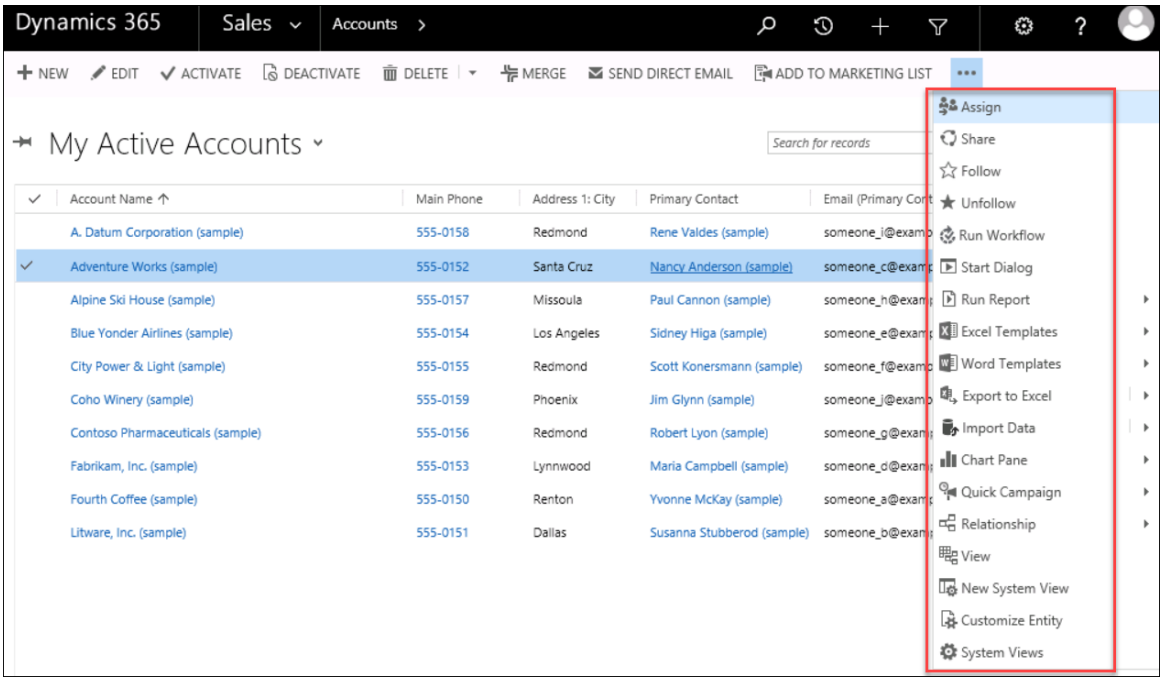
11) Click **OK**.



12) After finishing the **Publish** process, navigate to **Sales | Accounts**. You will need to refresh the page. The **Email a Link** button is no longer visible.



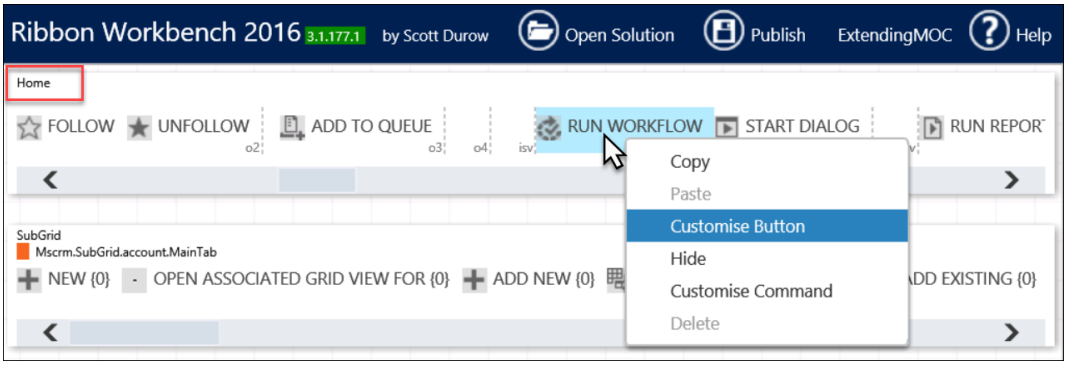
13) Select a record and click **More Commands**. The **Email a Link** button will not be visible under **More Commands**.



Task 4: Move Command Button

The Run Workflow button is usually displayed when one or more records are selected in the application. In this task you will move the Run Workflow command button from its current location to the main area of the command bar.

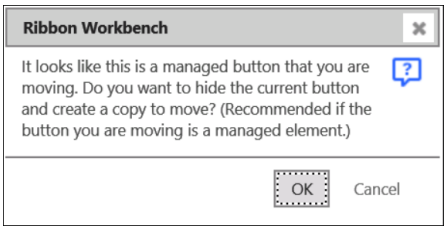
1) If the **Ribbon Workbench 2016** tool is not open, navigate to **Settings | Solutions** and click **Ribbon Workbench 2016**. Select the **ExtendingMOC** solution and click **OK**.
2) Make sure that the **Account** entity is selected in the **Entity** list.
3) Under **Home**, scroll to the right, locate and right-click the **Run Workflow** button and click **Customize Button**.



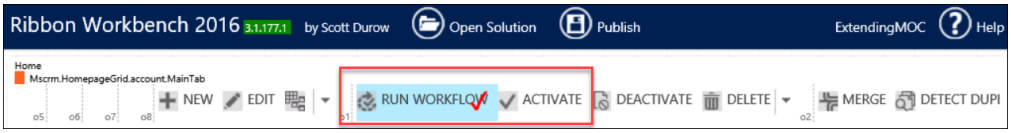
4) The element will go into editing mode, (indicated by the red checkmark). You can now drag the button to its new destination.

5) Drag the **Run Workflow** button just before the **Activate** button.

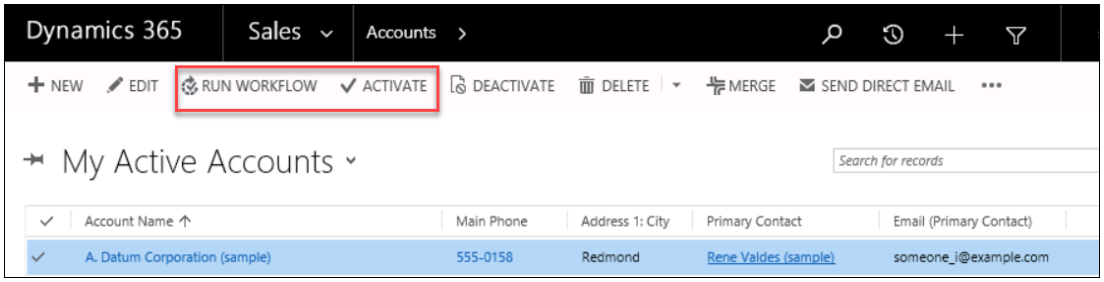
6) Click **OK**.



7) The **Run Workflow** button will now be before the **Activate** button.



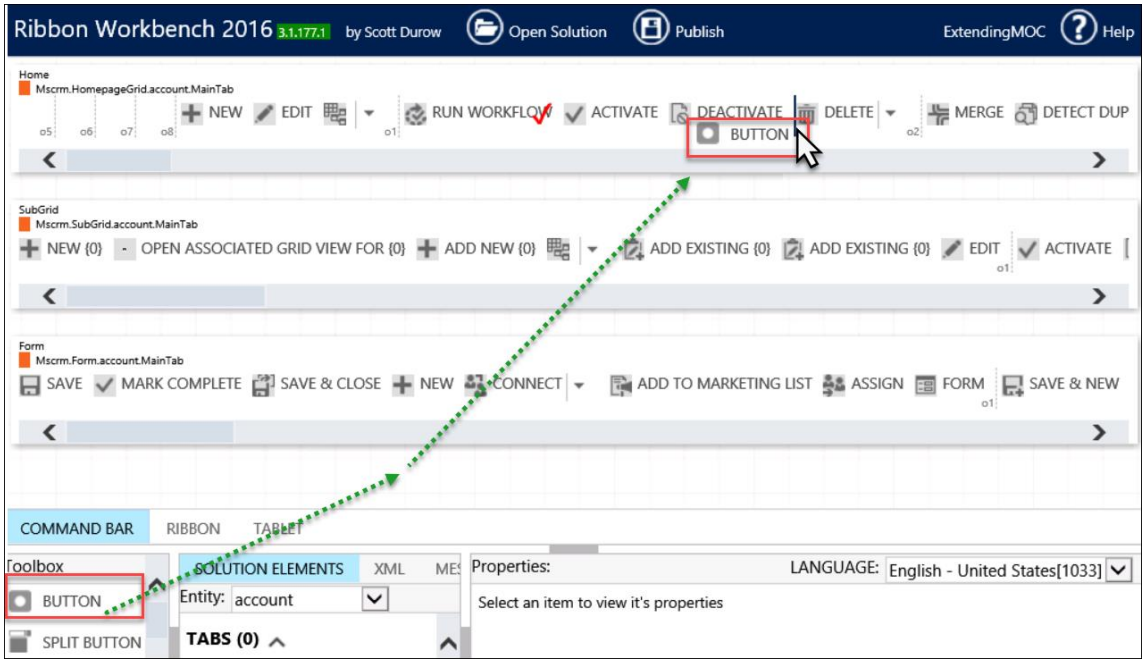
8) Click **Publish**.
9) Click **OK**.
10) After finishing the **Publish** process, navigate to **Sales | Accounts**. You will need to refresh the page.
11) Select an account record. The **Run Workflow** button will now be in the main area, before the **Activate** button.



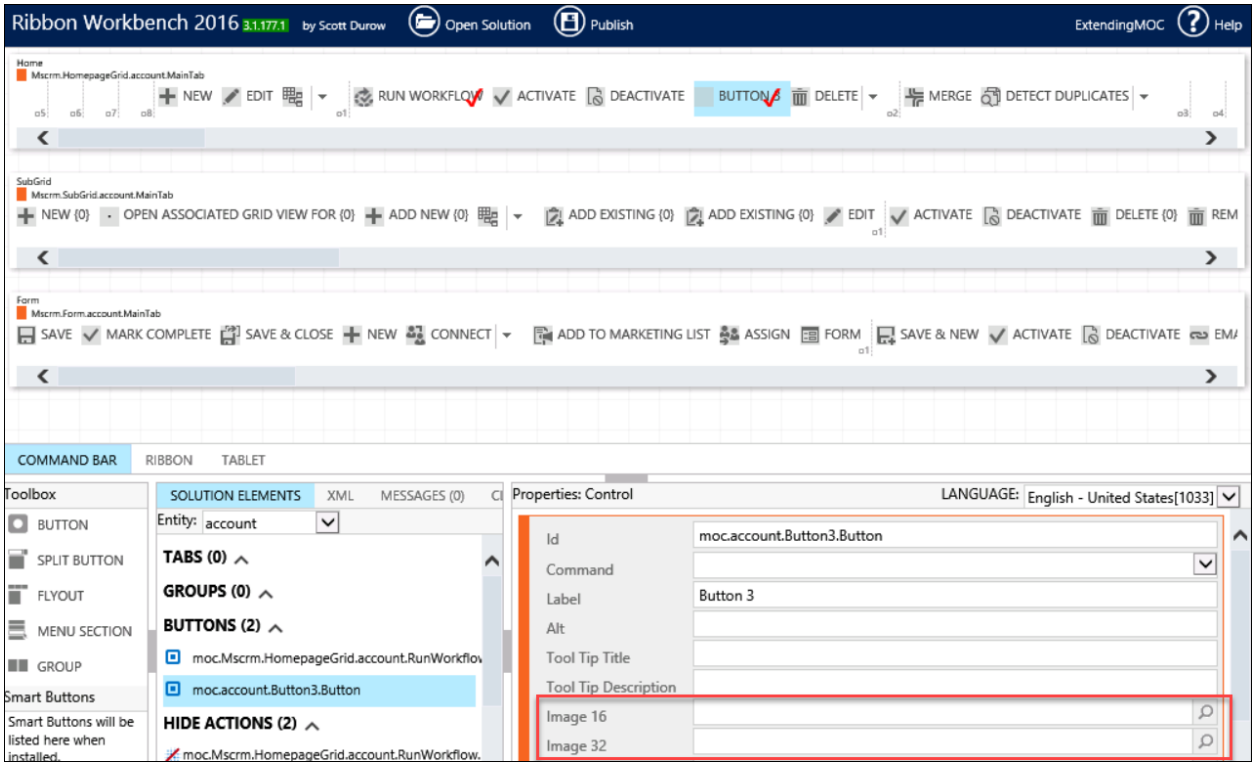
Task 5: Add Command Button

In this task you will add a command button that will open an external link to Bing.

1) If the **Ribbon Workbench 2016** tool is not open, navigate to **Settings | Solutions** and click **Ribbon Workbench 2016**. Select the **ExtendingMOC** solution and click **OK**.
2) Make sure that the **Account** entity is selected in the **Entity** list.
3) Drag **Button** from the toolbox and place it between the **Deactivate** and **Delete** buttons in the **Home** Bar.

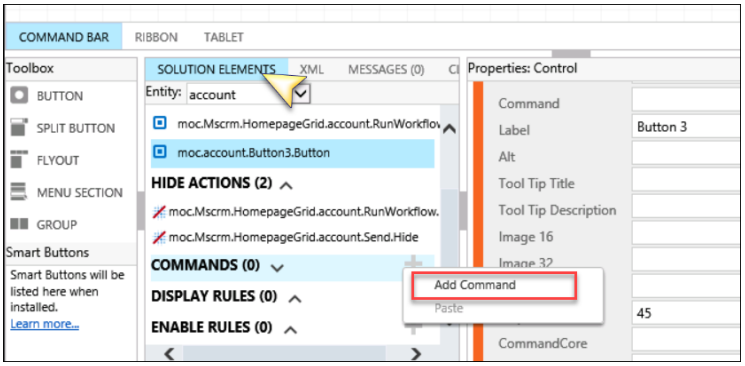


4) You can set Custom button images using the common lookups to select image web resources. For this lesson, we will just use the default here.

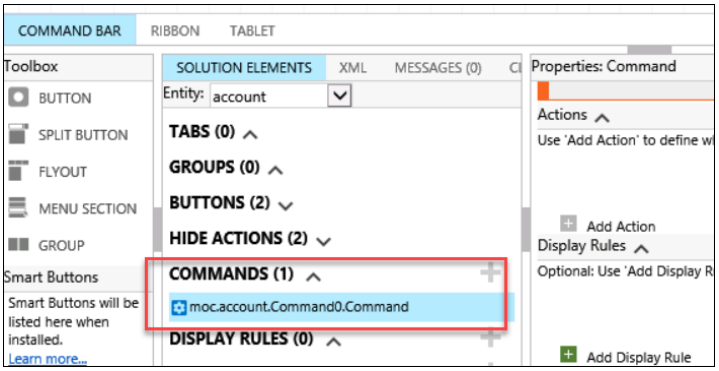


Next you will add a command to the button.

5) In the **Solution Elements** tab, right click **Commands**, and click **Add Command**.

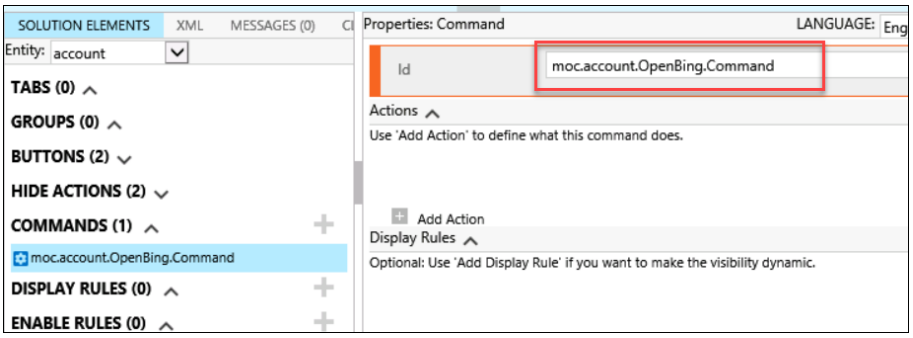


6) Expand **Commands** and you will see the new command.

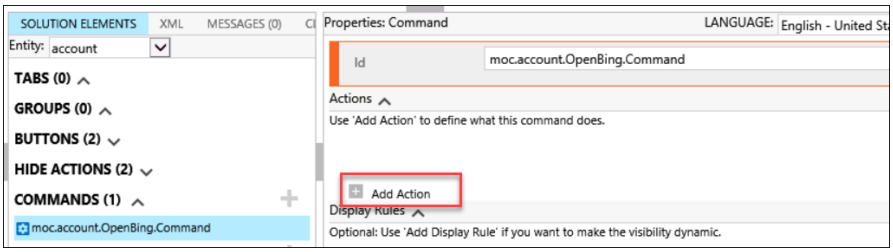


Note: *Remember that each button needs to have a unique name defined. The Ribbon Workbench uses a specific format to assist you in defining unique names.*

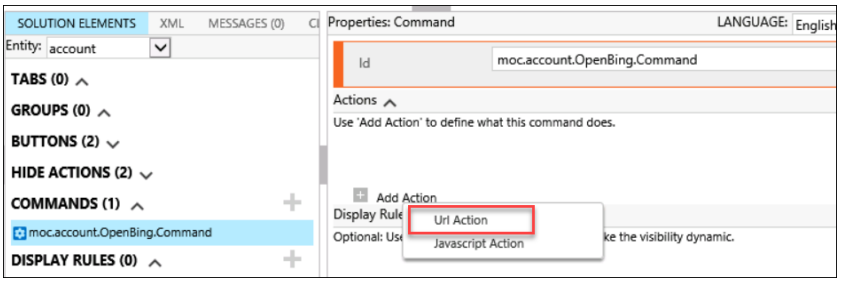
7) Change the **Id** to **moc.account.OpenBing.Command**



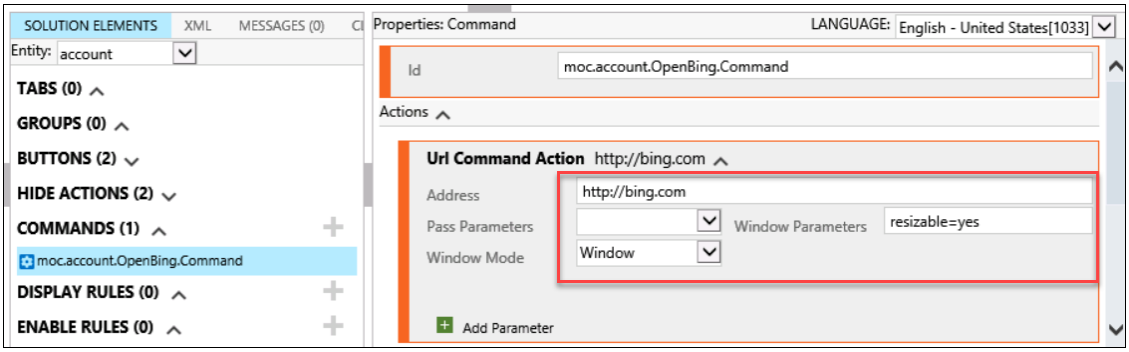
8) Click on the **Add Action** button.



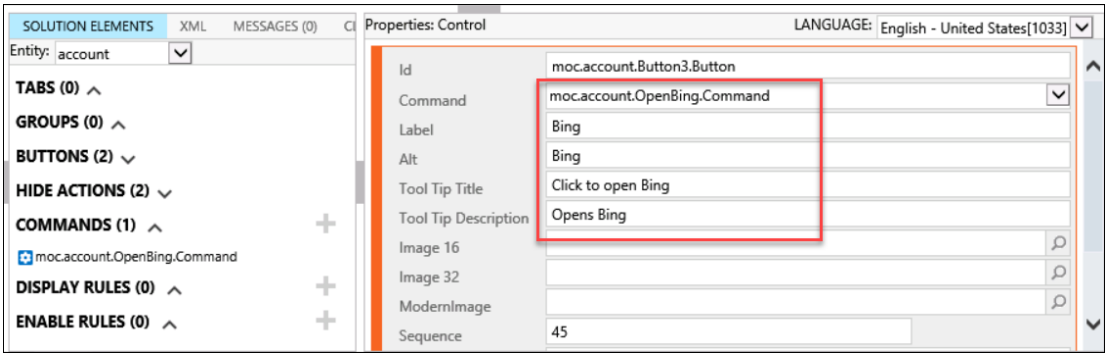
9) Click **Url Action**.



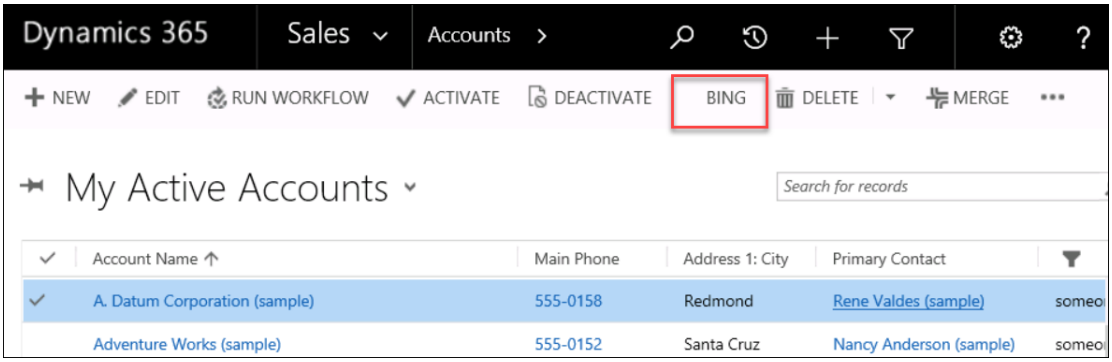
10) Enter **http://bing.com** for **Address**, select **Window** for **Window Mode** and type **resizable=yes** for **Window Parameters**.



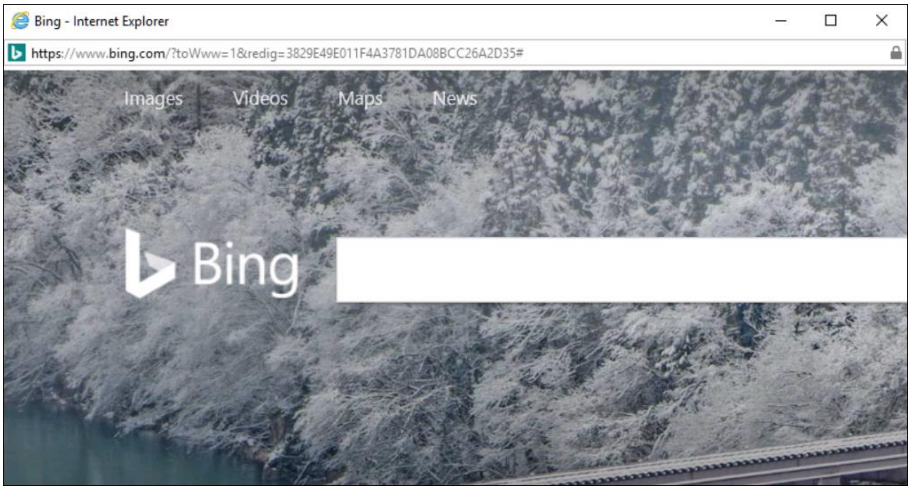
11) On the **Home** Section, select **Button**, (The **ButtonControl** Properties will display).
12) Fill in the **Properties** of the button as defined below:
    a. Set **Command** to: **moc.account.OpenBing.Command**
    b. Set both the **Label** and **Alt** fields to: **Bing**
    c. Set **Tool Tip Title** to: **Click to open Bing**
    d. Set **Tool Tip Description** to: **Opens Bing**



13) Click **Publish**.
14) Click **OK**.
15) After finishing the **Publish** process, navigate to **Sales | Accounts**. You will need to refresh the page. The **Email a Link** button is no longer visible.
16) Select an **Account** record. You will see the new **Bing** button.



17) Click the new **Bing** button.
18) The button will open **Bing** on a new browser window.



19) Close all the Web browser windows.

Lab 03 – Form Scripting

Overview

In this lab you will create add Vendor section to the Account form, add the Relationship Type field to the Account form, and then create a script that will show the Vendor section if the Account is a Vendor and hide it if the account is not a Vendor.

What you will learn

- How to create a project for Script Web Resources using the CRM Developer Tools
- How to add client script Web Resources to your Dynamics 365
- How to use form events
- How to hide/show form sections

What you need to complete this lab

- CRM Developer Toolkit

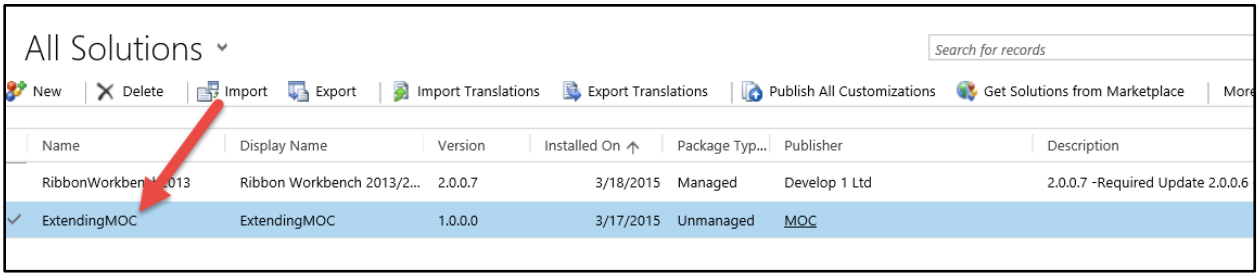
Exercise 1 – Add Fields and Vendor Section

In this exercise you will add the Relationship Type field to Account form and add new Vendor section to the Account form.

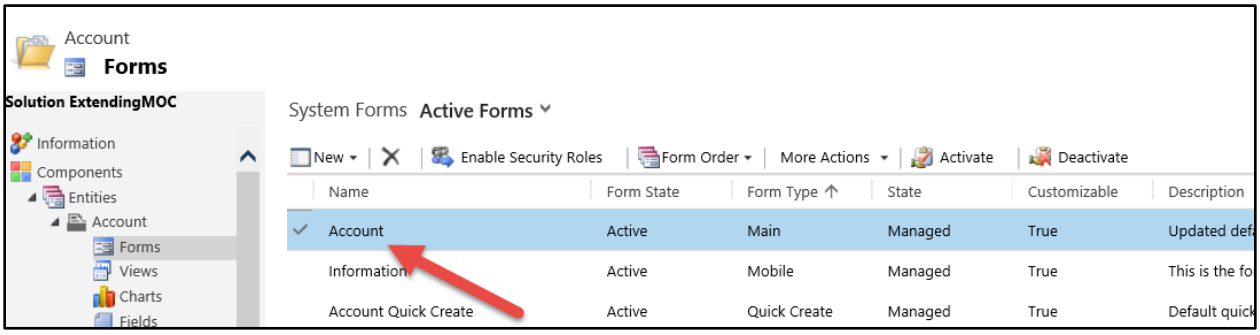
Task 1: Edit the Account Form

In this task you will add the Relationship Type to the Account form and create new Vendor section.

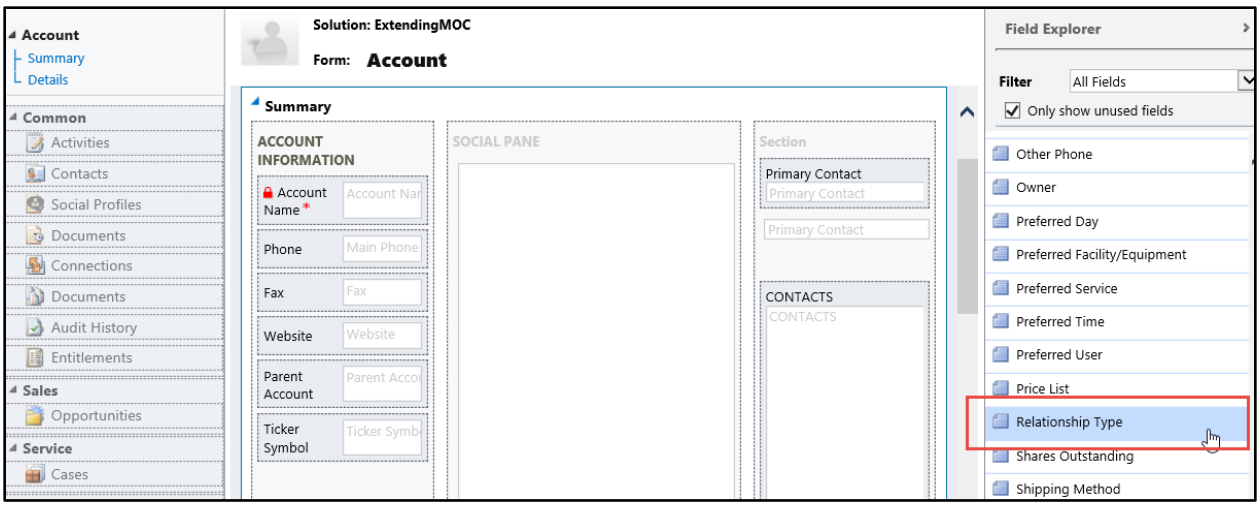
1) Navigate to **Settings | Solutions** and open the **ExtendingMOC** solution.



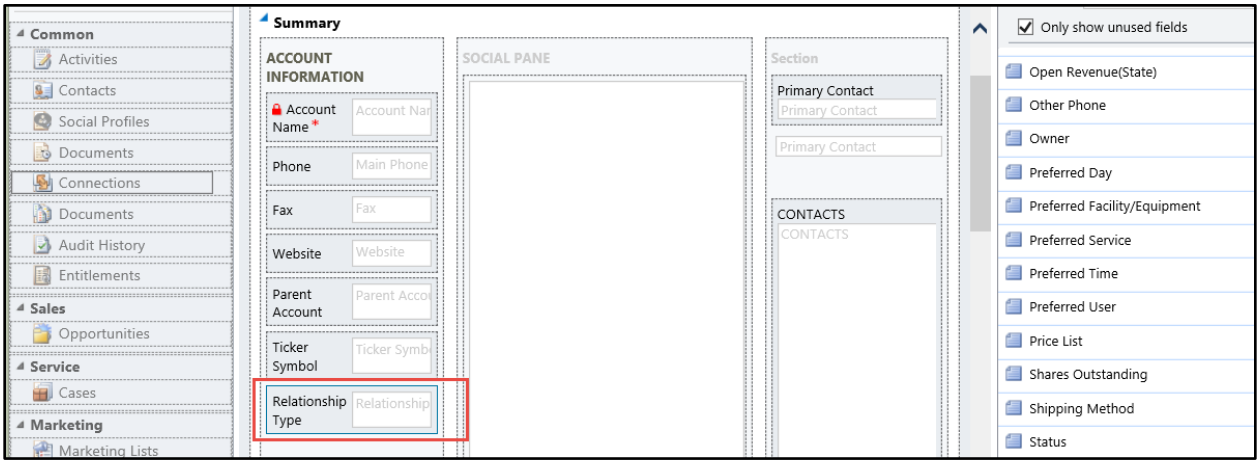
2) Expand **Entities**, expand the **Account** entity, select **Forms**, and open the **Main** account form.



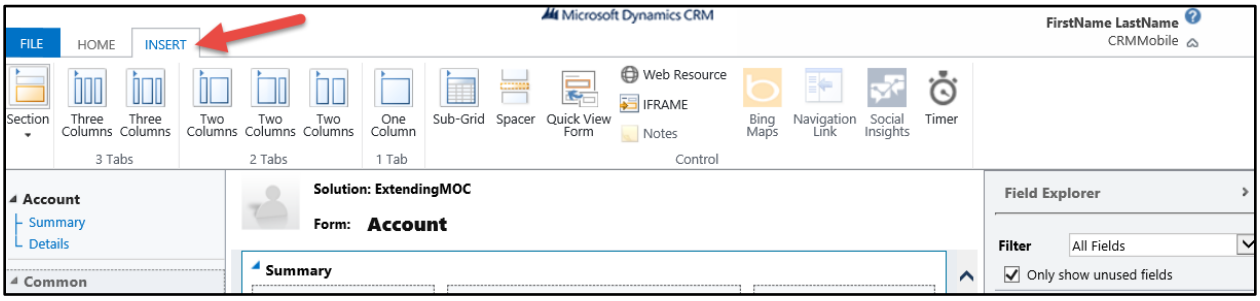
3) In the **Field Explorer**, locate the **Relationship Type** field.



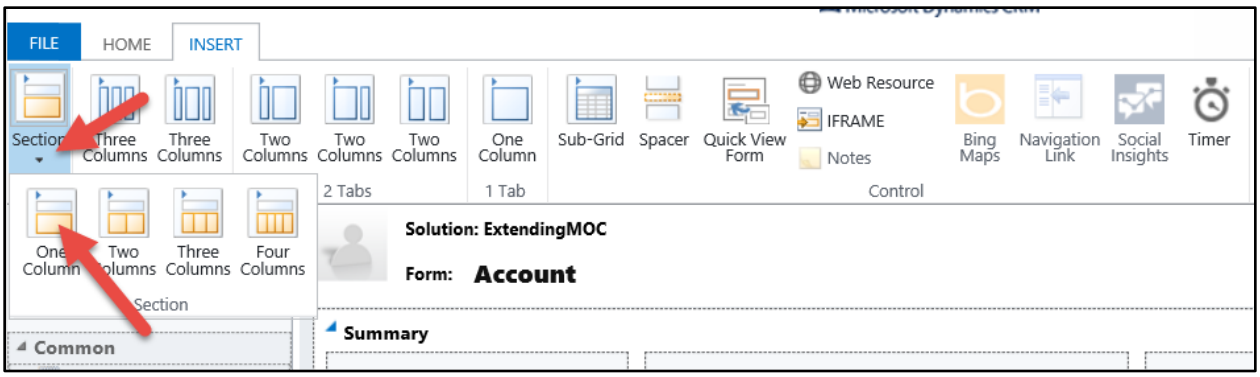
4) Drag the **Relationship** field and place it under the **Ticker Symbol** field.



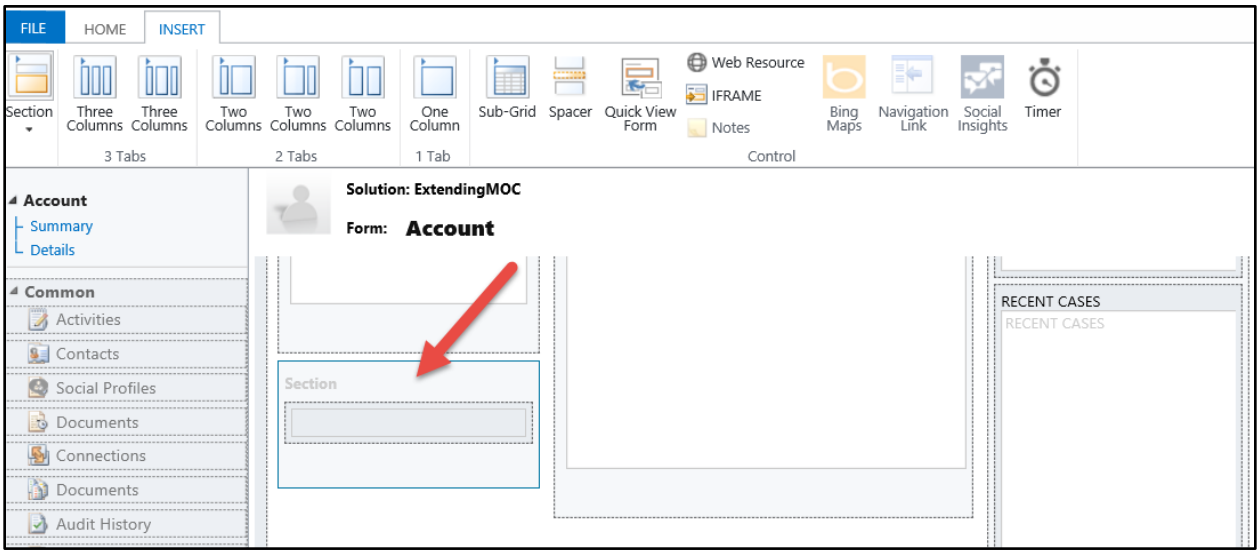
5) Select the **Insert** tab.



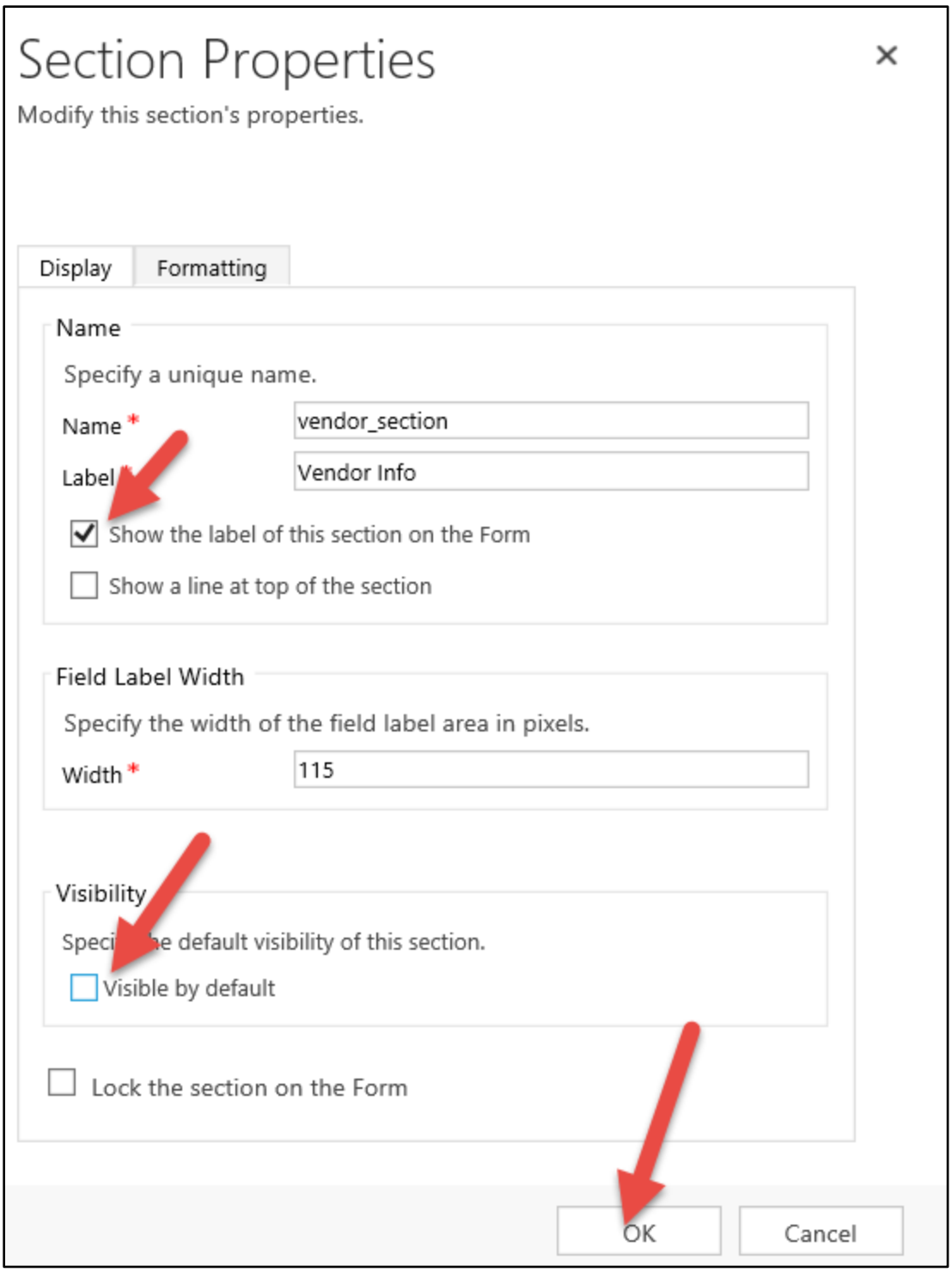
6) Click **Section** and select **One Column** Section.



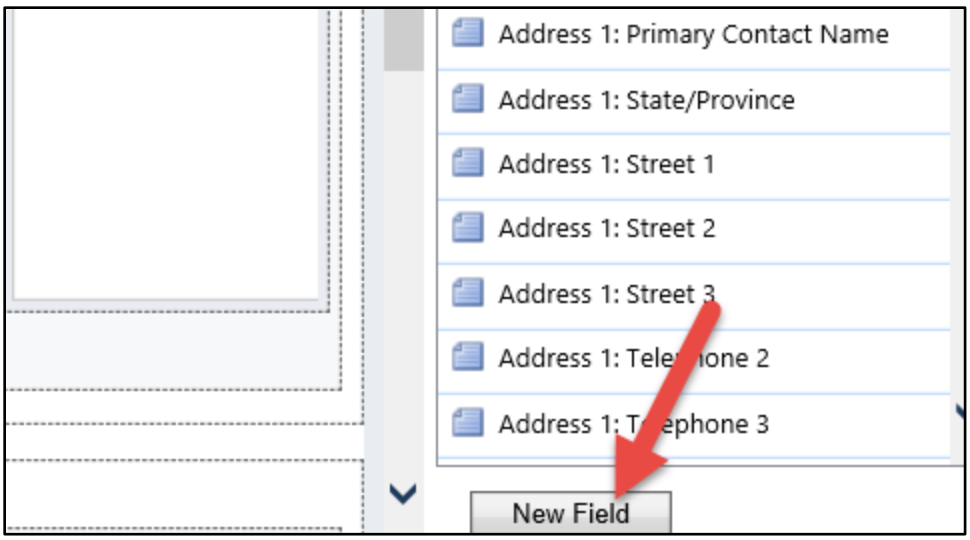
7) New section will be added to the currently selected tab. Double click on the new section.



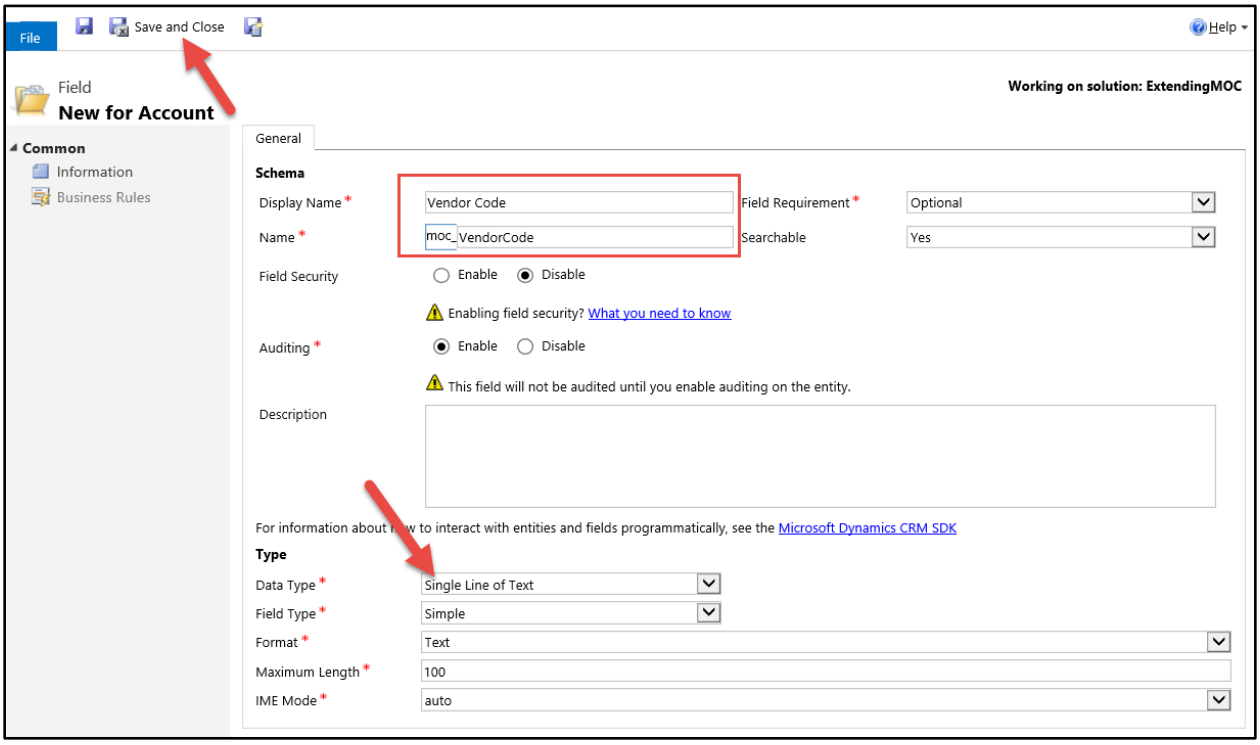
8) Enter **vendor_section** for **Name**, **Vendor Info** for **Label**, check the **Show Label** checkbox, uncheck the **Visible by Default** checkbox, and click **OK**.



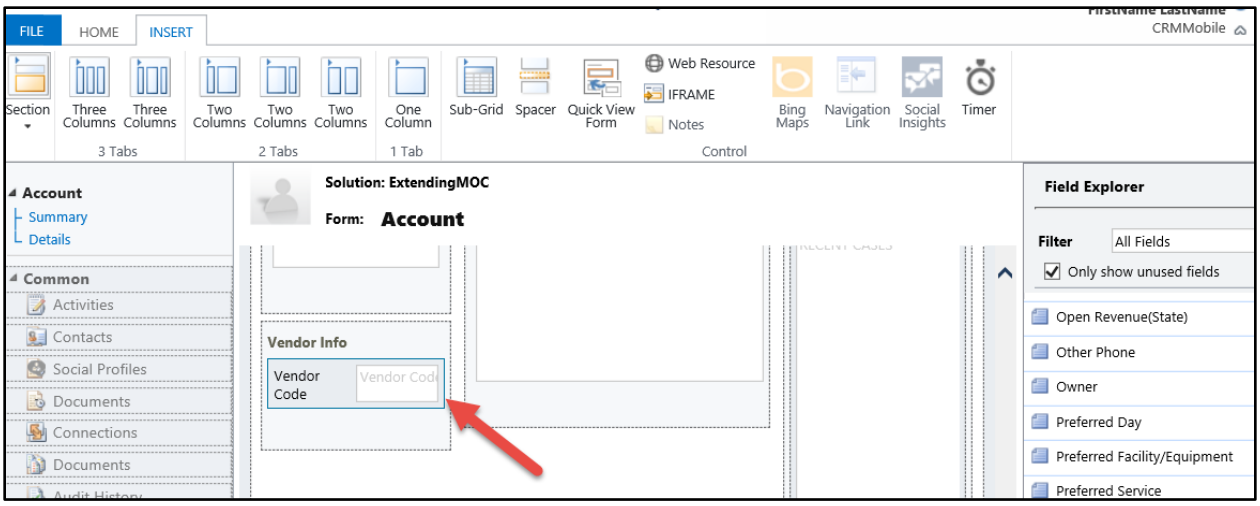
9) You will now add a new field to the **Vendor Info** section.
10) At the bottom of the **Field Explorer**, click **New Field**.



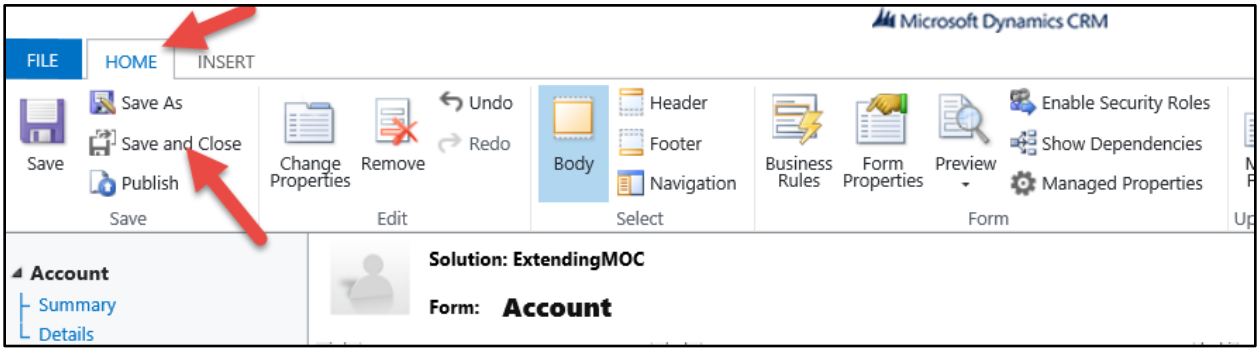
11) Enter **Vendor Code** for **Display Name**, select **Single Line of Text** for **Data Type**, then click **Save and Close**.



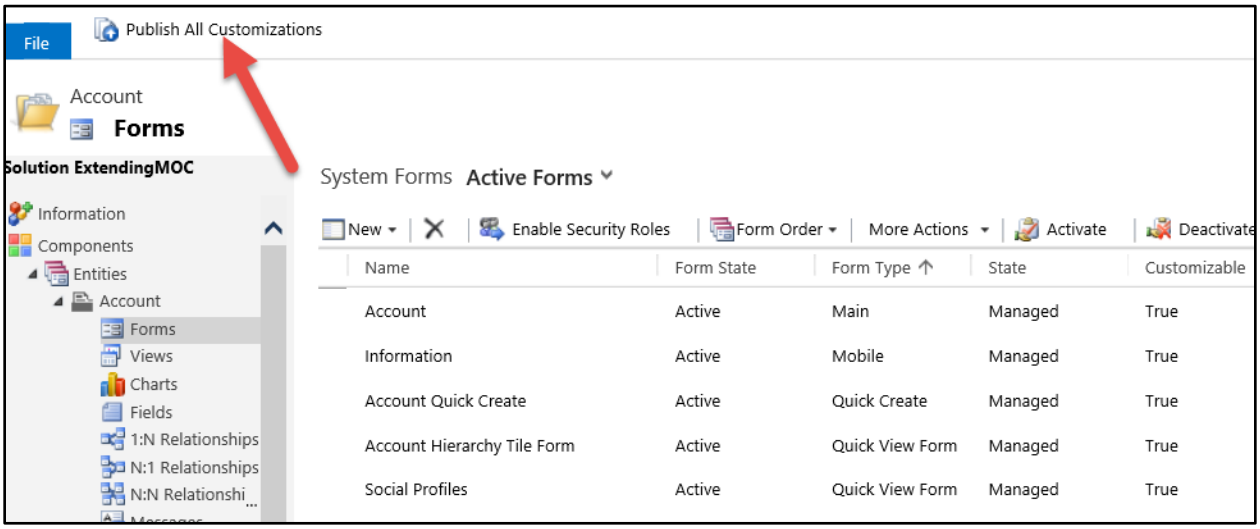
12) Drag the new field into the form and add it to the **Vendor Info** section.



13) Select the **Home** tab and click **Save & Close**.



14) Click **Publish All Customizations**.



15) Close the **Solution** window.

Exercise 2 – Setup Solution and Connect to CRM

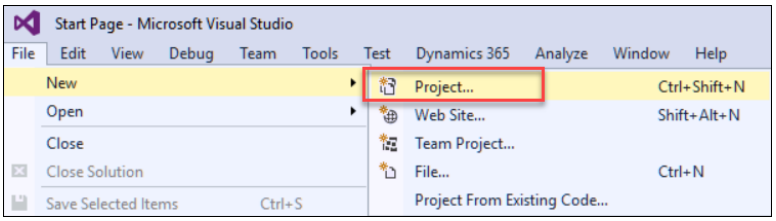
In this exercise you will be setting up a Visual Studio 2015 Solution using the CRM Developer Toolkit to manage and develop client side scripting. While it is not required to use the CRM Developer Toolkit to build CRM client scripting it can be helpful for managing and deploying the web resources.

Task 1: Create a new CRM Package Project

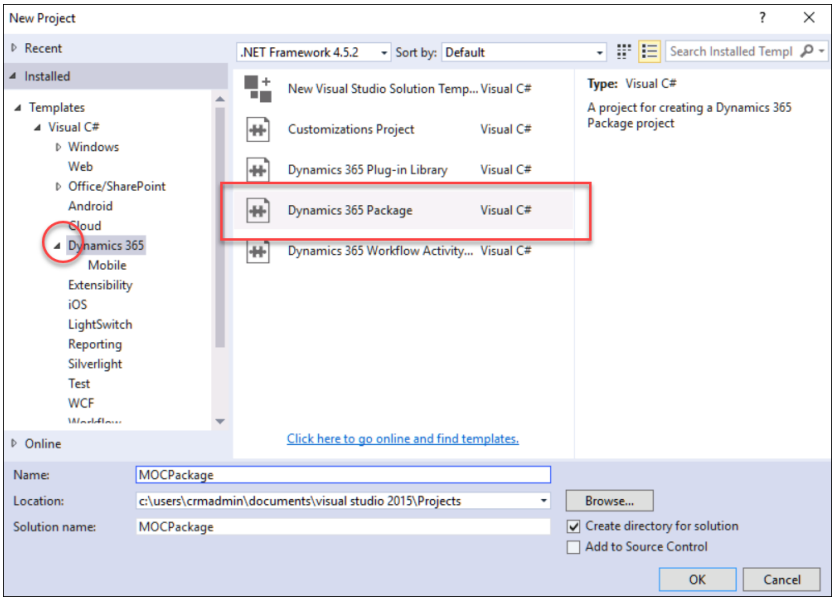
CRM Package is a special Dynamics 365 project template that is provided as part of the CRM Developer Toolkit. The CRM Package is designed to manage Web Resource components in a CRM solution.

1) Start Visual Studio 2015.

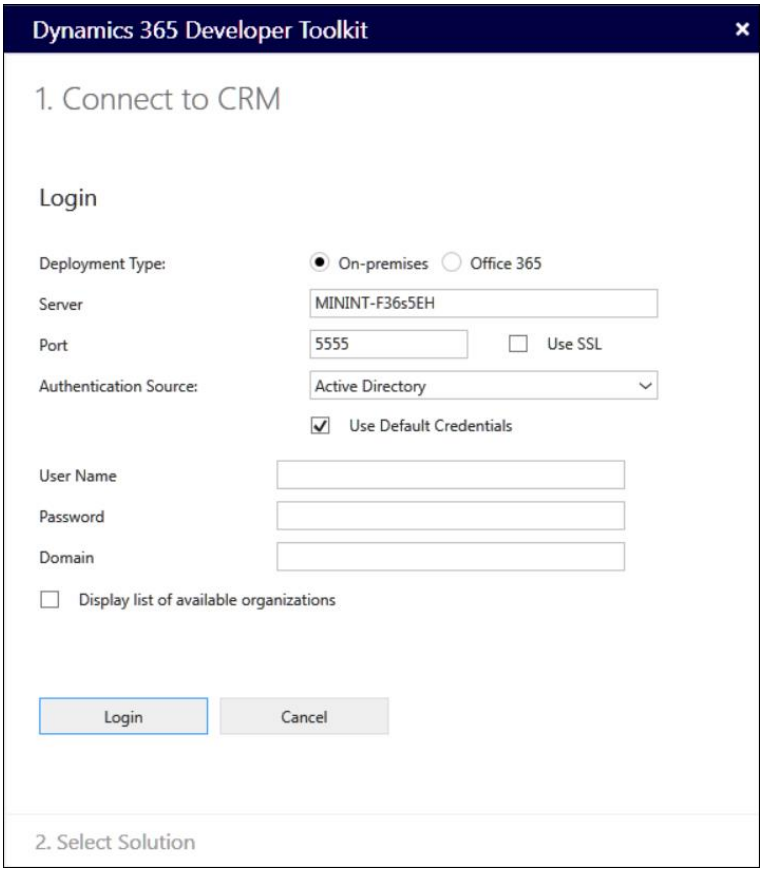
2) Click **File | New | Project**.



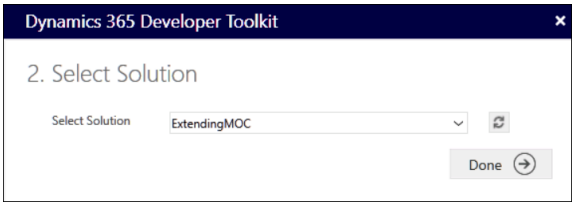
3) Under **Templates | Visual C# | Dynamics 365**, select **Dynamics 365 Package** from the list of templates, and enter **MOCPackage** as the name of the new project.



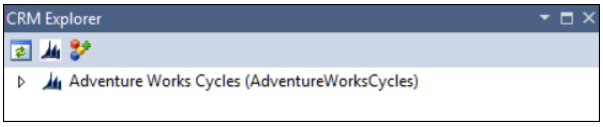
4) Click **OK**.
5) In the **Connect to CRM** dialog, in the **Server** textbox type **MININT-F36s5EH**.
6) In the Port Number textbox type **5555**.
7) In the **Authentication Source** list, make sure **Active Directory** is selected.
8) Select **Use Default Credentials**.



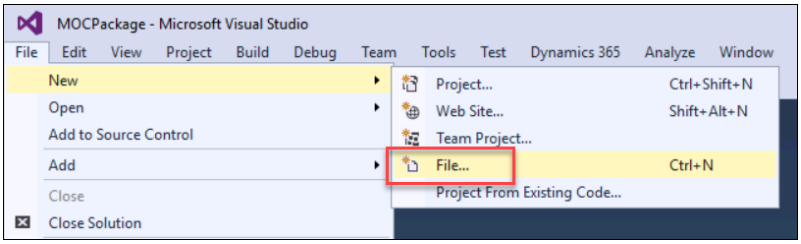
9) Click **Login**.
10) In the **Select Solution** dialog, in the **Select Solution** drop-down list select **ExtendingMOC**.



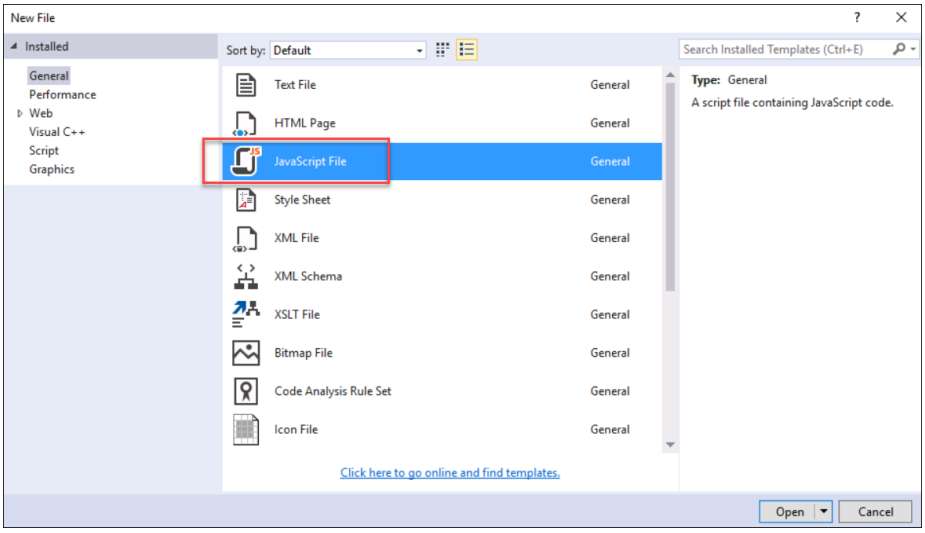
11) Click **Done**.
12) If the **CRM Explorer** window is open, close it.



13) In Visual Studio, click **File | New | File**.



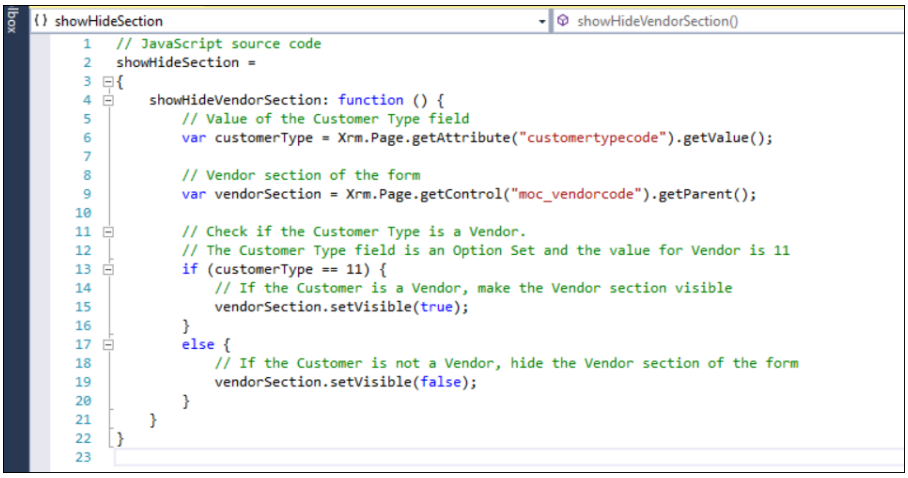
14) Select **JavaScript File**.



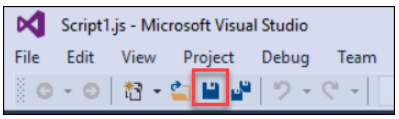
15) Click **Open**.
16) Add the following script to the Jscript file.

```
// JavaScript source code
showHideSection =
{
```

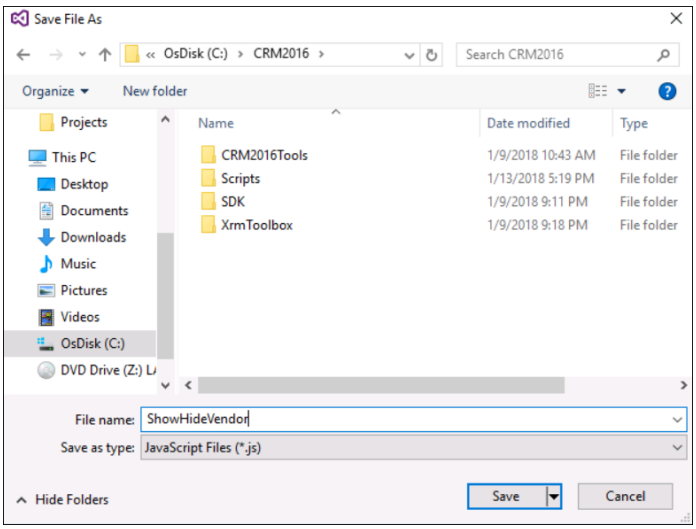
```
     showHideVendorSection: function ()
    {
        // Value of the Customer Type field
        var customerType = Xrm.Page.getAttribute("customertypecode").getValue();

        // Vendor section of the form
        var vendorSection = Xrm.Page.getControl("moc_vendorcode").getParent();

        // Check if the Customer Type is a Vendor.
        // The Customer Type field is an Option Set and the value for Vendor is 11
        if (customerType == 11)
        {
            // If the Customer is a Vendor, make the Vendor section visible
            vendorSection.setVisible(true);
        }
        else
       {
            // If the Customer is not a Vendor, hide the Vendor section of the form
            vendorSection.setVisible(false);
        }
    }
}
```



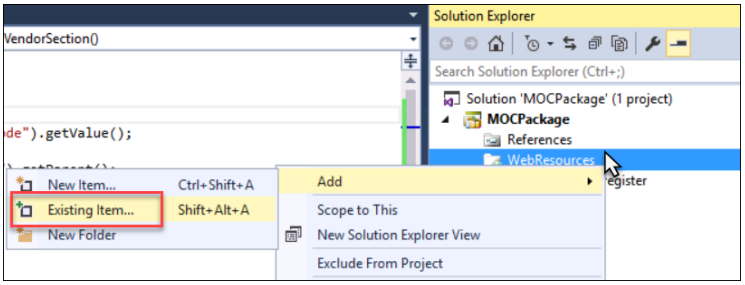
17) Click the **Save** icon on the toolbar.



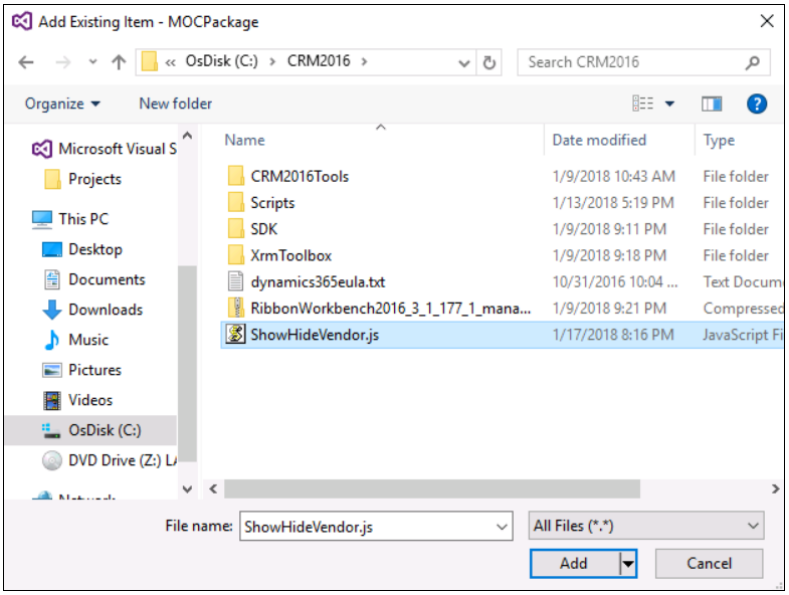
18) Navigate to **C:\CRM2016** folder and type **ShowHideVendor** as the name of the file.



19) In the **Solution Explorer**, right-click the **WebResources** folder, click **Add** and then select **Existing Item**.

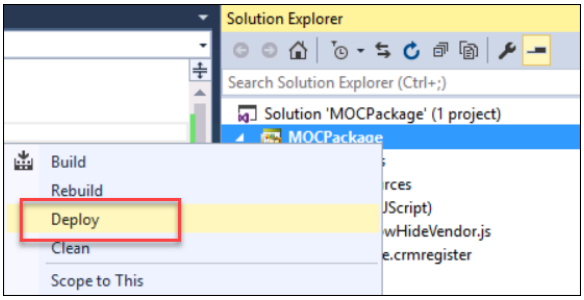


20) Navigate to **C:\CRM2016** folder and click the **ShowHideVendor.js** file.

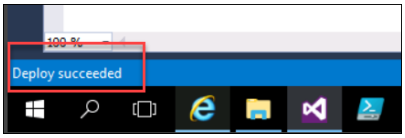


21) Click **Add**.

22) You are now ready to deploy. In the **Solution Explorer**, right click the **MOCPackage** project and click **Deploy**.



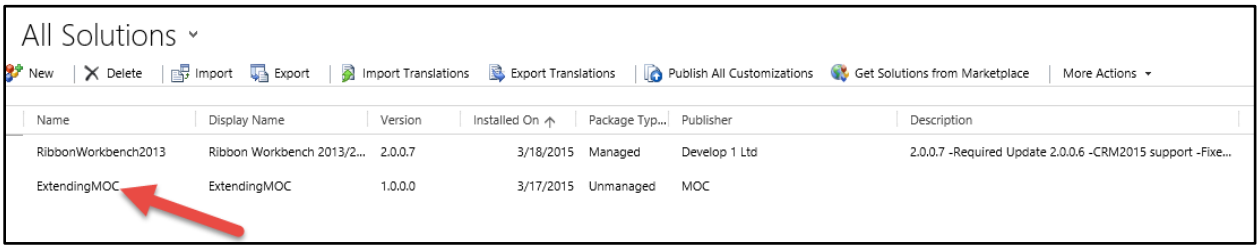
23) Wait for deployment to succeed.



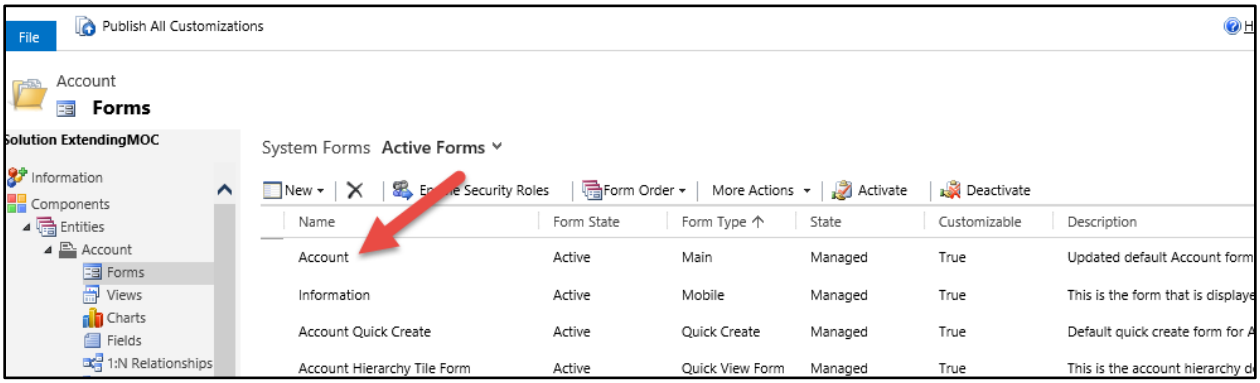
Task 2: Register Form Event Handler

In this task you will register the script you created in task 1 and insure your script gets called when a record gets loaded or updated.

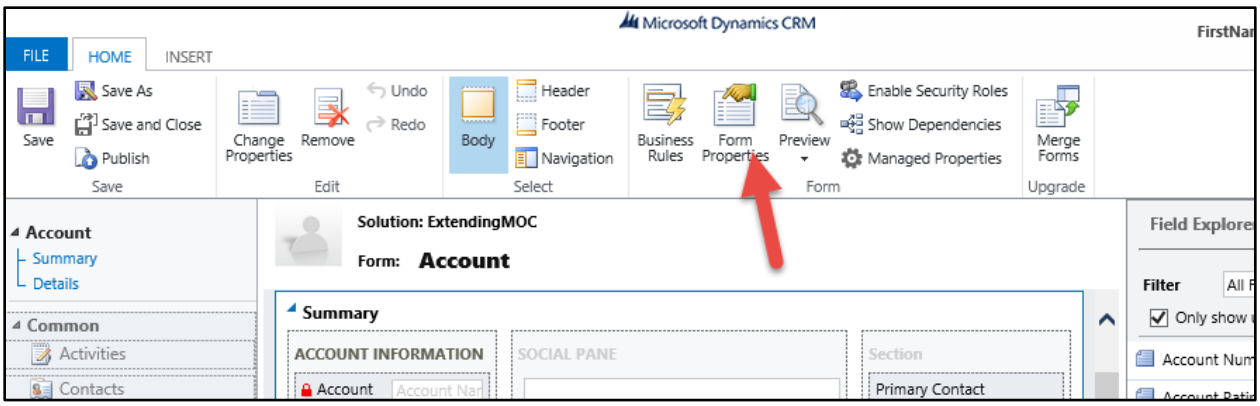
1) Navigate to **Settings | Solutions** and open the **ExtendingMOC** solution.



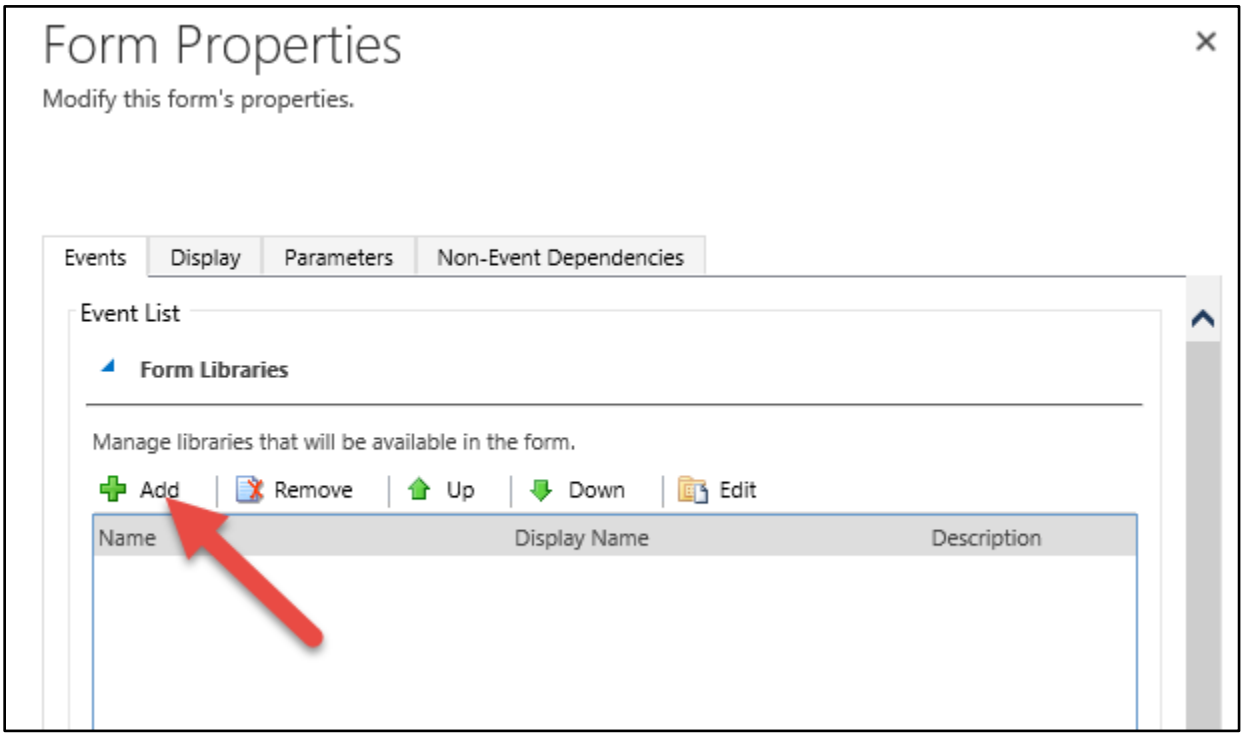
2) Expand **Entities**, expand **Account**, select **Forms**, and open the **Main** account form.



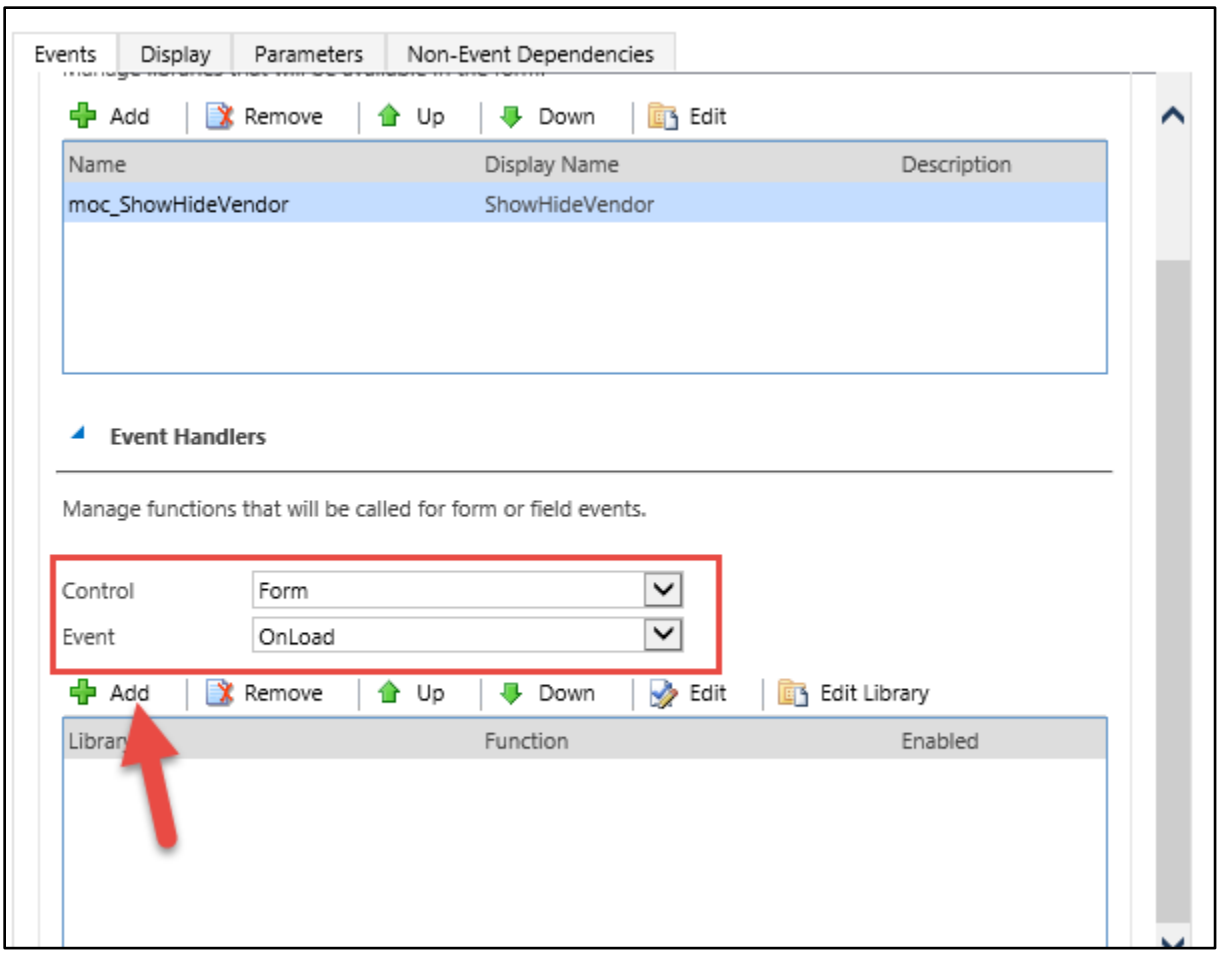
3) Click **Form properties**.



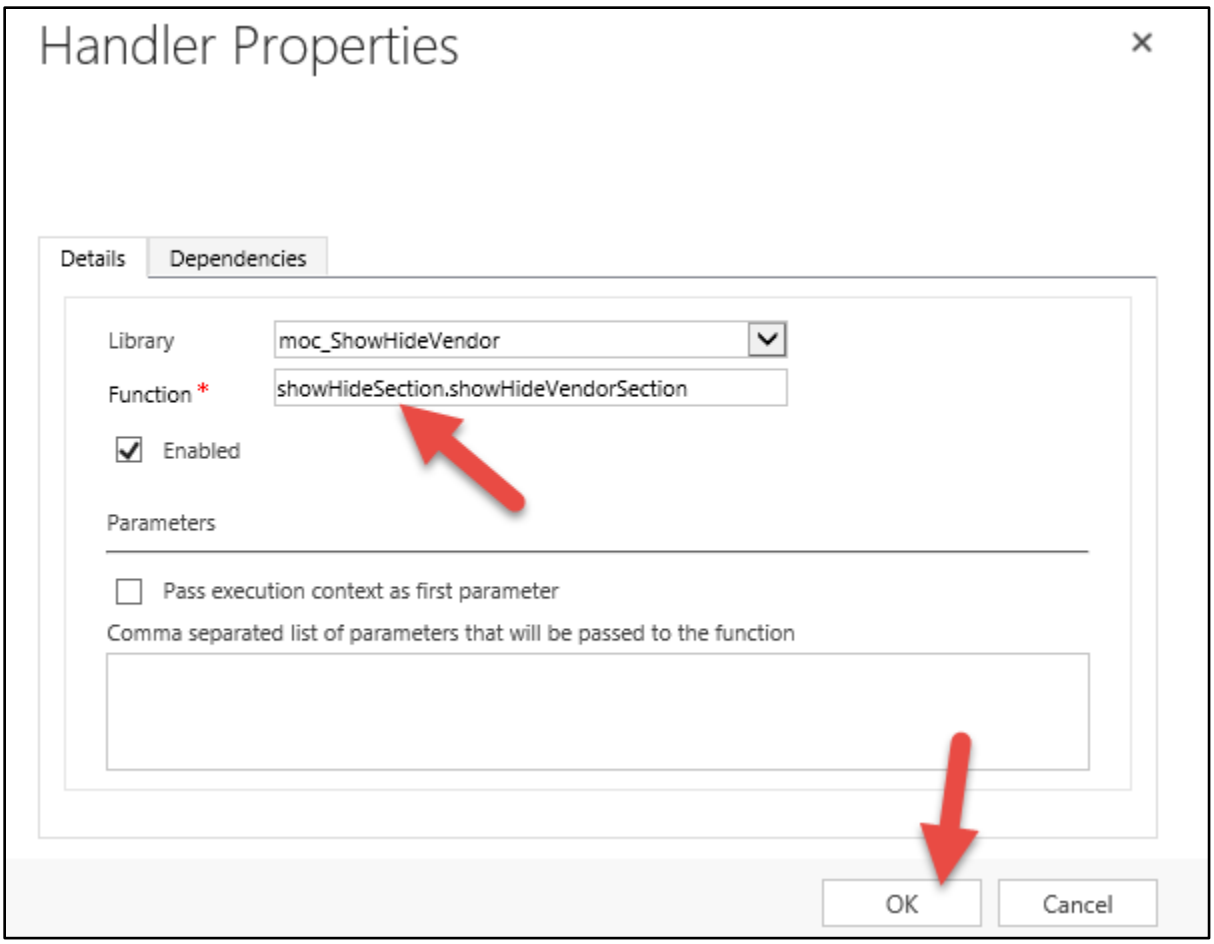
4) Under **Form Libraries**, Click **Add**.



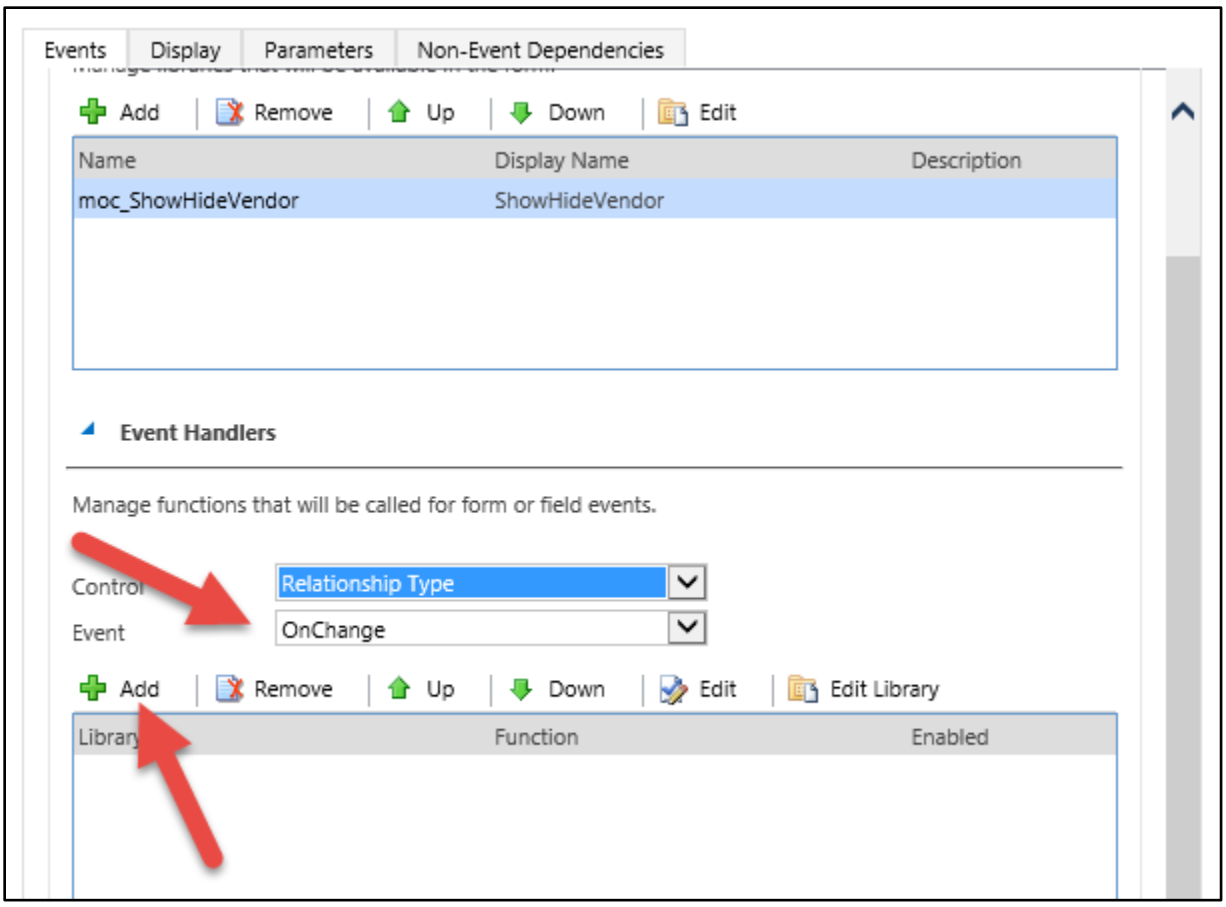
5) In the **Look Up Record** form, type **moc*** in the **Search** textbox.
6) Press Enter.
7) Select **moc_ShowHideVendor** and click **Add**.
8) Make sure **Form** is selected for **Control**, **Onload** is selected for **Event** and click **Add**.



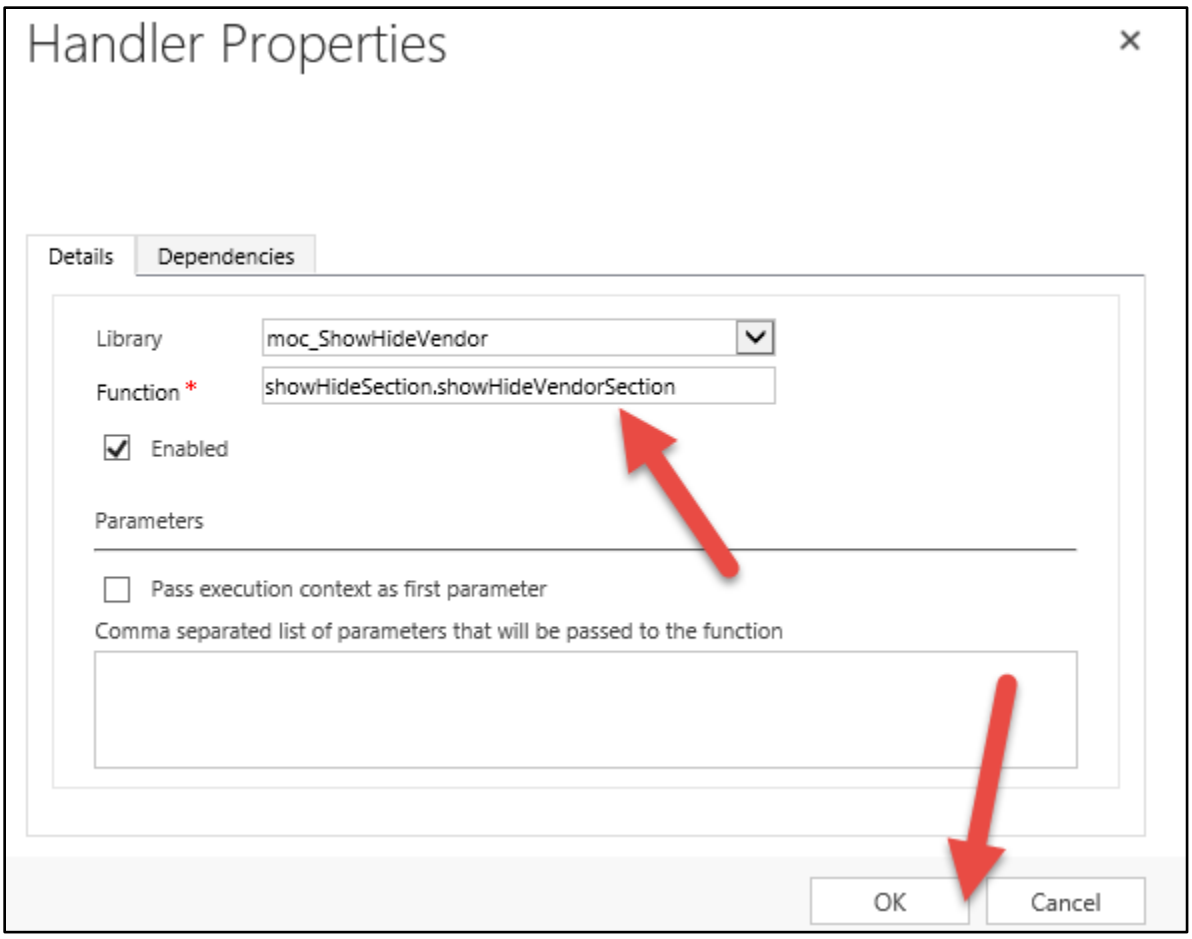
9) Enter **showHideSection.showHideVendorSection** for **Function** and click **OK**.



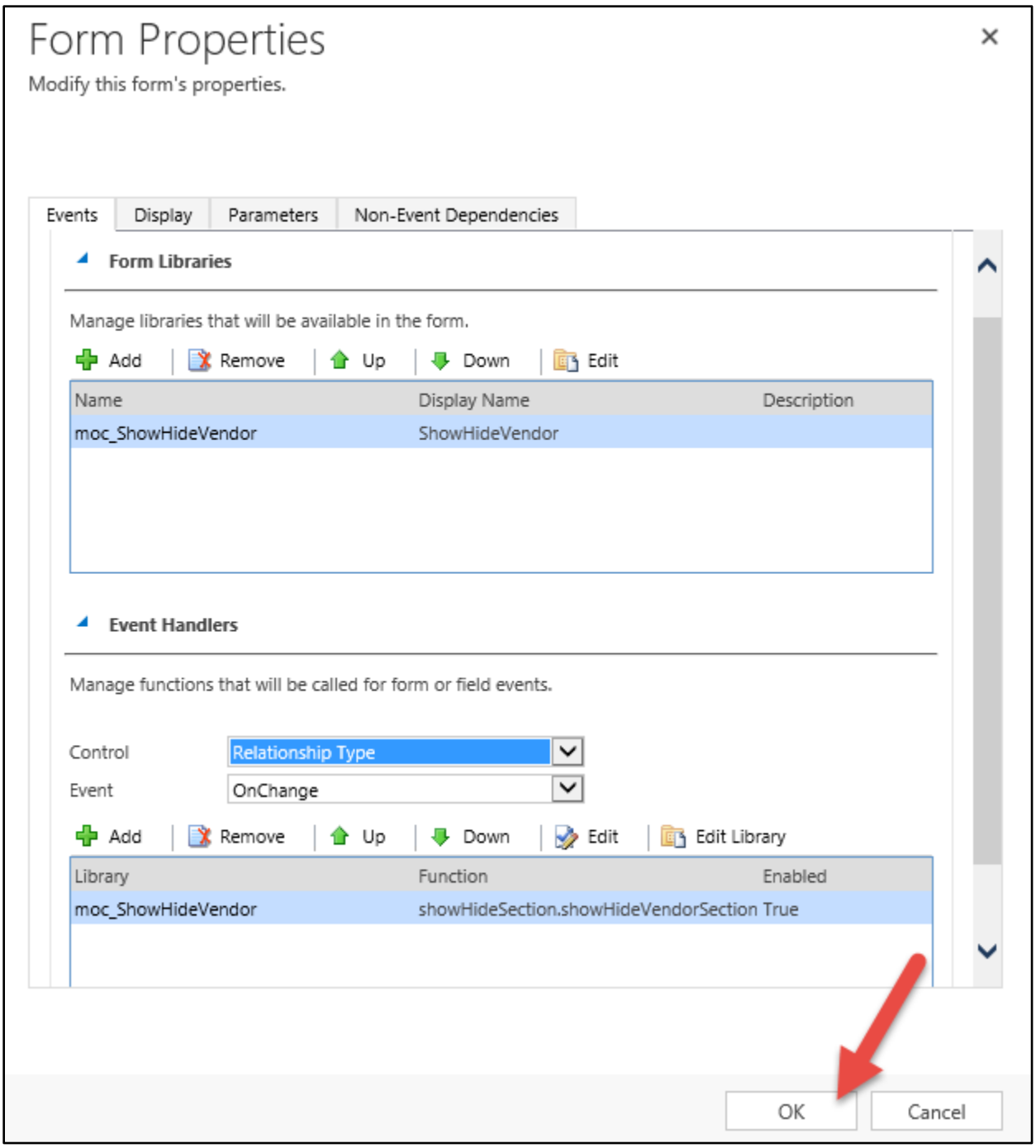
10) In the **Form Properties** form, in the **Event Handlers** section, select **Relationship Type** for the **Control** (Event should show **OnChange**) and click **Add**.



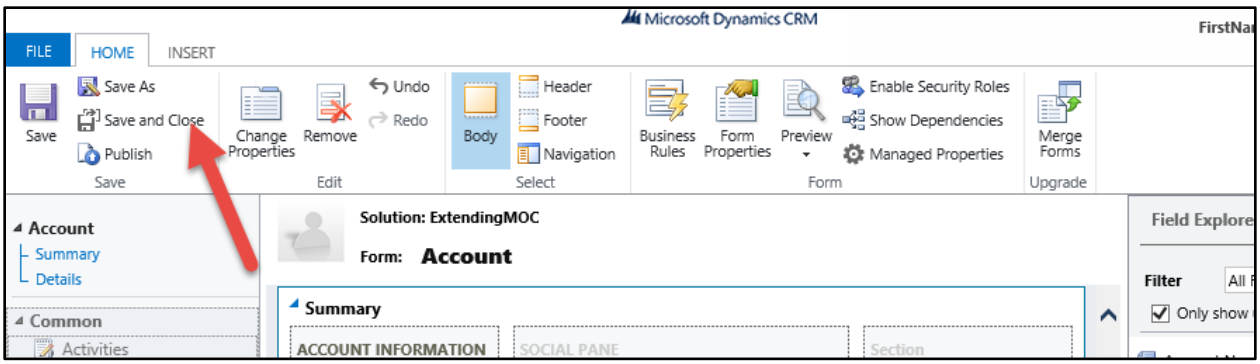
11) Enter **showHideSection.showHideVendorSection** for **Function** and click **OK**.



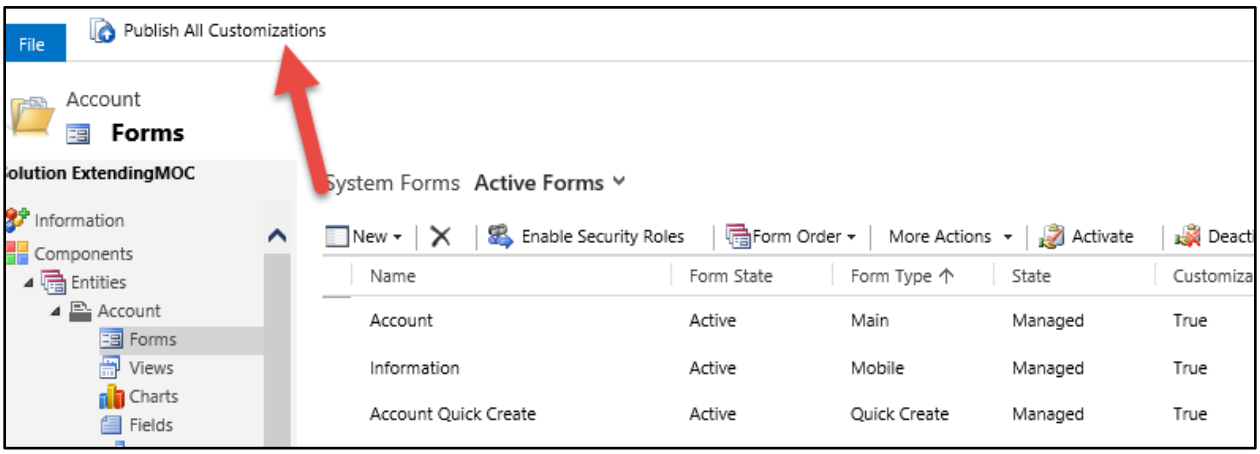
12) Click **OK** to close the **Form Properties** window.



13) Click **Save and Close**.



14) Click **Publish all Customizations**.

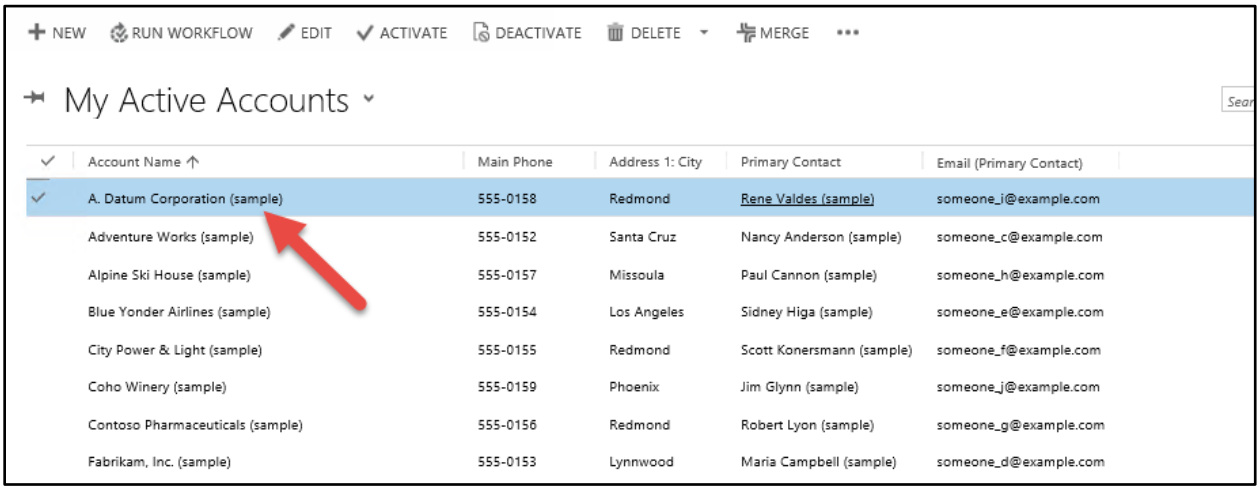


15) Close the **Solution** explorer window.

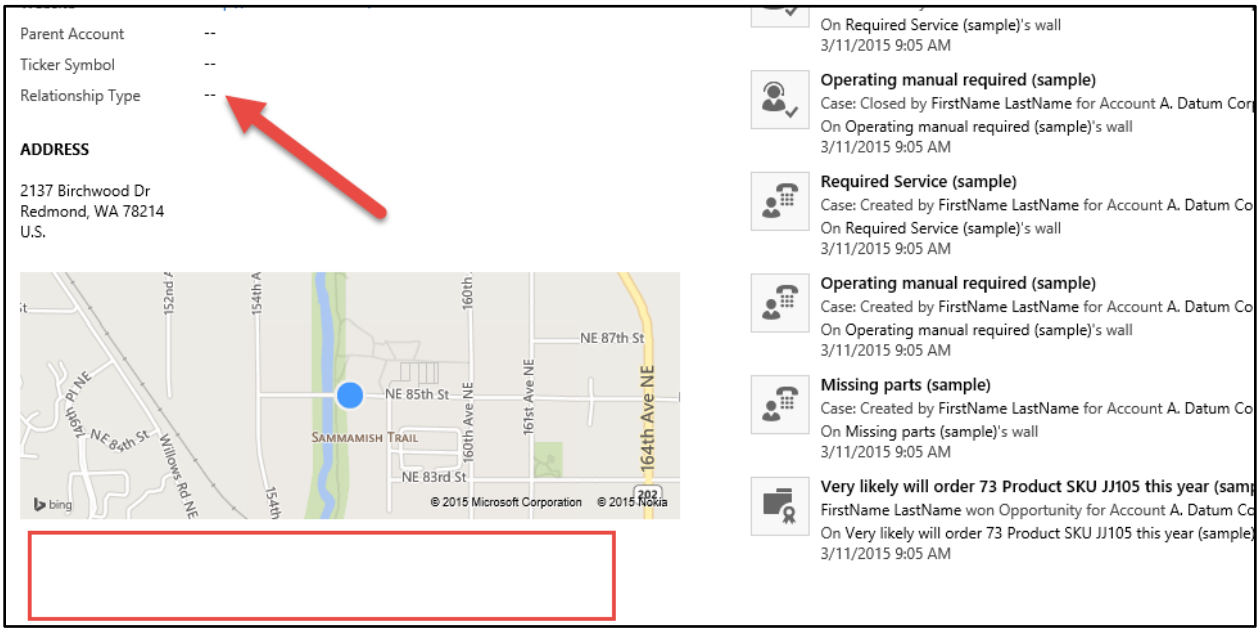
Task 3: Test Your work.

In this task you will test the show hide section script.

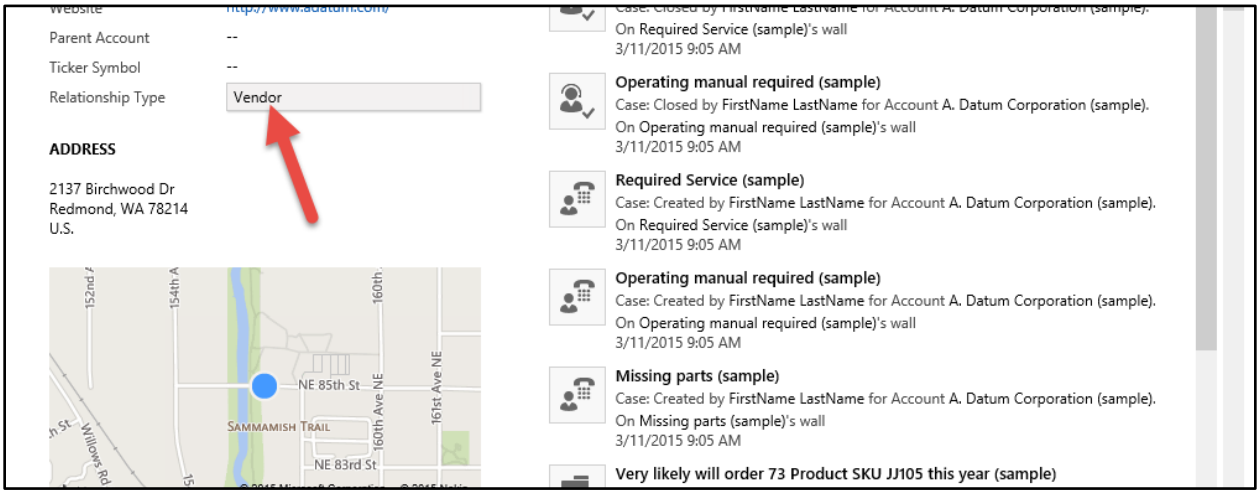
1) Navigate to **Sales | Accounts** and open **A. Datum Corporation**.



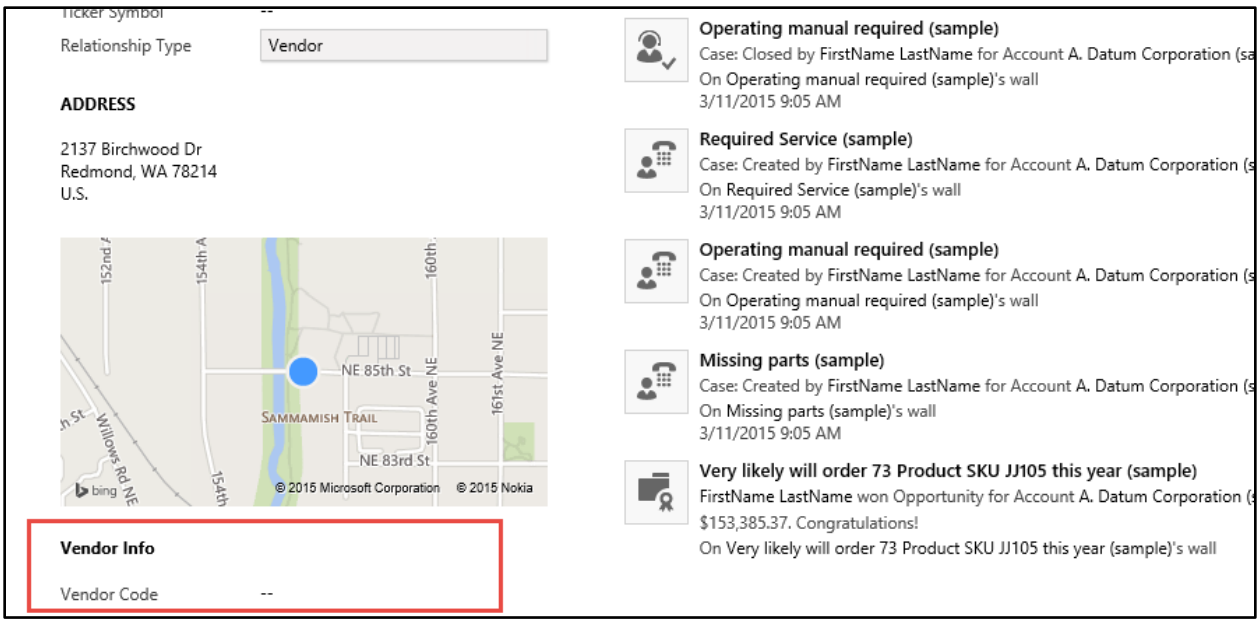
2) The **Relationship type** field is empty by default and the **Vendor Info** section is hidden.



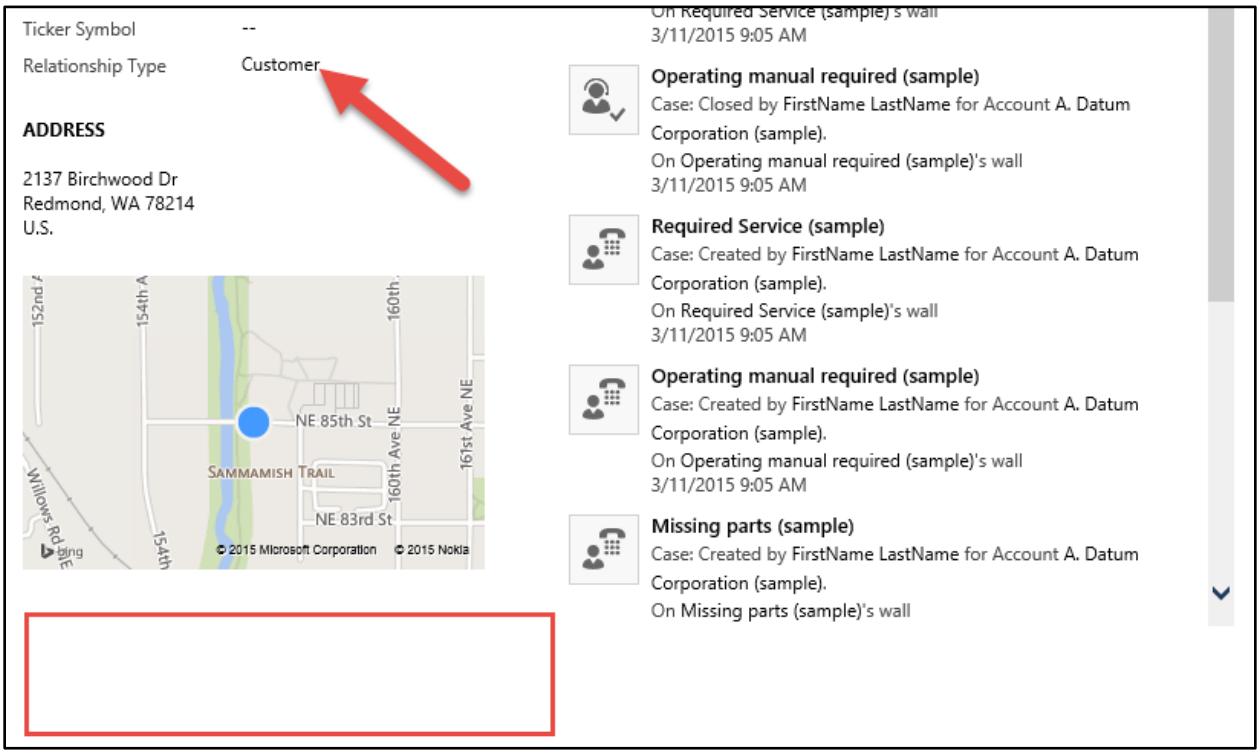
3) Select **Vendor** for **Relationship Type**



4) The **Vendor Info** section is now visible.



5) Change **Relationship Type** to **Customer**
6) The **Vendor Info** section is now hidden again.



Lab 04 – Using the Organization Service

Overview

In this lab you using the Organization service to build several methods that perform a variety of actions on records in Dynamics 365. You will be creating a console application that will connect to CRM and create a new account record. After a record is created you will then build a method that can be used to update an existing account. Before you can perform actions against a CRM instance using early binding, you will need to generate the code classes that will be needed in the project. This lab will guide you through using the CRMSvcUtil.exe tool that is part of the SDK get generate the class file to use in your project.

Finally, you will create a new Opportunity record using late binding, and use the Execute Method to perform more advanced record operations.

What you will learn

- Create a Console application that connects to Dynamics 365
- Create and Update records using Late binding
- Use CrmSvcUtil.exe to generate classes
- Create records using Early binding
- Use the Execute method

What you need to complete this lab

- Visual Studio 2015

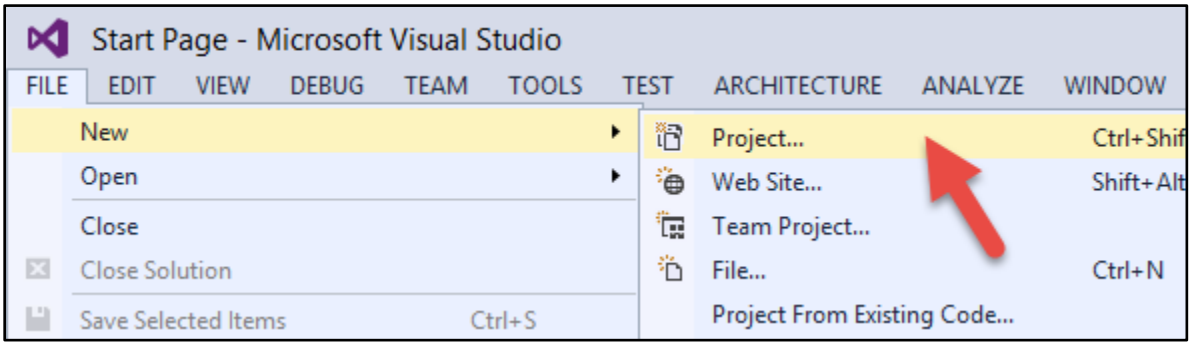
Exercise 1 – Connect to Dynamics 365

In this exercise you will create a project in Console Application called "CRMConnection" using Visual Studio 2015 and connect it your CRM Online Environment. NuGet package Manager will be used to add the appropriate .DLL files needed to successfully interact with a CRM Organization. The project will pass through user credentials of the login user to execute that operations defined in the Project.

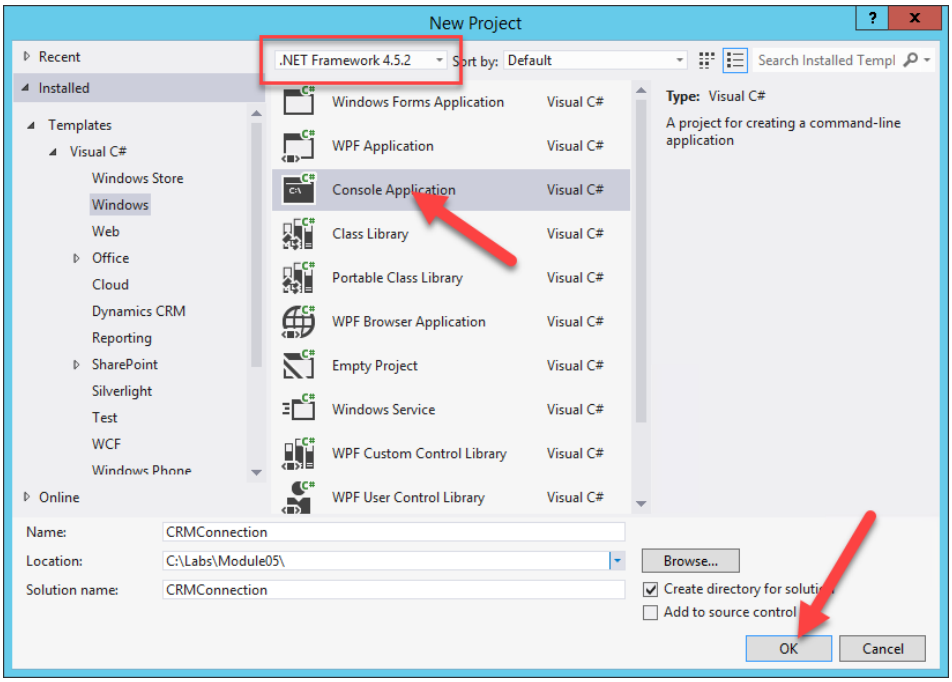
Task 1: Connect to Dynamics 365

In this task you will create a project that connects to your Dynamics 365 organization and passes through user credentials to authenticate. Upon successful execution of the Project the console application will return the ID numbers for the CRM organization and the Business Unit the user belongs to, along with the User ID of the Authenticated user.

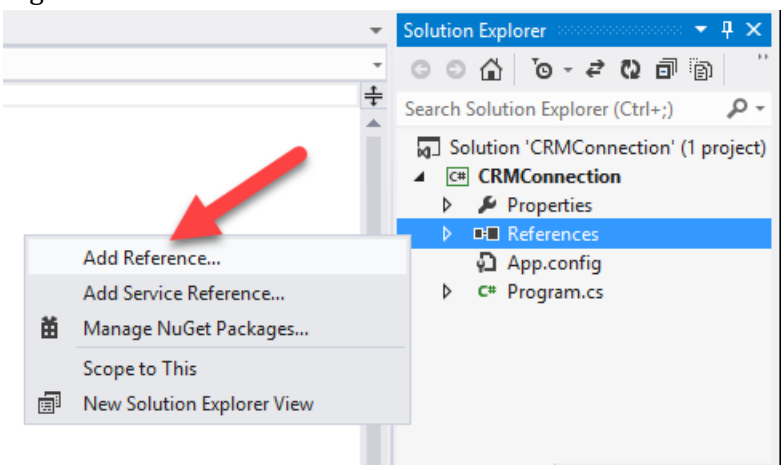
1) Open Visual Studio 2015 and click File | New | Project.



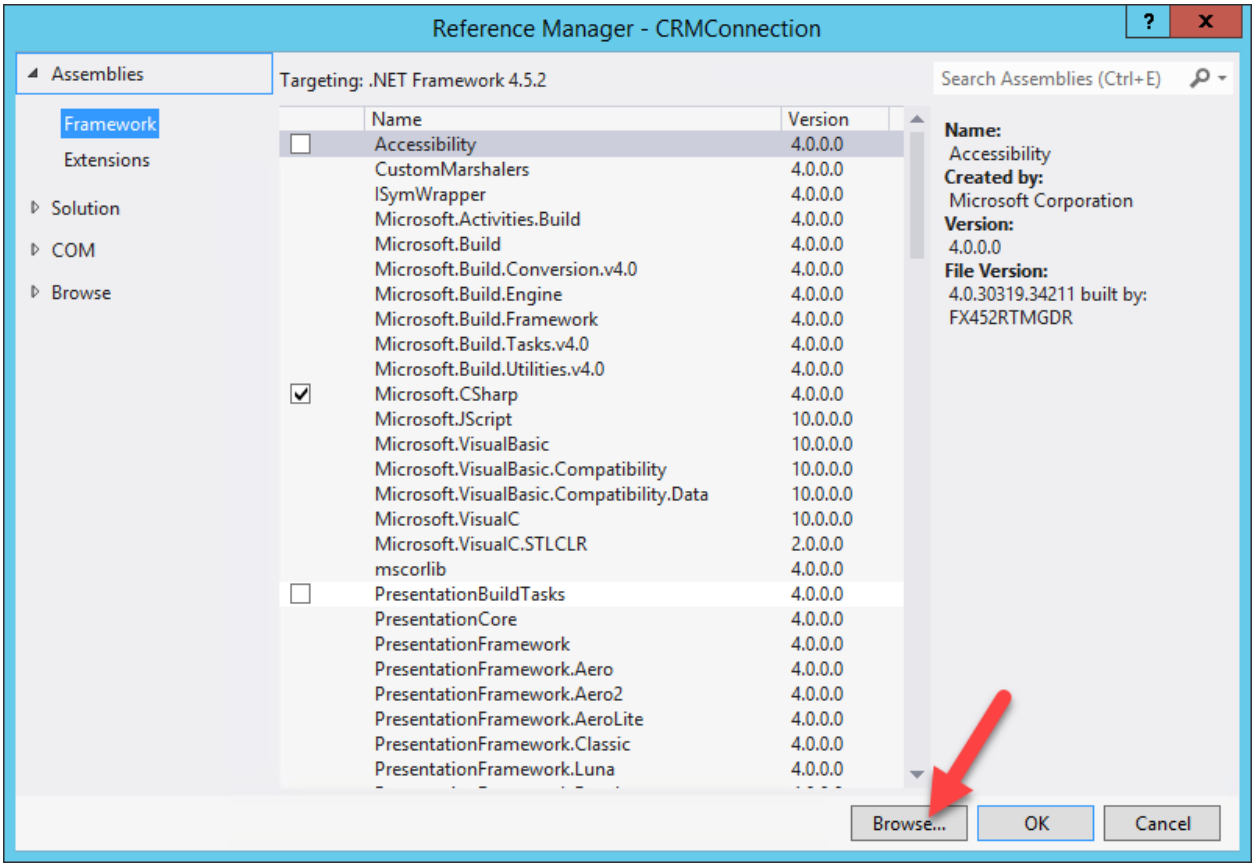
2) Under **Visual C# | Windows**, select **Console Application**, verify that **.NET Framework 4.5.2** is selected, enter **CRMConnection** for **name** and click **OK**.



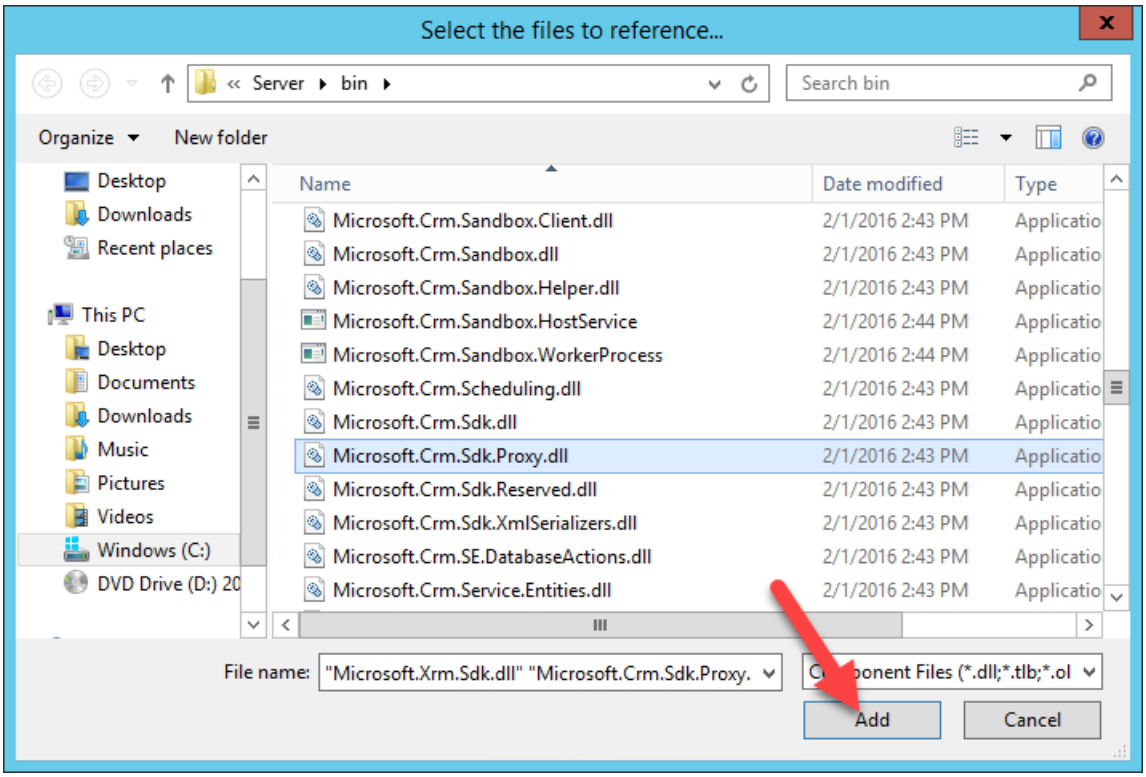
3) Right click the **References** folder and click **Add Reference**.



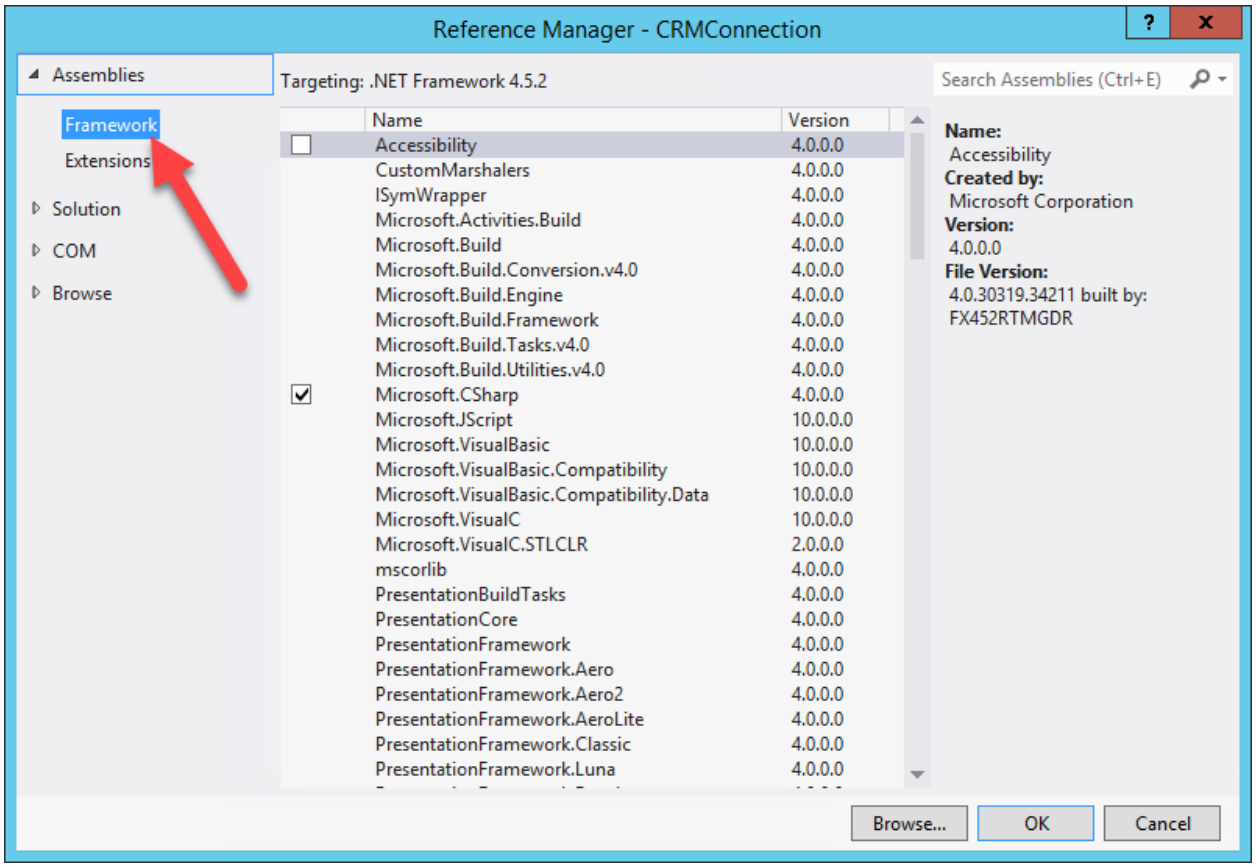
4) In the **Reference Manager** dialog, click **Browse**.



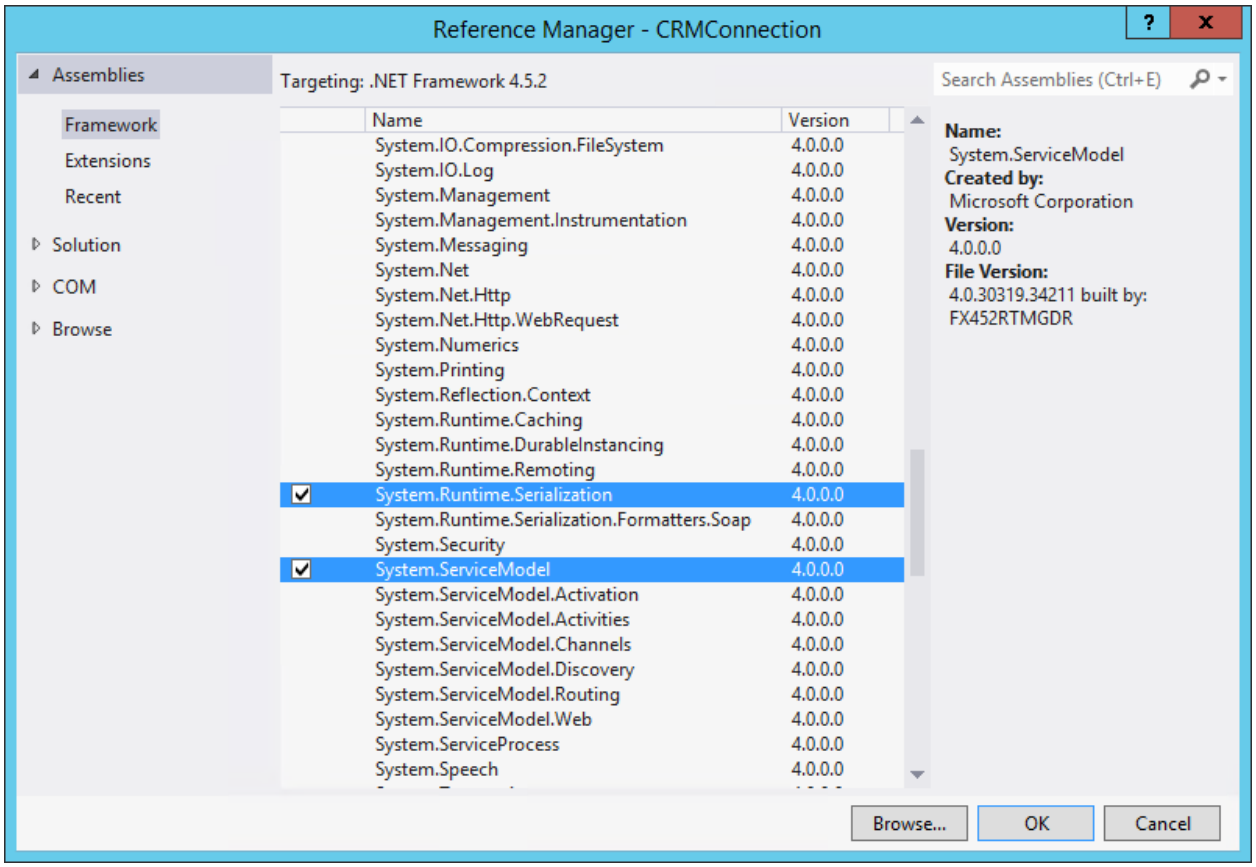
5) Navigate to the **C:\Program Files\Microsoft Dynamics CRM\Server\bin** folder. Select the **Microsoft.Crm.Sdk.Proxy.dll** and **Microsoft.Xrm.Sdk.dll** assemblies, and then click **Add**.



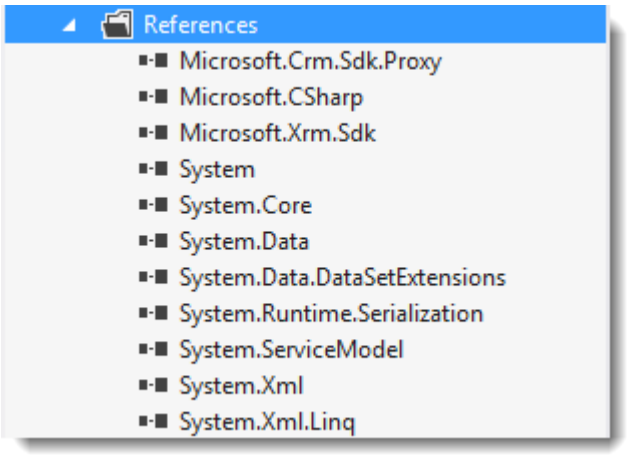
6) In the **Reference Manager** dialog, click **Framework**.



7) Search and select **System.Runtime.Serialization** and **System.ServiceModel** and click **OK**.



8) Make sure that all the required references -shown in the following diagram- were added to a project.



9) Open Program.cs and add the following **using** statements.

```
using Microsoft.Xrm.Sdk.Client;
using Microsoft.Xrm.Sdk;
using Microsoft.Crm.Sdk.Messages;
using System.ServiceModel.Description;
```

```
Program.cs*
CRMConnection.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Microsoft.Xrm.Sdk.Client;
using Microsoft.Xrm.Sdk;
using Microsoft.Crm.Sdk.Messages;
using System.ServiceModel.Description;
```

10) In the **Main** static method, add the following code to connect to the **Organization Service** WCF service.

```
ClientCredentials credentials = new ClientCredentials();
credentials.Windows.ClientCredential =
    System.Net.CredentialCache.DefaultNetworkCredentials;
OrganizationServiceProxy _serviceProxy =
   new OrganizationServiceProxy(new Uri(
"http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
                null, credentials, null);
```

```
CRMConnection | CRMConnection.Program | Main(string[] args)
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   using Microsoft.Xrm.Sdk.Client;
8   using Microsoft.Xrm.Sdk;
9   using Microsoft.Crm.Sdk.Messages;
10  using System.ServiceModel.Description;
11
12  namespace CRMConnection
13  {
        0 references
14      class Program
15      {
            0 references
16          static void Main(string[] args)
17          {
18              ClientCredentials credentials = new ClientCredentials();
19              credentials.Windows.ClientCredential =
20                  System.Net.CredentialCache.DefaultNetworkCredentials;
21              OrganizationServiceProxy _serviceProxy =
22                new OrganizationServiceProxy(new Uri(
23              "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                              null, credentials, null);
25
26          }
27      }
28  }
29
```

11) Create new instance of Who Am I Request and execute Who Am I Request and convert the result to Who Am I Response.

```
WhoAmIRequest whoAmI = new WhoAmIRequest();
WhoAmIResponse userLoggedId = (WhoAmIResponse) _serviceProxy.Execute(whoAmI);
```

12) Check if the result is not null and display Organization and user information.

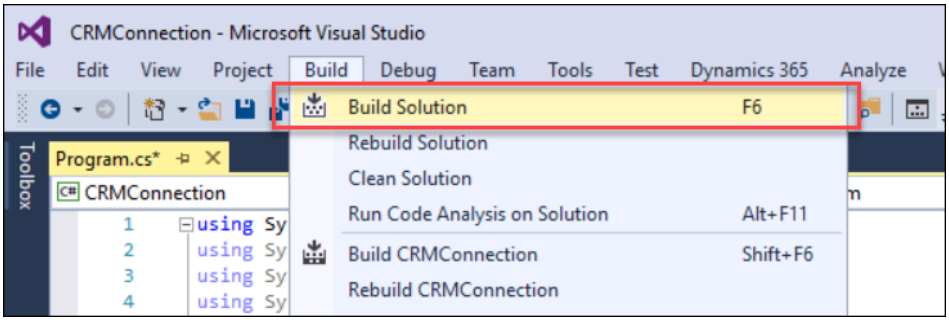
```
if (userLoggedId != null)
{
    Console.WriteLine(String.Format(
      "Organization ID: {0}\nBusiness Unit ID: {1}\nuser ID:{2}",
      userLoggedId.OrganizationId, userLoggedId.BusinessUnitId,
      userLoggedId.UserId));
    Console.ReadLine();
}
```

Program.cs

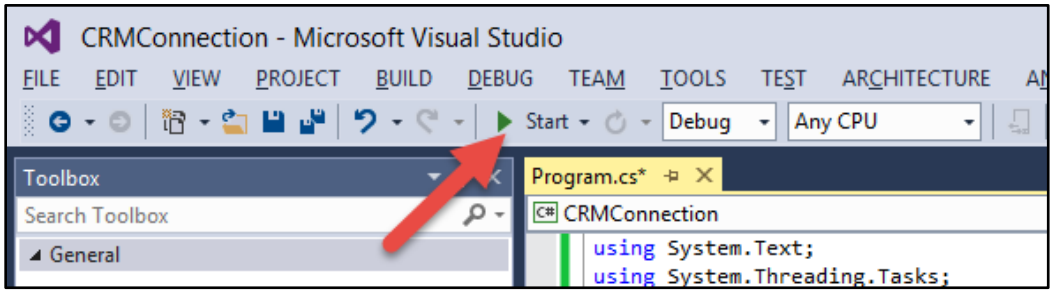
CRMConnection | CRMConnection.Program | Main(string[] args)

```
13  {
        0 references
14      class Program
15      {
            0 references
16          static void Main(string[] args)
17          {
18              ClientCredentials credentials = new ClientCredentials();
19              credentials.Windows.ClientCredential =
20                  System.Net.CredentialCache.DefaultNetworkCredentials;
21              OrganizationServiceProxy _serviceProxy =
22                new OrganizationServiceProxy(new Uri(
23             "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                            null, credentials, null);
25
26              WhoAmIRequest whoAmI = new WhoAmIRequest();
27              WhoAmIResponse userLoggedId = (WhoAmIResponse)_serviceProxy.Execute(whoAmI);
28
29              if (userLoggedId != null)
30              {
31                  Console.WriteLine(String.Format(
32                    "Organization ID: {0}\nBusiness Unit ID: {1}\nuser ID:{2}",
33                    userLoggedId.OrganizationId, userLoggedId.BusinessUnitId,
34                    userLoggedId.UserId));
35                  Console.ReadLine();
36              }
37          }
38      }
39  }
40
```

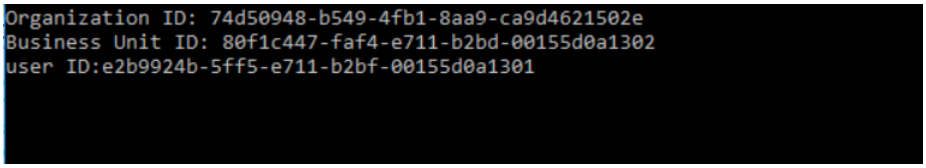
13) Click **Build | Build Solution**.



14) Run the project.



15) Console with display the result.



16) Press enter to close.

Exercise 2 – Use Late Binding

In this exercise you will perform two tasks using late binding. The first task will be to create a new Account record in Dynamics 365, and populate the Name and Address 1 fields for the record.

Task 1: Create Record Using Late Bound

In this task you will create a new Account record, and populate the Name and Address 1 fields for the record.

1) Open **Program.cs** if it is not open.
2) After the **Main** method, add the following static method that will build and return an account record.

```
static Entity BuildAccount()
{
    Entity newAccountRecord = new Entity("account");
    newAccountRecord["name"] = "Microsoft";
    newAccountRecord["address1_line1"] = "One Microsoft Way";

    return newAccountRecord;
}
```

```
CRMConnection | CRMConnection.Program | BuildAccount()
0 references
16  static void Main(string[] args)
17  {
18      ClientCredentials credentials = new ClientCredentials();
19      credentials.Windows.ClientCredential =
20          System.Net.CredentialCache.DefaultNetworkCredentials;
21      OrganizationServiceProxy _serviceProxy =
22        new OrganizationServiceProxy(new Uri(
23     "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                     null, credentials, null);
25
26      WhoAmIRequest whoAmI = new WhoAmIRequest();
27      WhoAmIResponse userLoggedId = (WhoAmIResponse)_serviceProxy.Execute(whoAmI);
28
29      if (userLoggedId != null)
30      {
31          Console.WriteLine(String.Format(
32            "Organization ID: {0}\nBusiness Unit ID: {1}\nuser ID:{2}",
33            userLoggedId.OrganizationId, userLoggedId.BusinessUnitId, userLoggedId.UserId));
34          Console.ReadLine();
35      }
36  }
37
    0 references
38  static Entity BuildAccount()
39  {
40      Entity newAccountRecord = new Entity("account");
41      newAccountRecord["name"] = "Microsoft";
42      newAccountRecord["address1_line1"] = "One Microsoft Way";
43
44      return newAccountRecord;
45  }
```

3) In the **Main** method, after creating the instance of the **OrganizationServiceProxy** class, call the **BuildAccount** method and assign the result to an entity instance.

```
Entity newAccount = BuildAccount();
```

4) Call the **Create** method by passing the new account record.

```
Guid createdRecordId = _serviceProxy.Create(newAccount);
```

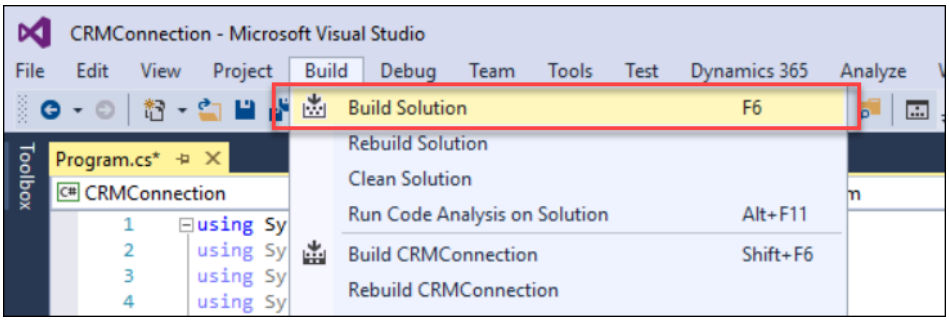
5) Display the GUID value of the new account.

```
Console.WriteLine(String.Format("Account GUID: {0}", createdRecordId ));
```

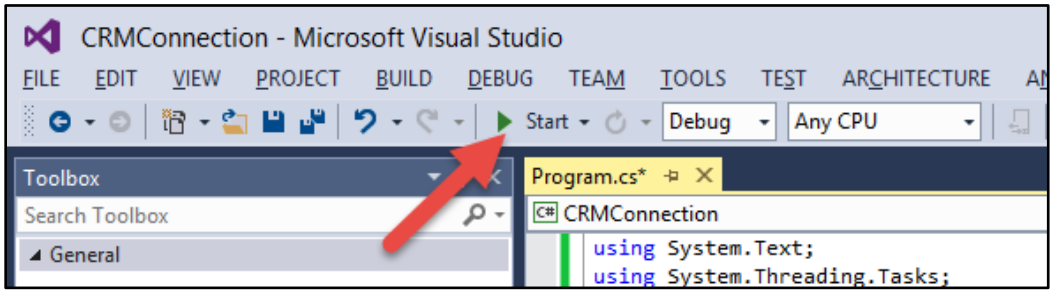
Your completed code should look as follows:

```
Program.cs*
CRMConnection | CRMConnection.Program | Main(string[] args)
0 references
16  static void Main(string[] args)
17  {
18      ClientCredentials credentials = new ClientCredentials();
19      credentials.Windows.ClientCredential =
20          System.Net.CredentialCache.DefaultNetworkCredentials;
21      OrganizationServiceProxy _serviceProxy =
22        new OrganizationServiceProxy(new Uri(
23     "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                    null, credentials, null);
25
26      Entity newAccount = BuildAccount();
27      Guid createdRecordId = _serviceProxy.Create(newAccount);
28      Console.WriteLine(String.Format("Account GUID: {0}", createdRecordId));
29
30      WhoAmIRequest whoAmI = new WhoAmIRequest();
31      WhoAmIResponse userLoggedId = (WhoAmIResponse)_serviceProxy.Execute(whoAmI);
32
33      if (userLoggedId != null)
34      {
35          Console.WriteLine(String.Format(
36            "Organization ID: {0}\nBusiness Unit ID: {1}\nuser ID:{2}",
37            userLoggedId.OrganizationId, userLoggedId.BusinessUnitId, userLoggedId.UserId));
38          Console.ReadLine();
39      }
40
41  }
42
1 reference
43  static Entity BuildAccount()
44  {
45      Entity newAccountRecord = new Entity("account");
```

6) Click **Build | Build Solution**.



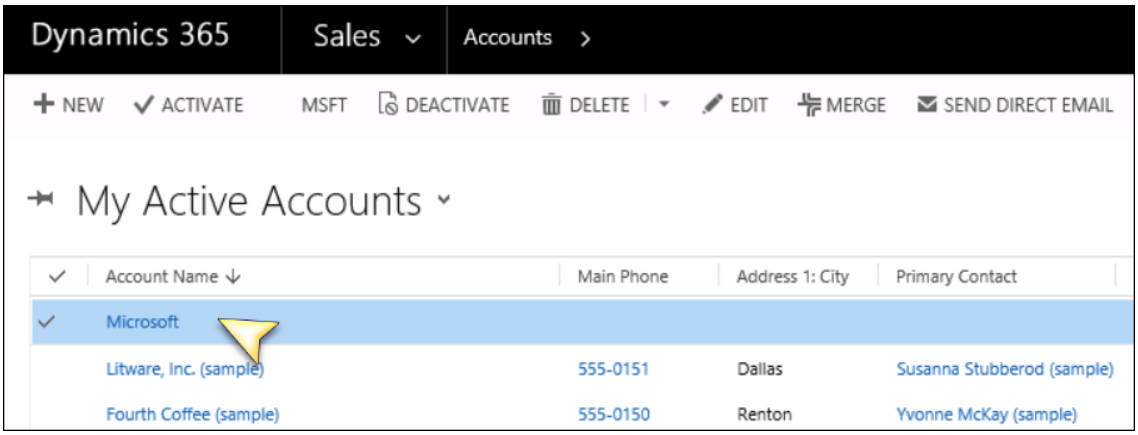
7) Run the project.



8) Console with display the result.

```
Account GUID: e714155a-abfc-e711-b2c8-00155d018000
Organization ID: 74d50948-b549-4fb1-8aa9-ca9d4621502e
Business Unit ID: 80f1c447-faf4-e711-b2bd-00155d0a1302
user ID:e2b9924b-5ff5-e711-b2bf-00155d0a1301
```

9) Press enter to close.
10) In Dynamics 365, navigate to **Sales | Accounts** and make sure the account record was created.



Exercise 3 – Using Early Bound Entities

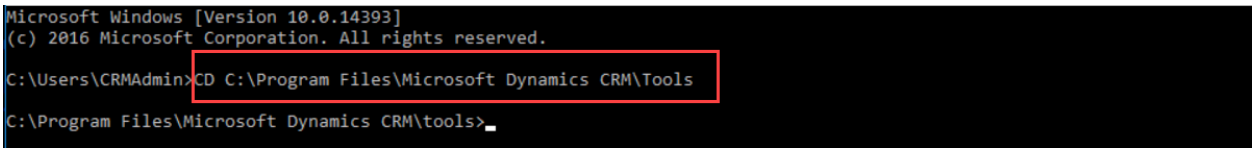
Using early binding in your projects provides you with IntelliSense and type checking. To accomplish this, you will generate a file that contains classes for all the entities in your CRM organization. As you are developing your project, any errors related to entities, relationships, etc. will be identified. In this exercise you will be using the CRMSvcUtil.exe file to generate the early bound classes for you CRM organization and then using them to create a new Contact record by using Early Binding.

Task 1: Generate Type Classes

In this task generate type classes.

1) Start Command Prompt.
2) Change directory to the **C:\Program Files\Microsoft Dynamics CRM\Tools** folder.

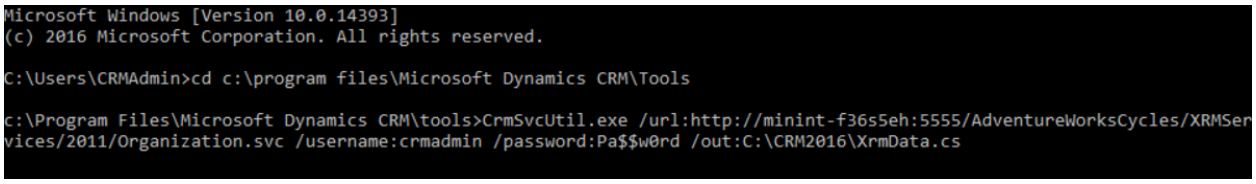
```
CD C:\Program Files\Microsoft Dynamics CRM\Tools
```



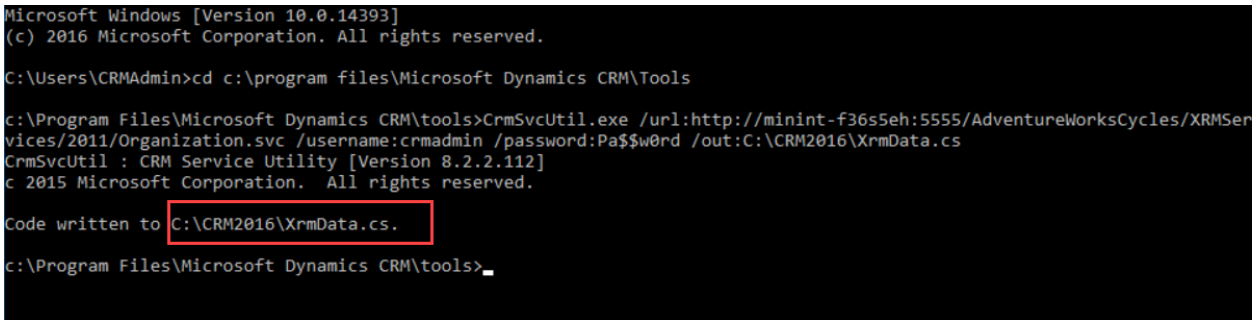
Note: This lab assumes that there is a folder named **CRM2016** located in your **C:** drive.

3) Type in one single line the command below.

```
CrmSvcUtil.exe
/url:http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc
/username:crmadmin /password:Pa$$w0rd /out:C:\CRM2016\XrmData.cs
```

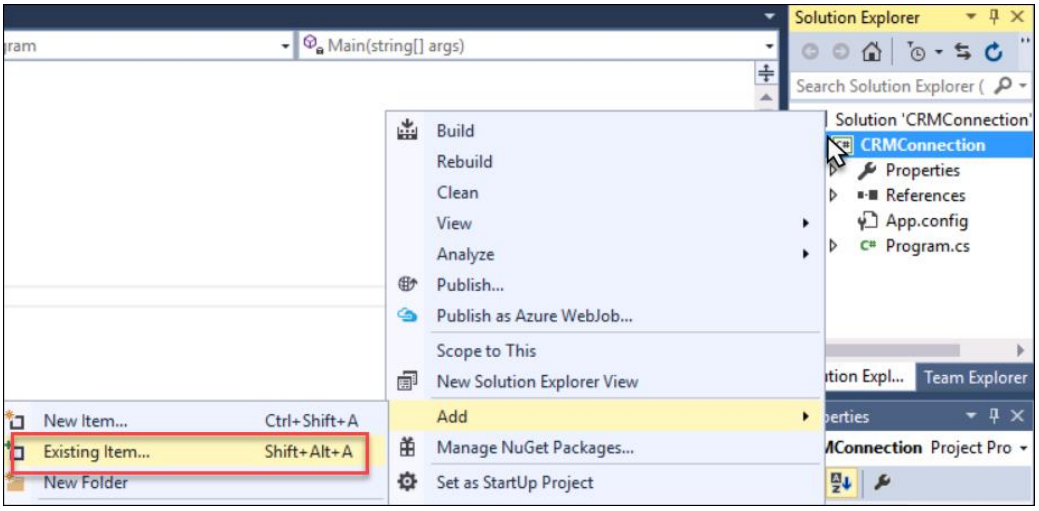


4) Press Enter.
5) Wait for the Type Classes to be generated. The code will be saved in the **C:\CRM2016** folder.

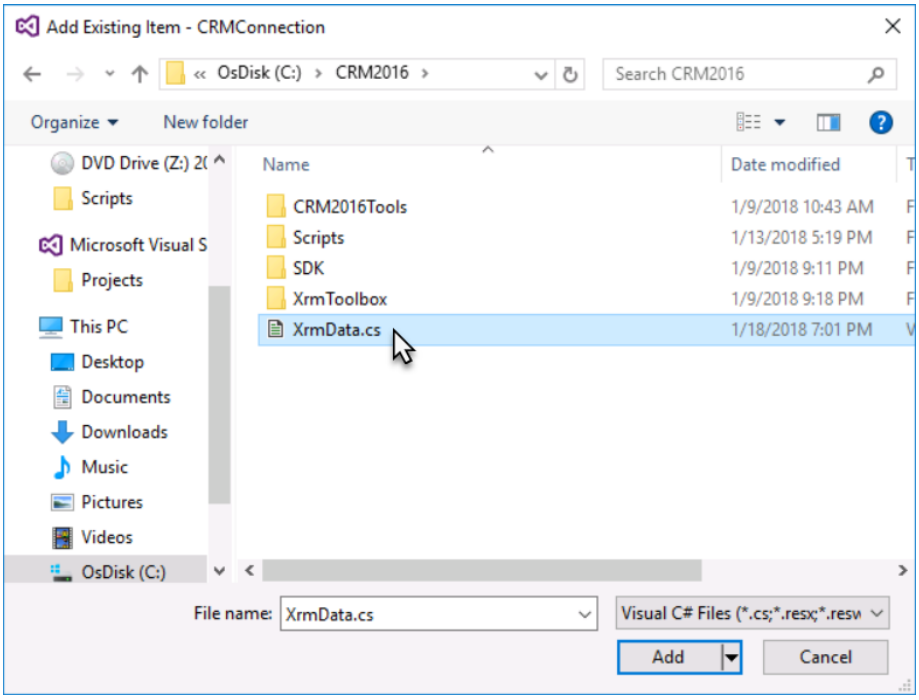


6) Close Command Prompt.

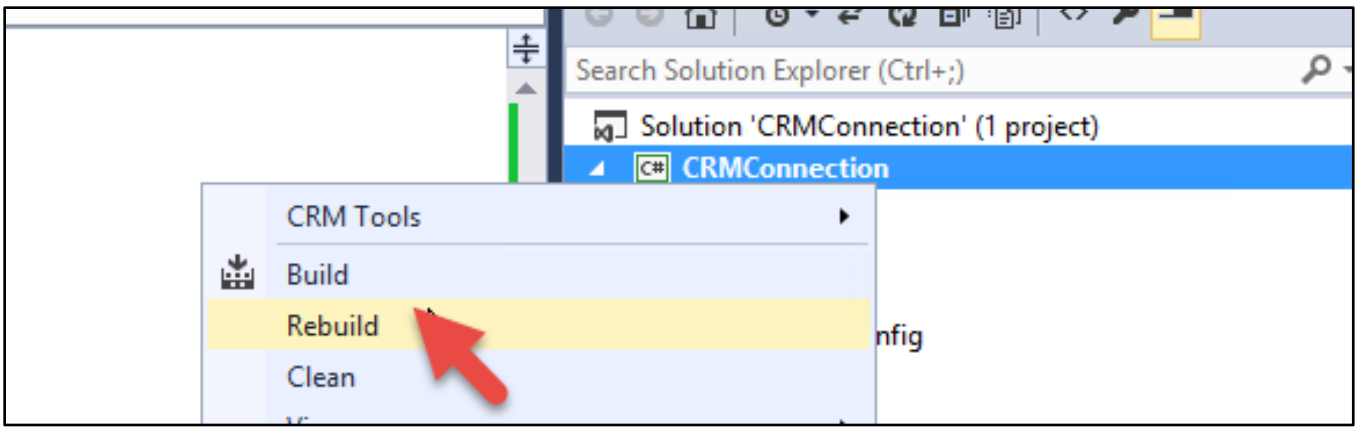
7) Go back to Visual Studio, right click the **CRMConnection** project and click **Add | Existing Item**.



8) Browse to the **C:\CRM2016** folder, select **XrmData.cs** and click **Add**.



9) Right click the project, click **Rebuild**, and make sure the project builds successfully.

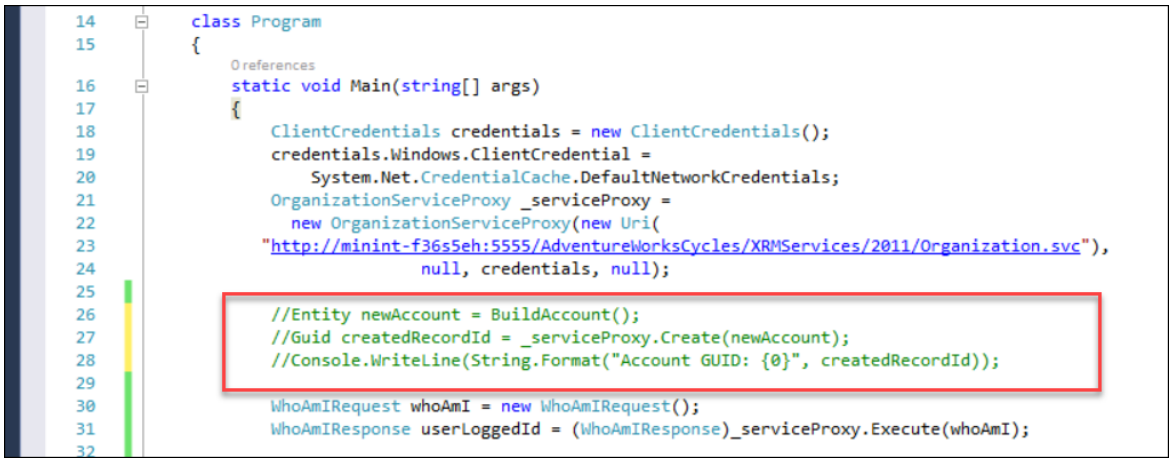


Task 2: Create a Case Using Early Bound

In this task you will create new contact using the generated early bound classes.

1) Go back to Visual Studio, open the **Program.cs** file if it is not open.
2) Go to the **Main** method and comment out the following lines.

```
//Entity newAccount = BuildAccount();
//Guid createdRecordId = _serviceProxy.Create(newAccount);
//Console.WriteLine(String.Format("Account GUID: {0}", createdRecordId ))
```



```
14  class Program
15  {
        0 references
16      static void Main(string[] args)
17      {
18          ClientCredentials credentials = new ClientCredentials();
19          credentials.Windows.ClientCredential =
20              System.Net.CredentialCache.DefaultNetworkCredentials;
21          OrganizationServiceProxy _serviceProxy =
22            new OrganizationServiceProxy(new Uri(
23         "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                        null, credentials, null);
25
26          //Entity newAccount = BuildAccount();
27          //Guid createdRecordId = _serviceProxy.Create(newAccount);
28          //Console.WriteLine(String.Format("Account GUID: {0}", createdRecordId));
29
30          WhoAmIRequest whoAmI = new WhoAmIRequest();
31          WhoAmIResponse userLoggedId = (WhoAmIResponse)_serviceProxy.Execute(whoAmI);
32
```

3) After the comments add the following to enable early bound entity types on the service proxy.

```
_serviceProxy.EnableProxyTypes();
```

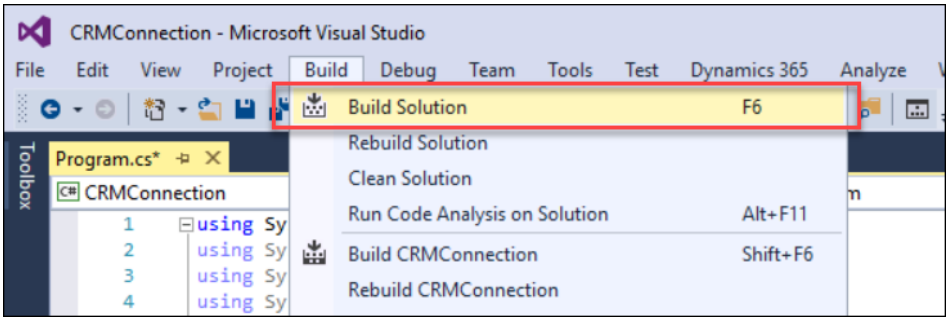
```
Program.cs*
CRMConnection | CRMConnection.Program | Main(string[] args)
13  {
    0 references
14  class Program
15  {
        0 references
16      static void Main(string[] args)
17      {
18          ClientCredentials credentials = new ClientCredentials();
19          credentials.Windows.ClientCredential =
20              System.Net.CredentialCache.DefaultNetworkCredentials;
21          OrganizationServiceProxy _serviceProxy =
22            new OrganizationServiceProxy(new Uri(
23        "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                        null, credentials, null);
25
26          //Entity newAccount = BuildAccount();
27          //Guid createdRecordId = _serviceProxy.Create(newAccount);
28          //Console.WriteLine(String.Format("Account GUID: {0}", createdRecordId));
29
30          _serviceProxy.EnableProxyTypes();
31
```

4) Add the following lines of code to create a new Contact record.

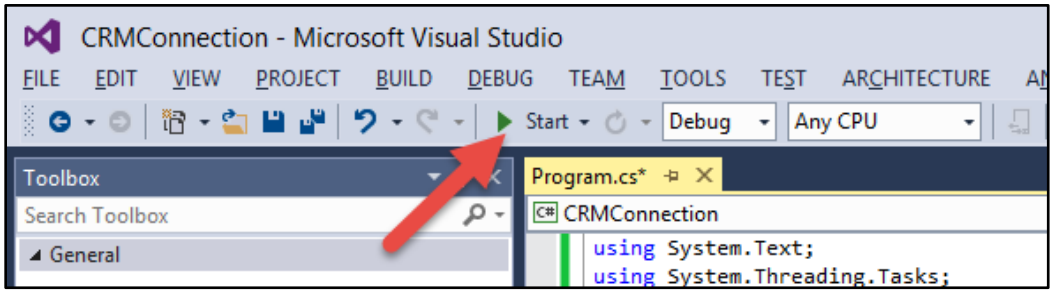
```
Contact newContact = new Contact;
NewContact.FirstName = "Alan";
newContact.LastName = "Jackson";
newContact.Telephone1 = "555-9999";
Guid createdContactId = _serviceProxy.Create(newContact);
Console.WriteLine(String.Format("Contact GUID: {0}", createdContactId ))
```

```
Program.cs
CRMConnection | CRMConnection.Program | Main(string[] args)
15  {
    0 references
16  static void Main(string[] args)
17  {
18      ClientCredentials credentials = new ClientCredentials();
19      credentials.Windows.ClientCredential =
20          System.Net.CredentialCache.DefaultNetworkCredentials;
21      OrganizationServiceProxy _serviceProxy =
22        new OrganizationServiceProxy(new Uri(
23    "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                    null, credentials, null);
25
26      //Entity newAccount = BuildAccount();
27      //Guid createdRecordId = _serviceProxy.Create(newAccount);
28      //Console.WriteLine(String.Format("Account GUID: {0}", createdRecordId));
29
30      _serviceProxy.EnableProxyTypes();
31
32      Contact newContact = new Contact();
33      newContact.FirstName = "Alan";
34      newContact.LastName = "Jackson";
35      newContact.Telephone1 = "555-9999";
36      Guid createdContactId = _serviceProxy.Create(newContact);
37      Console.WriteLine(String.Format("Contact GUID: {0}", createdContactId));
38
39      WhoAmIRequest whoAmI = new WhoAmIRequest();
40      WhoAmIResponse userLoggedId = (WhoAmIResponse)_serviceProxy.Execute(whoAmI);
```

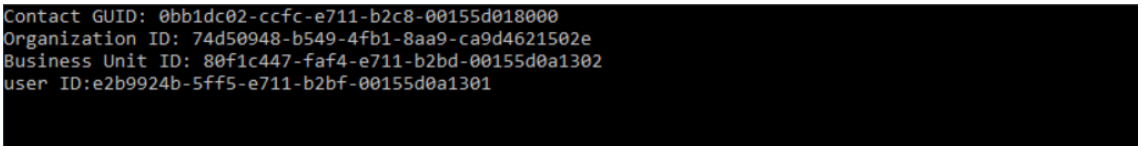
5) Click **Build | Build Solution**.



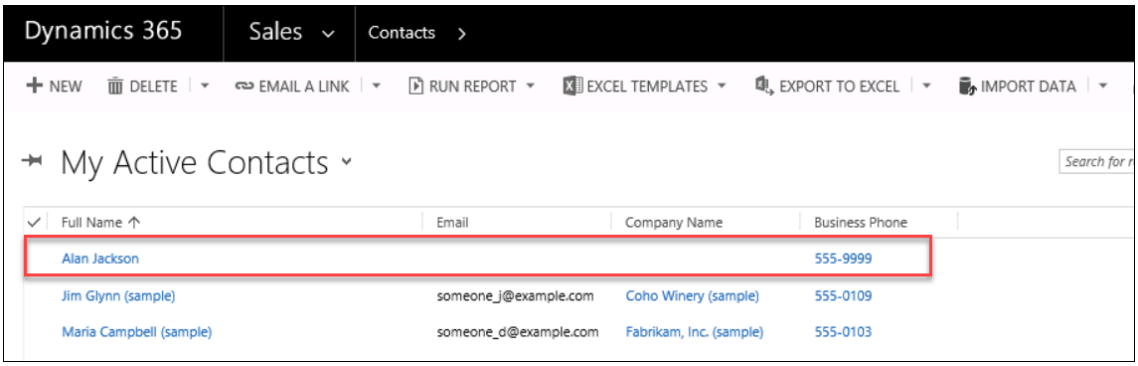
6) Run the project.



7) Console with display the result.



8) Press enter to close the console application.
9) In Dynamics 365, navigate to **Sales | Contacts** and make sure the contact record was created.



Lab 05 – Querying Dynamics 365 Data

Overview

In this lab you be querying Dynamics 365 opportunities using 2 different query types available in CRM. The end result for each query will be the same, but you be executing them using different methods. The intent of the lab is to show you how to accomplish a task using any of the methods. Each exercise will create a basic Query that will display estimated revenue for Opportunities.

What you will learn

- How to use Query Expression
- How to use FetchXML

What you need to complete this lab

- Microsoft Visual Studio 2015

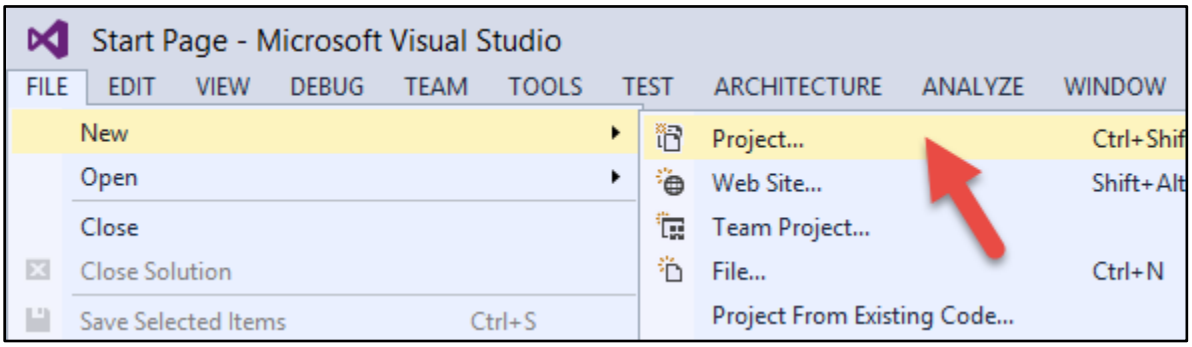
Exercise 1 – Query Expression

In this exercise, you will use Query Expression to create a basic query that will display estimated revenue for Opportunities.

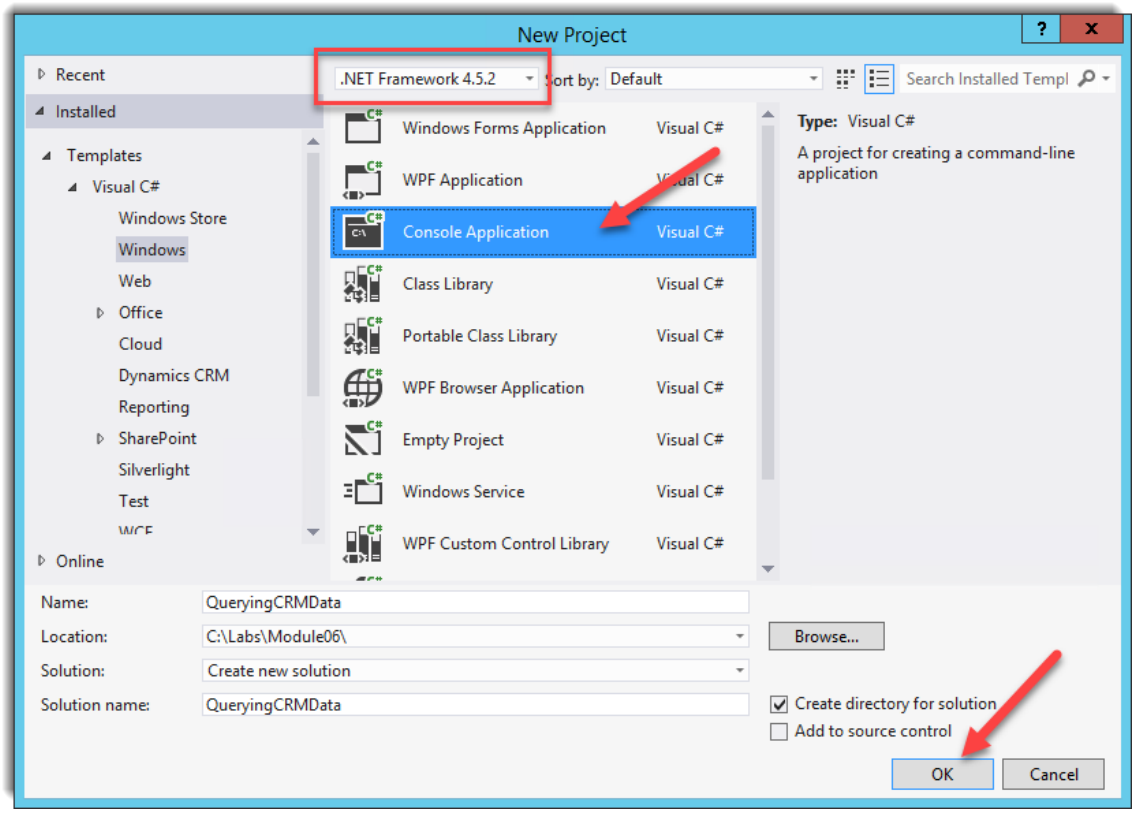
Task 1: Write a basic Query Expression

In this task, you will write a basic Query Expression to display estimated revenue for Opportunities.

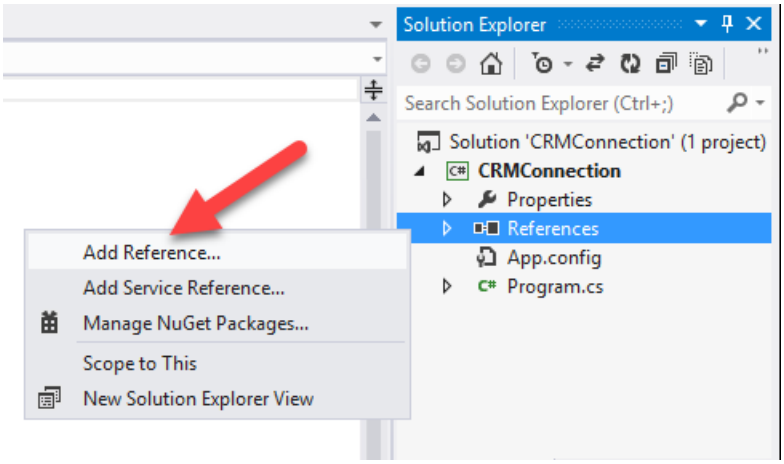
1) Open Visual Studio 2015 and click File | New | Project.



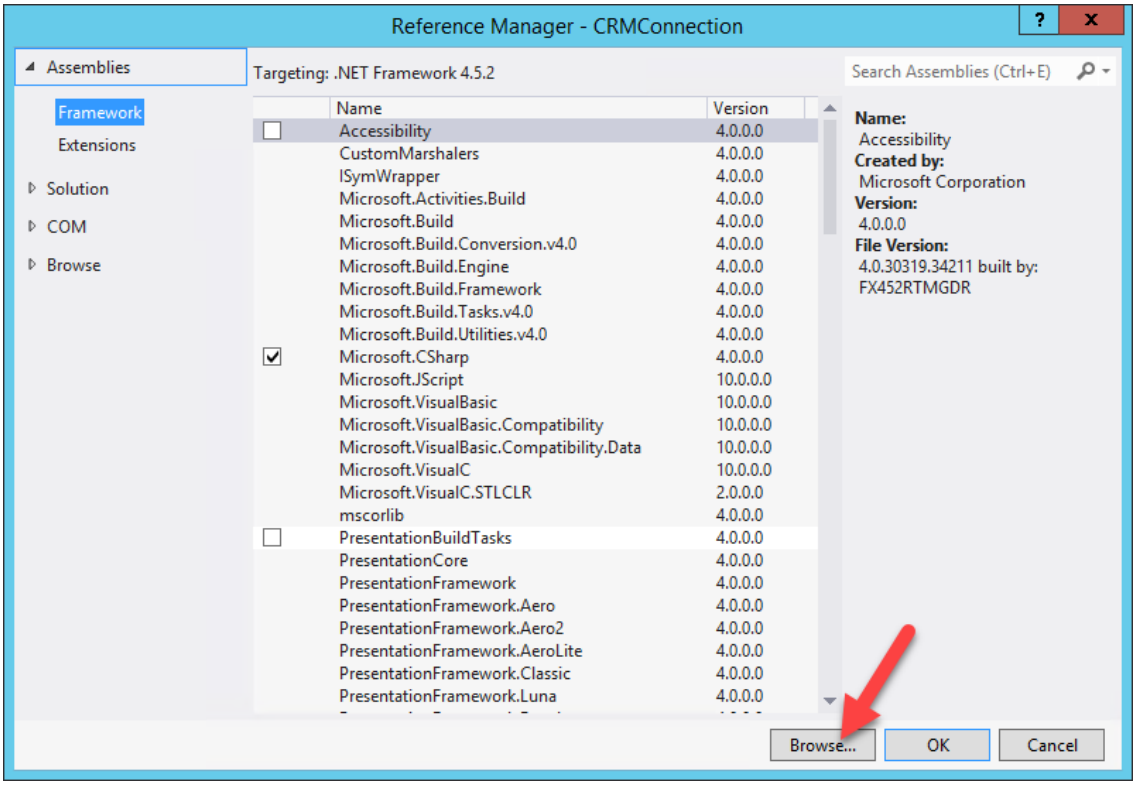
2) Select Console Application, select **.NET Framework 4.5.2**, enter **QueryingCRMData** for **name** and click **OK**.



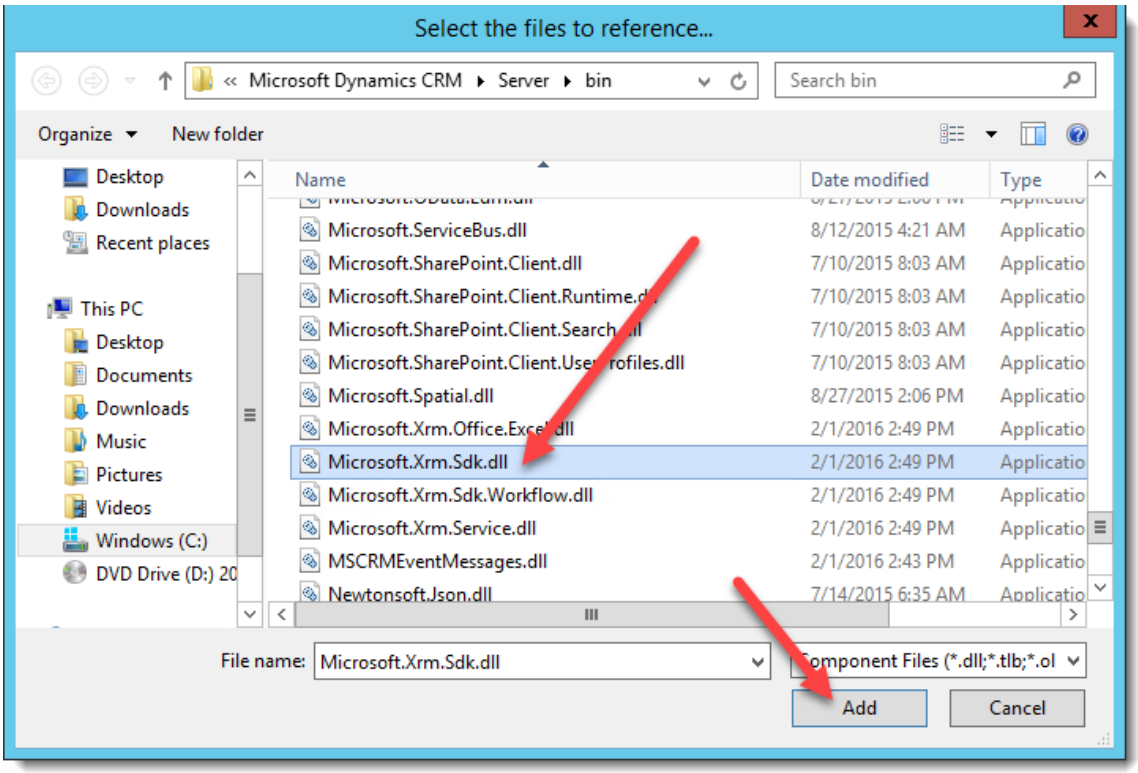
3) Right click the **References** folder and click **Add Reference**.



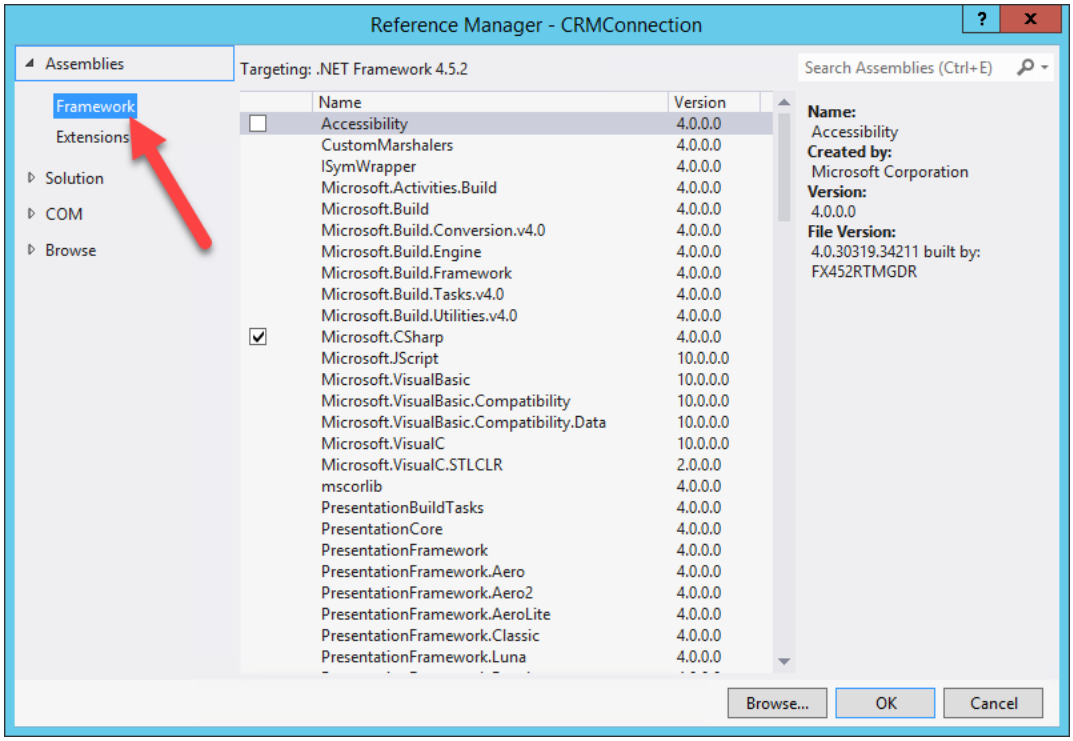
4) In the **Reference Manager** dialog, click **Browse**.



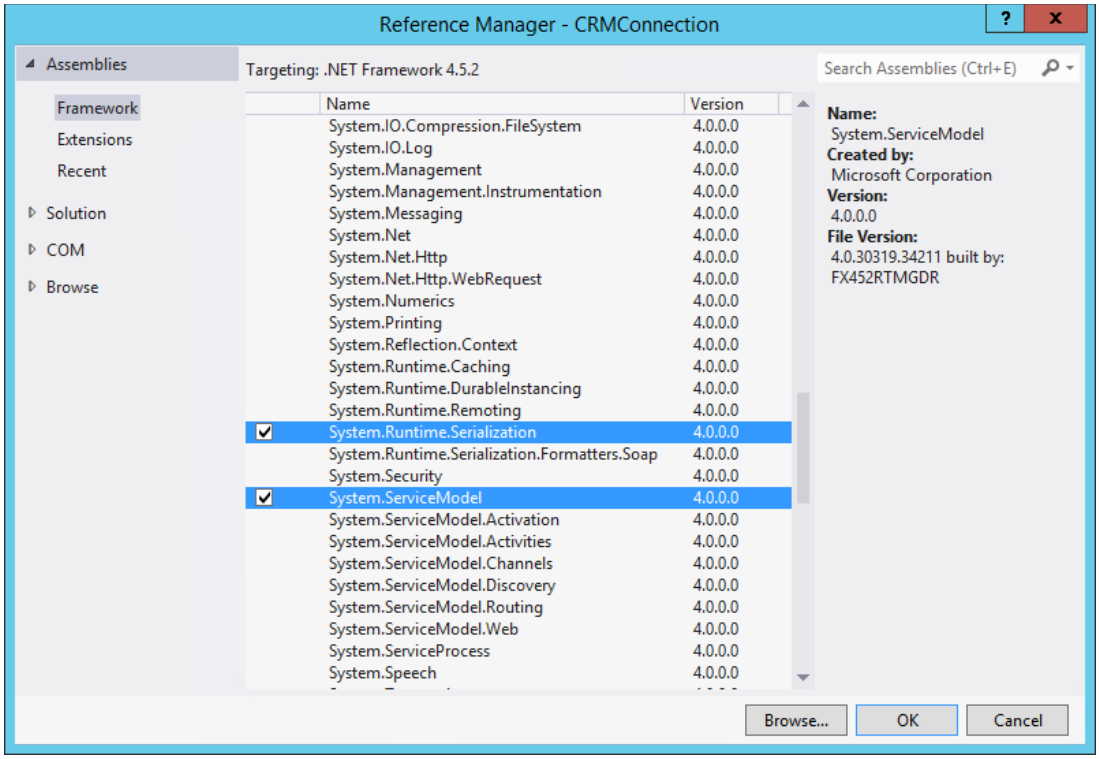
5) Navigate to the **C:\Program Files\Microsoft Dynamics CRM\Server\bin** folder. Select the **Microsoft.Xrm.Sdk.dll** assembly, and then click **Add**.



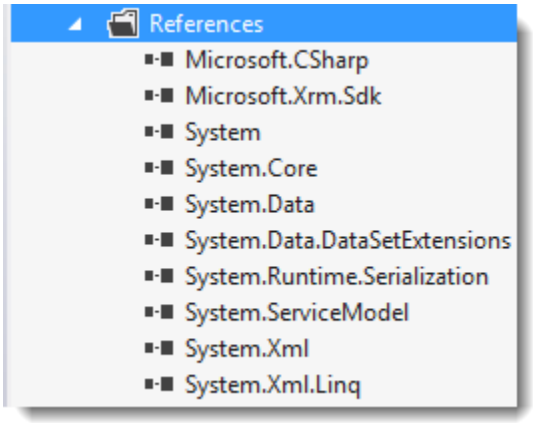
6) In the **Reference Manager** dialog, click **Framework**.



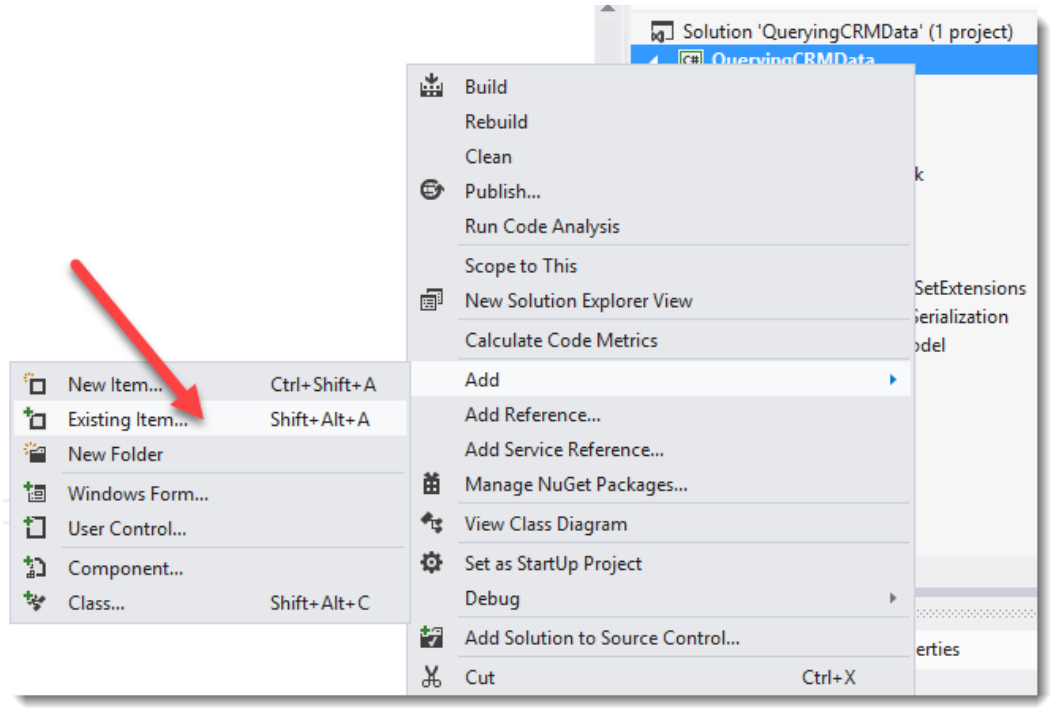
7) Search and select **System.Runtime.Serialization** and **System.ServiceModel** and click **OK**.



8) Make sure that all the required references -shown in the following diagram- were added to a project.



9) Right click on the **Project** and click **Add | Existing Item**.



10) Browse to the **C:\CRM2014** folder, select **XrmData.cs** and click **Add**.
11) In the Solution Explorer, right click the **QueryingCRMData** project, click **Rebuild**, and make sure the project builds successfully.
12) Open **Program.cs** and add the following lines of code.

```
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Client;
using Microsoft.Xrm.Sdk.Query;
using System.ServiceModel.Description;
```

```
Program.cs
QueryingCRMData.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Client;
using Microsoft.Xrm.Sdk.Query;
using System.ServiceModel.Description;

namespace QueryingCRMData
```

13) In the **Main** static method, add the following code to connect to the **Organization Service**.

```
ClientCredentials credentials = new ClientCredentials();
credentials.Windows.ClientCredential =
    System.Net.CredentialCache.DefaultNetworkCredentials;
 OrganizationServiceProxy _serviceProxy =
   new OrganizationServiceProxy(new Uri(
"http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
                null, credentials, null);
```

14) Add the following method after the **Main** method of the **Program** class.

```
static DataCollection<Entity> QueryOpportunities(OrganizationServiceProxy crmSvc)
{
    return null;
}
```

```
Program.cs*
QueryingCRMData | QueryingCRMData.Program | QueryOpportunities(OrganizationServiceProxy
14  0 references
    class Program
15  {
        0 references
16      static void Main(string[] args)
17      {
18          ClientCredentials credentials = new ClientCredentials();
19          credentials.Windows.ClientCredential =
20              System.Net.CredentialCache.DefaultNetworkCredentials;
21          OrganizationServiceProxy _serviceProxy =
22            new OrganizationServiceProxy(new Uri(
23        "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                        null, credentials, null);
25
26      }
27
        0 references
28      static DataCollection<Entity> QueryOpportunities(OrganizationServiceProxy crmSvc)
29      {
30          return null;
31      }
32
33  }
34 }
```

15) Inside the **QueryOpportunities** method, create a new Query Expression that will retrieve name and estimated values for all Opportunities.

```
var queryExpression = new QueryExpression()
{
    EntityName = Opportunity.EntityLogicalName,
    ColumnSet = new ColumnSet("name", "estimatedvalue")
};
```

16) Call the RetrieveMultiple method by passing the Query Expression object.

```
var queryExpressionResult =  crmSvc.RetrieveMultiple(queryExpression);
```

```
26          }
27
            0 references
28          static DataCollection<Entity> QueryOpportunities(OrganizationServiceProxy crmSvc)
29          {
30              var queryExpression = new QueryExpression()
31              {
32                  EntityName = Opportunity.EntityLogicalName,
33                  ColumnSet = new ColumnSet("name", "estimatedvalue")
34              };
35
36              var queryExpressionResult = crmSvc.RetrieveMultiple(queryExpression);
37
38              return null;
39          }
40
```

17) Replace the return null line with the line below.

```
return queryExpressionResult.Entities;
```

```
            0 references
28          static DataCollection<Entity> QueryOpportunities(OrganizationServiceProxy crmSvc)
29          {
30              var queryExpression = new QueryExpression()
31              {
32                  EntityName = Opportunity.EntityLogicalName,
33                  ColumnSet = new ColumnSet("name", "estimatedvalue")
34              };
35
36              var queryExpressionResult = crmSvc.RetrieveMultiple(queryExpression);
37
38              return queryExpressionResult.Entities;
39          }
```

18) In the **Main** method, call the **EnableProxyTypes** method of the **OrganizationServiceProxy** instance to enable support for the early-bound entity types.

```
_serviceProxy.EnableProxyTypes();
```

19) Call the QueryOpportunities method from the main method.

```
var opportunities = QueryOpportunities(_serviceProxy);
```

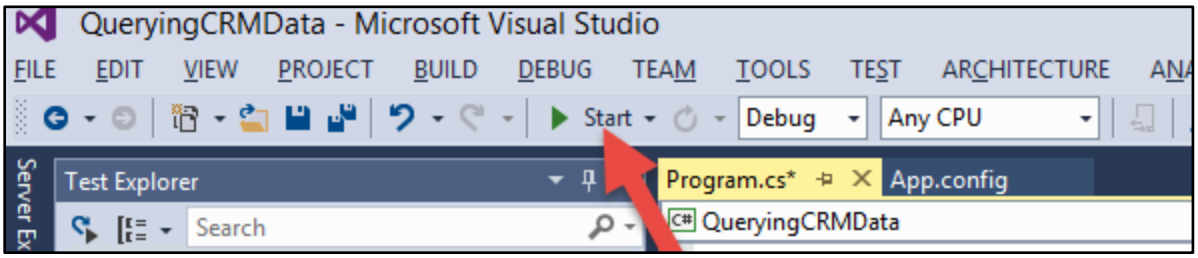
```
14      class Program
15      {
            0 references
16          static void Main(string[] args)
17          {
18              ClientCredentials credentials = new ClientCredentials();
19              credentials.Windows.ClientCredential =
20                  System.Net.CredentialCache.DefaultNetworkCredentials;
21              OrganizationServiceProxy _serviceProxy =
22                new OrganizationServiceProxy(new Uri(
23             "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                            null, credentials, null);
25
26              _serviceProxy.EnableProxyTypes();
27              var opportunities = QueryOpportunities(_serviceProxy);
28
29          }
```

20) Display the name and estimated revenue for each Opportunity on the console.

```
foreach (var record in opportunities)
{
    Opportunity opportunity = record as Opportunity;
    decimal estRevenue = 0;
    if (opportunity.EstimatedValue != null)
        estRevenue = decimal.Floor(opportunity.EstimatedValue.Value);

    Console.WriteLine(String.Format("Topic: {0}, Est. Revenue ${1}\n",
opportunity.Name, estRevenue));
}
Console.ReadLine();
```

```
16          static void Main(string[] args)
17          {
18              ClientCredentials credentials = new ClientCredentials();
19              credentials.Windows.ClientCredential =
20                  System.Net.CredentialCache.DefaultNetworkCredentials;
21              OrganizationServiceProxy _serviceProxy =
22                new OrganizationServiceProxy(new Uri(
23             "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                            null, credentials, null);
25
26              _serviceProxy.EnableProxyTypes();
27              var opportunities = QueryOpportunities(_serviceProxy);
28
29              foreach (var record in opportunities)
30              {
31                  Opportunity opportunity = record as Opportunity;
32                  decimal estRevenue = 0;
33                  if (opportunity.EstimatedValue != null)
34                      estRevenue = decimal.Floor(opportunity.EstimatedValue.Value);
35
36                  Console.WriteLine(String.Format("Topic: {0}, Est. Revenue ${1}\n", opportunity.Name, estRevenue));
37              }
38              Console.ReadLine();
39          }
```

21) Click **Build | Build Solution**.
22) Run the project.



23) The console will display name and estimated revenue for all opportunities.

```
Topic: 10 orders of Product SKU JJ202 (sample), Est. Revenue $22000

Topic: 10 orders or Product SKU AX305 this summer (sample), Est. Revenue $15000

Topic: Some interest in our JJ line of products (sample), Est. Revenue $95000

Topic: 6 orders of Product SKU JJ202 (sample), Est. Revenue $10000

Topic: Needs to restock their supply of Product SKU AX305; will purchase at least 25-50 (sample), Est. Revenue $25000

Topic: Very interested in our products - need to follow up (sample), Est. Revenue $40000

Topic: They sell many of the same items that we do - need to follow up (sample), Est. Revenue $26000

Topic: Very likely will order 18 Product SKU JJ202 this year (sample), Est. Revenue $30000

Topic: Very likely will order 73 Product SKU JJ105 this year (sample), Est. Revenue $16000

Topic: Will be ordering about 110 items of all types (sample), Est. Revenue $25000
```

24) Press enter to close the console application.

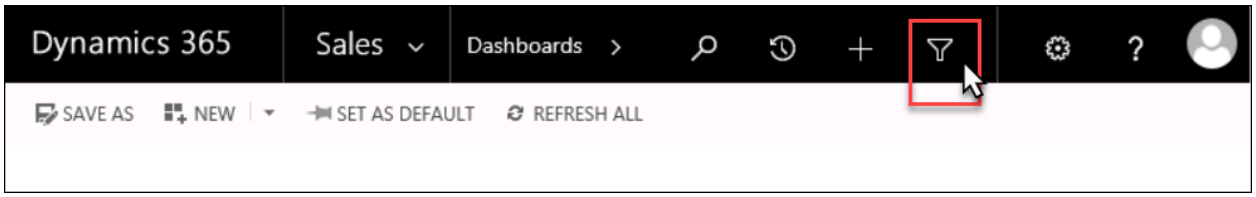
Exercise 2 – Fetch Expression

In this exercise, you will use FetchXML to create a Query that will display estimated revenue for Opportunities. As you did in the previous exercise, you will create a basic Query that will display estimated revenue for Opportunities.

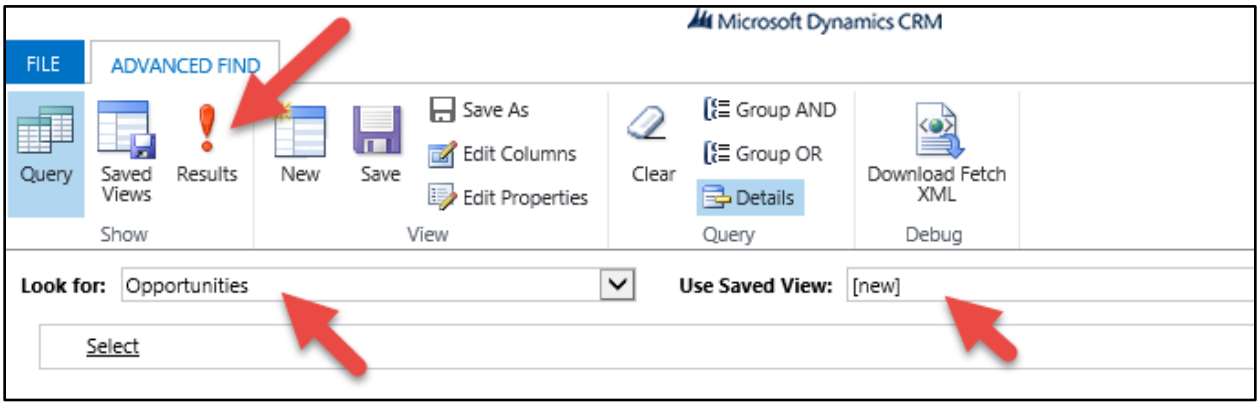
Task 1: FetchXML

In this task you will be creating a basic FetchXML query using the Advanced Find in Dynamics 365.

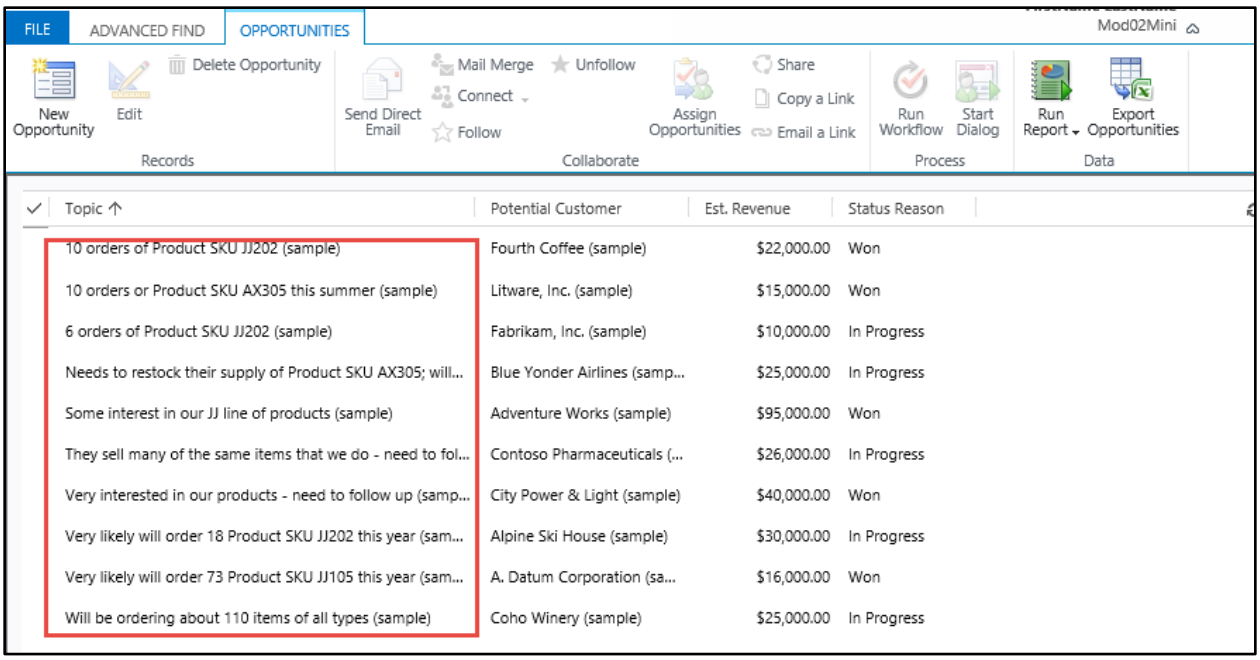
1) In your Dynamics 365, click Advanced Find.



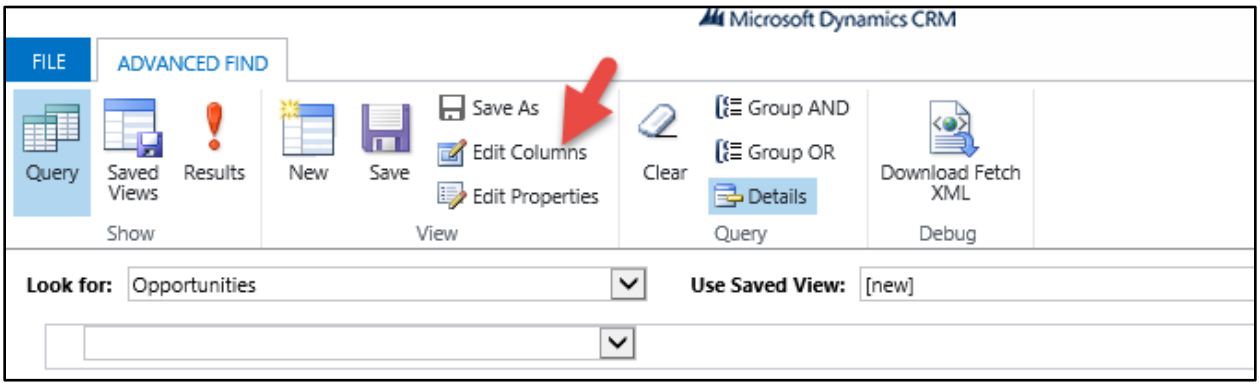
2) Select **Opportunities** from the **Look for** dropdown, select **New** from the **Use Saved View** dropdown and click **Results**.



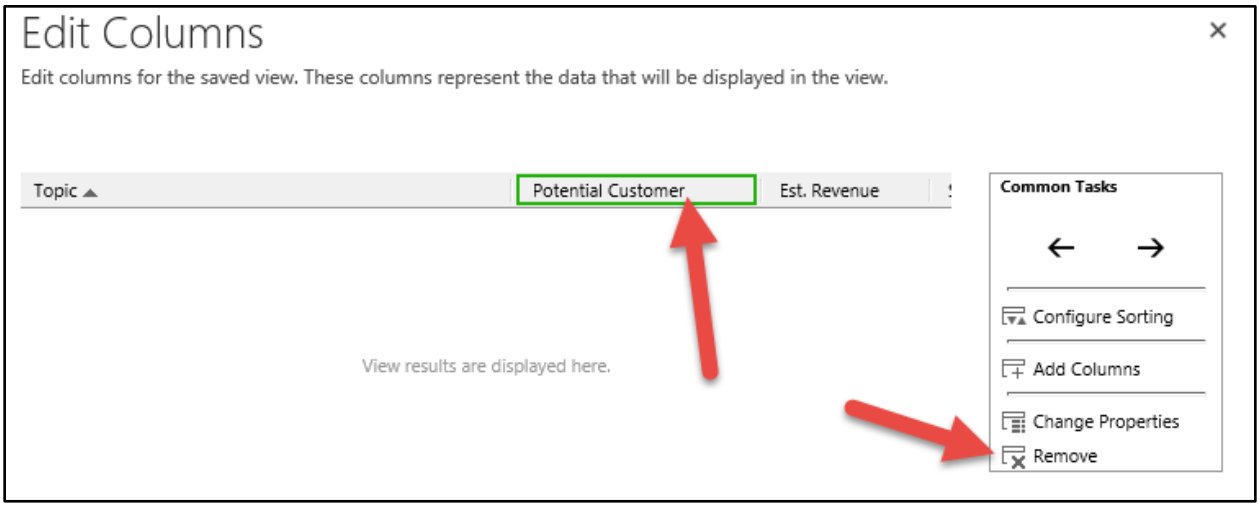
3) Make sure you get some Opportunities for result.



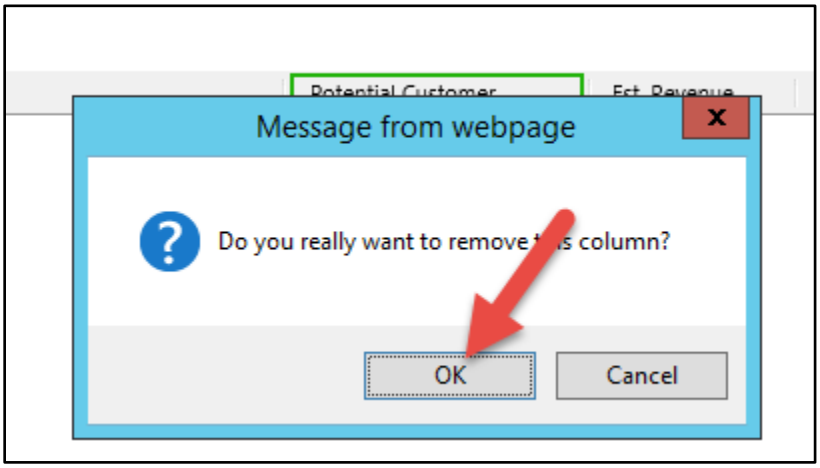
4) Select the Advanced Find tab and click Edit Columns.



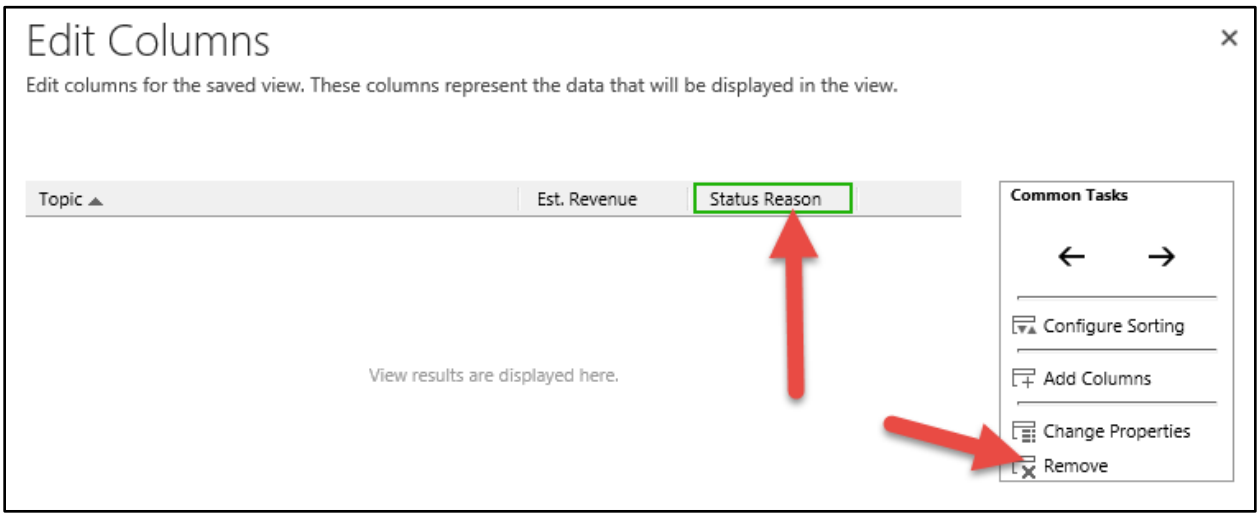
5) Select the **Potential Customer** column and click **Remove**.



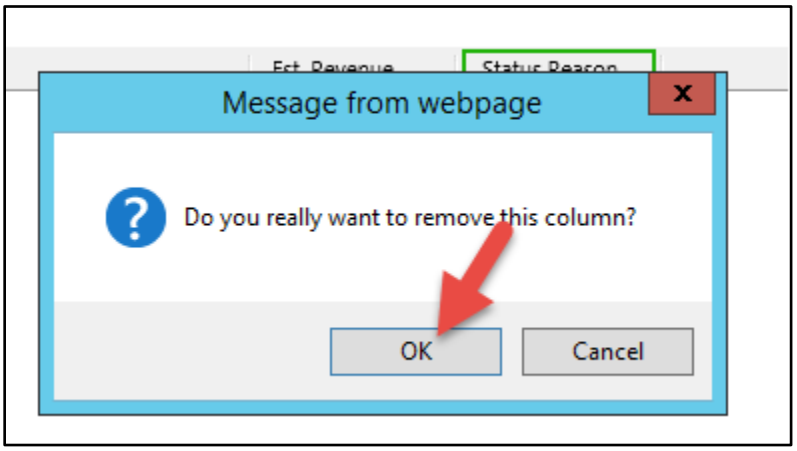
6) Click **OK**.



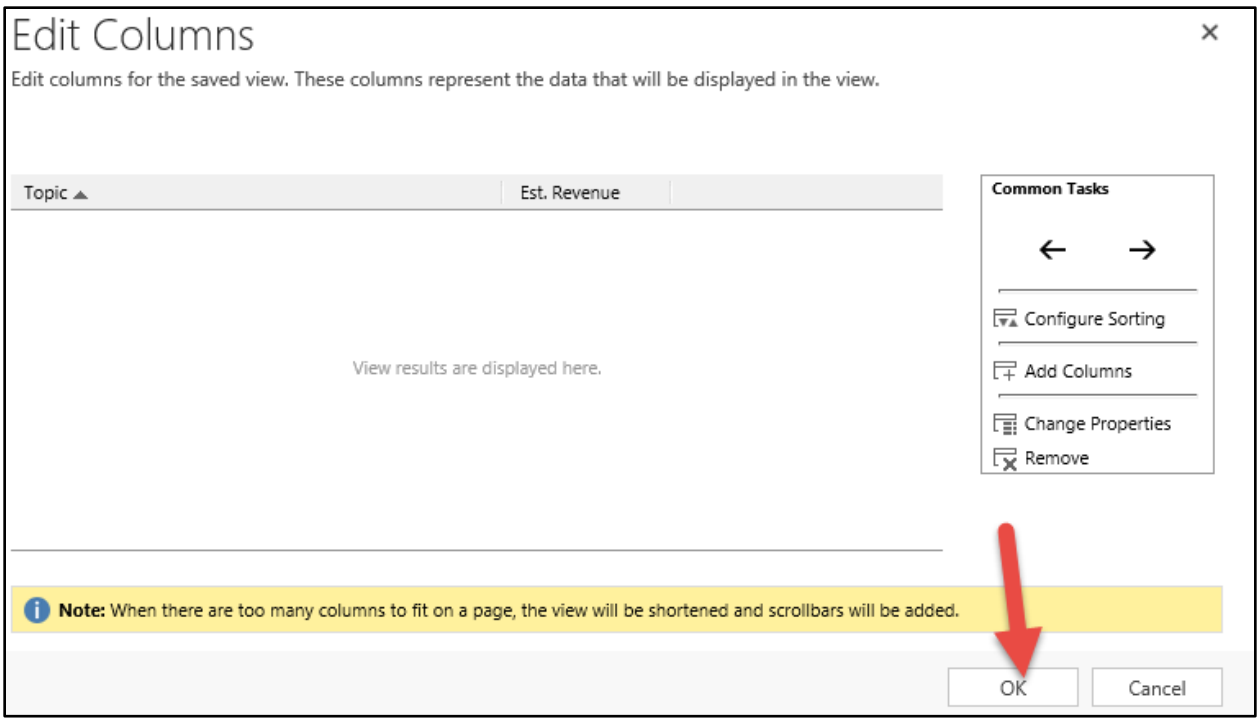
7) Select the **Status Reason** column and click **Remove**.



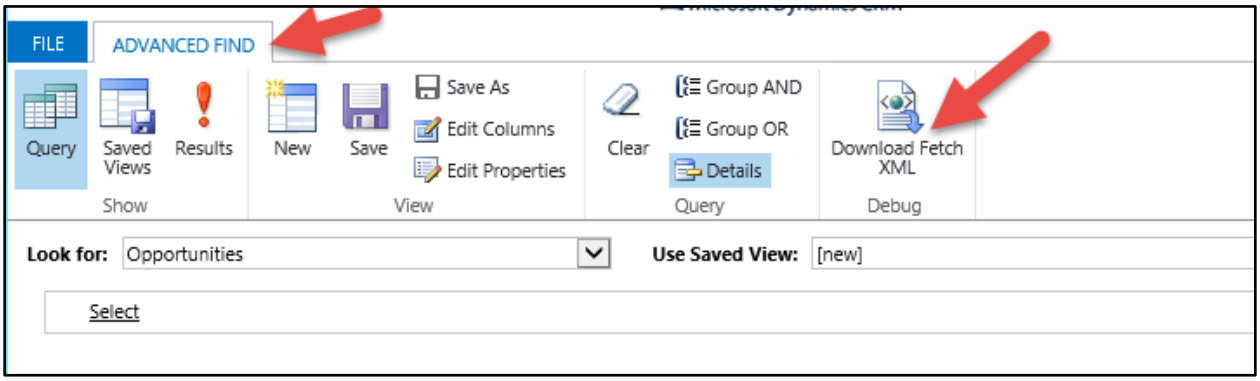
8) Click **OK**.



9) Click **OK**.



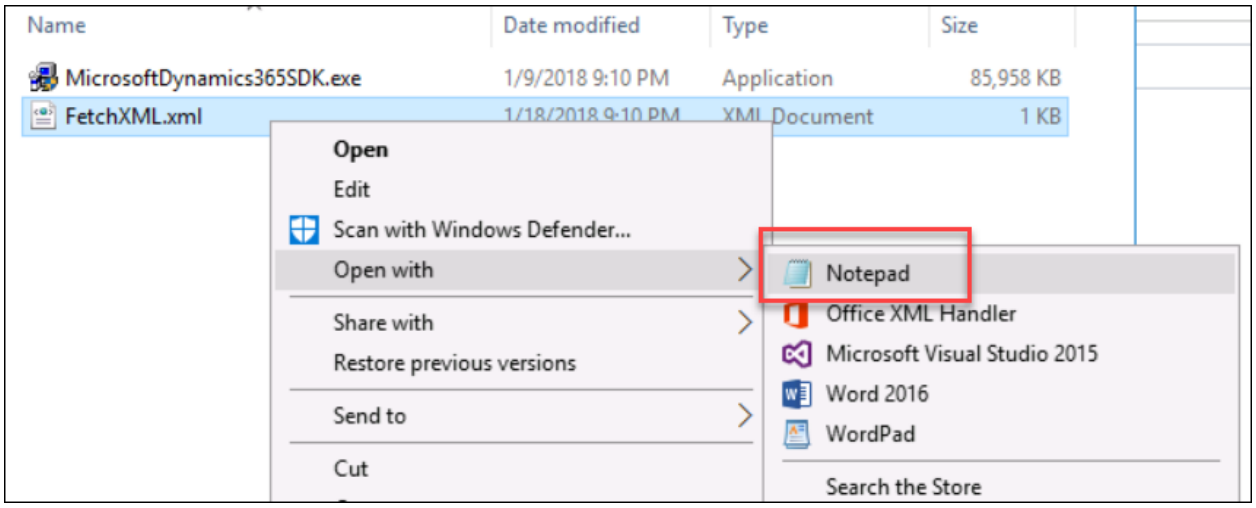
10) Click Download **FetchXML**.



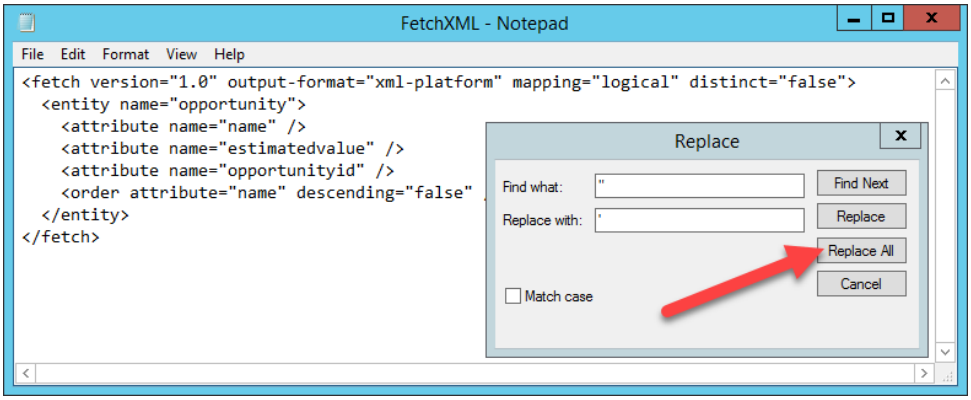
11) Click **Save**.

12) Click **Open folder**.

13) Right-click **Fetch.xml**, click **Open with** and select **Notepad**.



14) Replace all double quotes with single quote.



15) Close the Replace dialog box and keep Notepad open.
16) In Visual Studio, in the **Program.cs** file and add the following method.

```
static List<Entity> FetchXMLOpportunities(IOrganizationService service)
{
    return null;
}
```

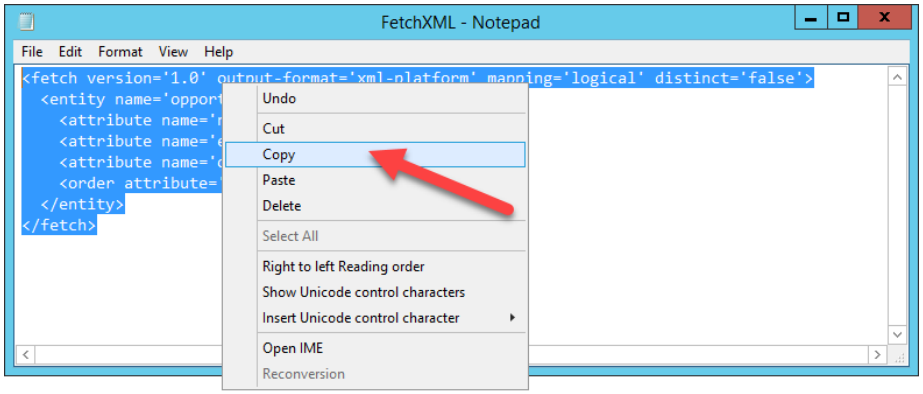
```
40
        1 reference
41      static DataCollection<Entity> QueryOpportunities(OrganizationServiceProxy crmSvc)
42      {
43          var queryExpression = new QueryExpression()
44          {
45              EntityName = Opportunity.EntityLogicalName,
46              ColumnSet = new ColumnSet("name", "estimatedvalue")
47          };
48
49          var queryExpressionResult = crmSvc.RetrieveMultiple(queryExpression);
50
51          return queryExpressionResult.Entities;
52      }
53
        0 references
54      static List<Entity> FetchXMLOpportunities(IOrganizationService service)
55      {
56          return null;
57      }
58
59
60  }
61 }
```

17) Inside the new method, add new variable that will hold the FetchXML.

```
var fetchXml = @"";
```

```
        0 references
54      static List<Entity> FetchXMLOpportunities(IOrganizationService service)
55      {
56          var fetchXml = @"";
57
58          return null;
59      }
```

18) Copy the entire XML fragment from Notepad.



19) Paste the XML you copied inside of the quotation marks.

```
54          static List<Entity> FetchXMLOpportunities(IOrganizationService service)
55          {
56              var fetchXml = @"<fetch version='1.0' output-format='xml-platform' mapping='logical' distinct='false'>
57        <entity name='opportunity'>
58          <attribute name='name' />
59          <attribute name='estimatedvalue' />
60          <attribute name='opportunityid' />
61          <order attribute='name' descending='false' />
62        </entity>
63      </fetch>";
64
65              return null;
66          }
```

20) Create new FetchExpression passing the FetchXML string.

```
var fetchExpression = new FetchExpression(fetchXml);
```

```
        static List<Entity> FetchXMLOpportunities(IOrganizationService service)
        {
            var fetchXml = @"<fetch version='1.0' output-format='xml-platform' mapping='logical' distinct='false'>
  <entity name='opportunity'>
    <attribute name='name' />
    <attribute name='estimatedvalue' />
    <attribute name='opportunityid' />
    <order attribute='name' descending='false' />
  </entity>
</fetch>";

            var fetchExpression = new FetchExpression(fetchXml);

            return null;
        }
```

21) Call Retrieve Multiple.

```
var fetchResult  = service.RetrieveMultiple(fetchExpression);
```

```
        static List<Entity> FetchXMLOpportunities(IOrganizationService service)
        {
            var fetchXml = @"<fetch version='1.0' output-format='xml-platform' mapping='logical' distinct='false'>
  <entity name='opportunity'>
    <attribute name='name' />
    <attribute name='estimatedvalue' />
    <attribute name='opportunityid' />
    <order attribute='name' descending='false' />
  </entity>
</fetch>";

            var fetchExpression = new FetchExpression(fetchXml);
            var fetchResult  = service.RetrieveMultiple(fetchExpression);

            return null;
        }
```

22) Return the Fetch Result Entities, instead of null.

```
return fetchResult.Entities.ToList();
```

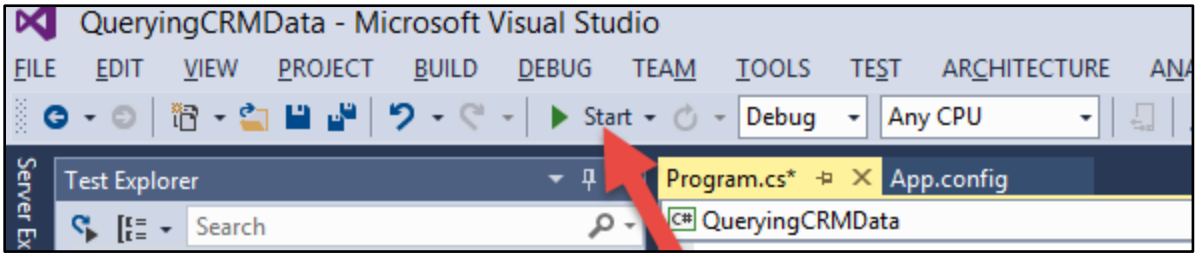
```
        static List<Entity> FetchXMLOpportunities(IOrganizationService service)
        {
            var fetchXml = @"<fetch version='1.0' output-format='xml-platform' mapping='logical' distinct='false'>
  <entity name='opportunity'>
    <attribute name='name' />
    <attribute name='estimatedvalue' />
    <attribute name='opportunityid' />
    <order attribute='name' descending='false' />
  </entity>
</fetch>";

            var fetchExpression = new FetchExpression(fetchXml);
            var fetchResult  = service.RetrieveMultiple(fetchExpression);

            return fetchResult.Entities.ToList();
        }
```

23) Go to the **Main** method and call **FetchXMLOpportunities** instead of **QueryOpportunities**.

```
16          static void Main(string[] args)
17          {
18              ClientCredentials credentials = new ClientCredentials();
19              credentials.Windows.ClientCredential =
20                  System.Net.CredentialCache.DefaultNetworkCredentials;
21              OrganizationServiceProxy _serviceProxy =
22                new OrganizationServiceProxy(new Uri(
23             "http://minint-f36s5eh:5555/AdventureWorksCycles/XRMServices/2011/Organization.svc"),
24                            null, credentials, null);
25
26              _serviceProxy.EnableProxyTypes();
27              var opportunities = FetchXMLOpportunities(_serviceProxy);
28
29              foreach (var record in opportunities)
30              {
31                  Opportunity opportunity = record as Opportunity;
32                  decimal estRevenue = 0;
33                  if (opportunity.EstimatedValue != null)
34                      estRevenue = decimal.Floor(opportunity.EstimatedValue.Value);
35
36                  Console.WriteLine(String.Format("Topic: {0}, Est. Revenue ${1}\n", opportunity.Name, estRevenue));
37              }
38              Console.ReadLine();
39          }
```

24) Click **Build | Build Solution**.

25) Run the project.

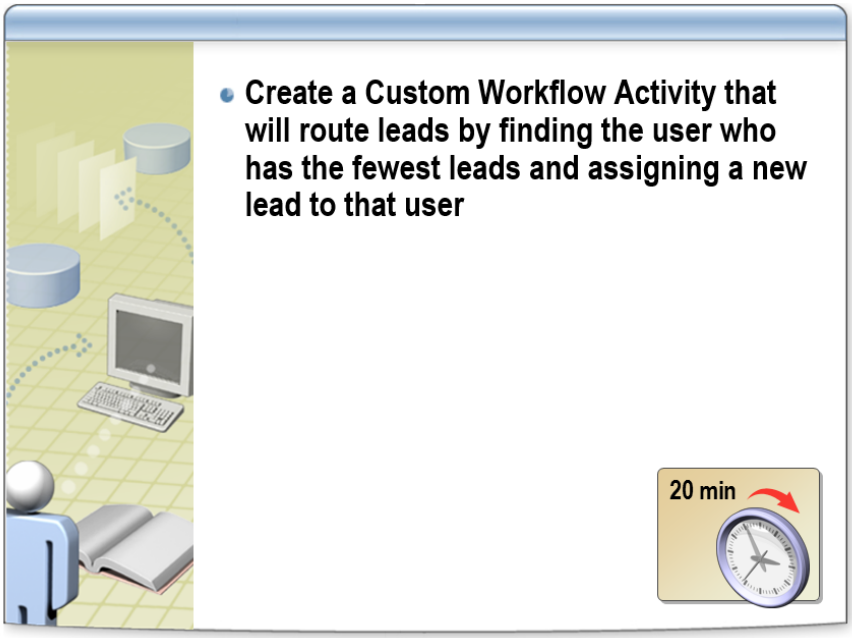


26) The console will display all Opportunities.

```
Topic: 10 orders of Product SKU JJ202 (sample), Est. Revenue $22000

Topic: 10 orders or Product SKU AX305 this summer (sample), Est. Revenue $15000

Topic: 6 orders of Product SKU JJ202 (sample), Est. Revenue $10000

Topic: Needs to restock their supply of Product SKU AX305; will purchase at least 25-50 (sample), Est. Revenue $25000

Topic: Some interest in our JJ line of products (sample), Est. Revenue $95000

Topic: They sell many of the same items that we do - need to follow up (sample), Est. Revenue $26000

Topic: Very interested in our products - need to follow up (sample), Est. Revenue $40000

Topic: Very likely will order 18 Product SKU JJ202 this year (sample), Est. Revenue $30000

Topic: Very likely will order 73 Product SKU JJ105 this year (sample), Est. Revenue $16000

Topic: Will be ordering about 110 items of all types (sample), Est. Revenue $25000
```

25) Press enter to close the console application.

Lab 06: Creating a Custom Workflow Activity



Scenario

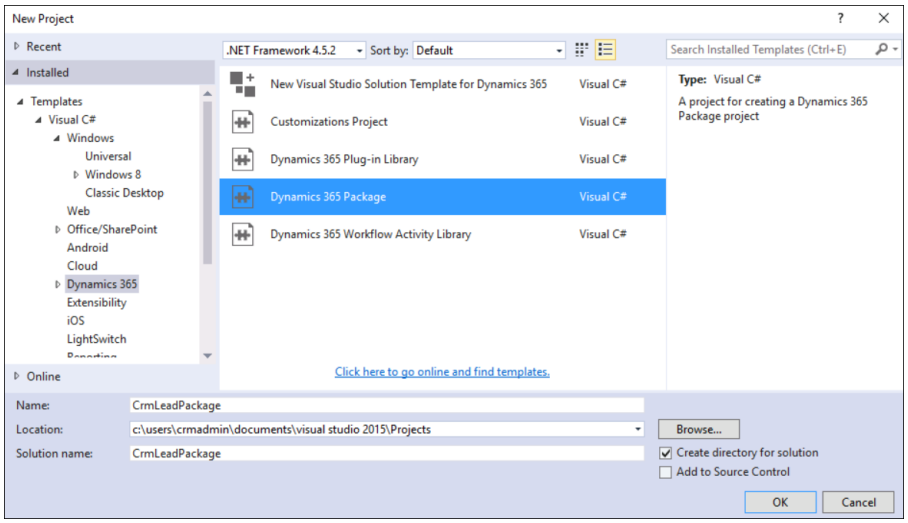
A company is using a telemarketing firm to generate leads from a proprietary call-down list. If people express an interest in the company's product they are sent to the company, where they are entered into Microsoft Dynamics 365 as leads. The company wants the new leads to be assigned to the salesperson with the fewest number of leads, to make sure each salesperson always has a pipeline of new leads to work. This method also rewards salespeople that quickly qualify or disqualify the leads they are given.

In this lab you will create a Custom Workflow Activity that will route leads by finding the user who has the fewest leads and assigning a new lead to that user.

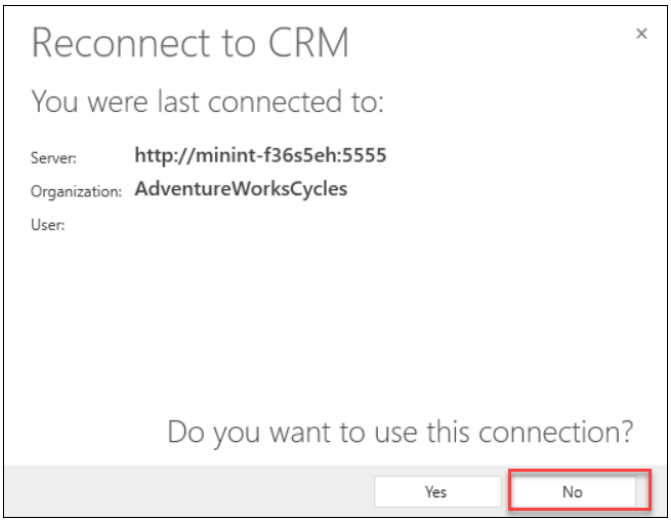
Create a new Dynamics 365 Developer Toolkit project

1. Open **Visual Studio 2015**.
2. Click the **File** menu.
3. Click **New** and then click **Project**.
4. In the **New Project** dialog box under **Templates** > **Visual C#**, select the **Dynamics 365** group to view the available project templates.
5. Select **Dynamics 365 Package**.
6. Change the **Name** text box to **CrmLeadPackage**.

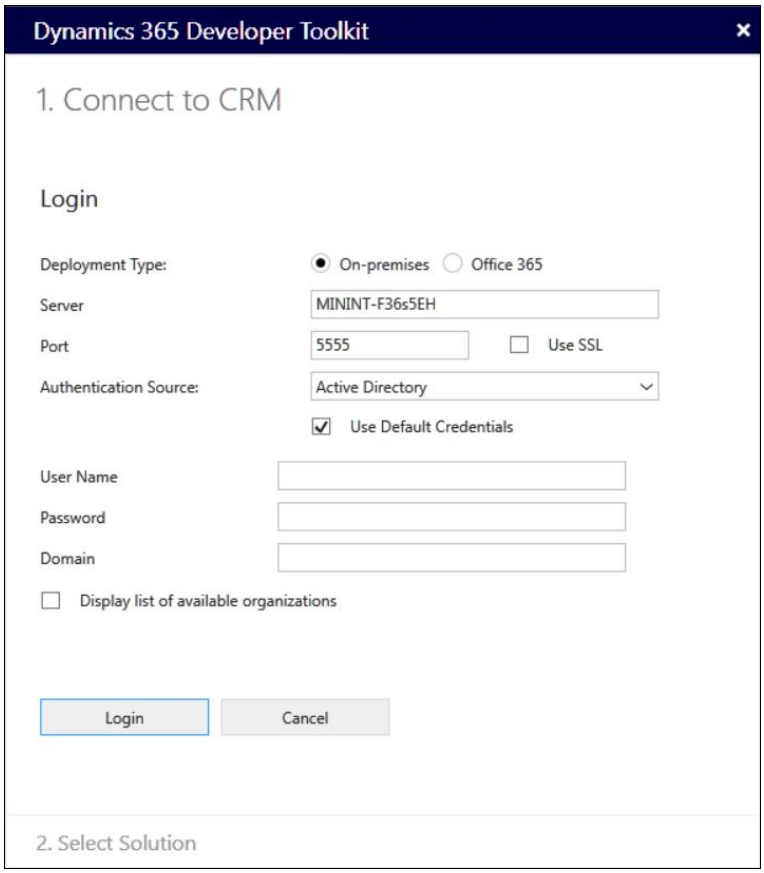
In your new solution, the CrmPackage project is a manifest project that contains all assets to be deployed to Microsoft Dynamics 365 combined with their deployment settings. By default, the outputs from each other project are added as references to this project in so that they can be deployed to the server.



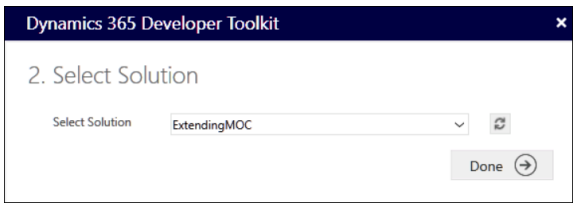
7. Click **OK**.
8. If the **Reconnect to CRM** dialog box is displayed, click **No**.



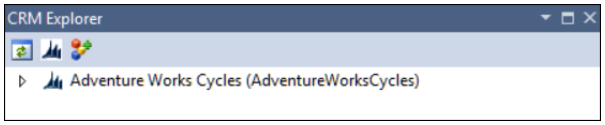
9. In the **Connect to CRM** dialog, in the **Server** textbox type **MININT-F36s5EH**.
10. In the **Port Number** textbox type **5555**.
11. In the **Authentication Source** list, make sure **Active Directory** is selected.
12. Select **Use Default Credentials**.



1) Click **Login**.
2) In the **Select Solution** dialog, in the **Select Solution** drop-down list select **ExtendingMOC**.



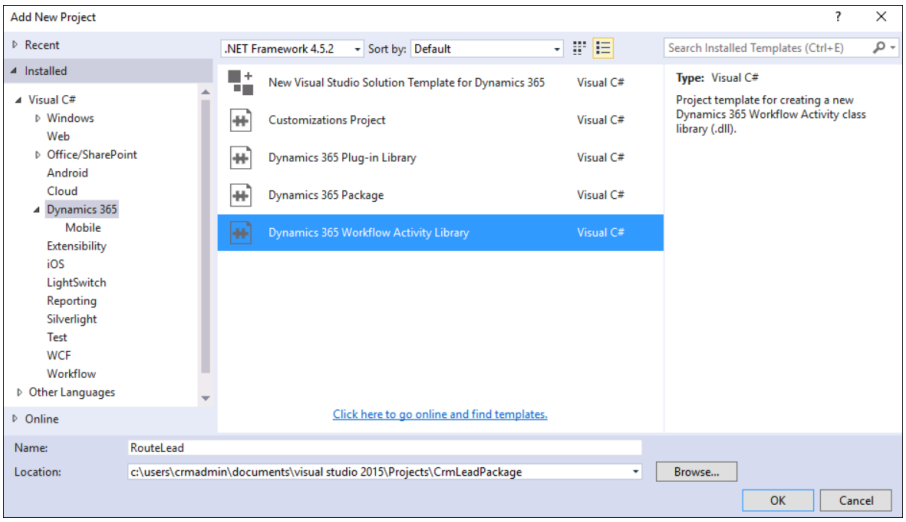
3) Click **Done**.
4) If the **CRM Explorer** window is open, close it.



Add a new Dynamics 365 Workflow Library project

1. In **Solution Explorer**, right-click the solution node, select **Add** and then click **New Project**.
2. In the **Add New Project** dialog box under **Visual C#**, select the **Dynamics 365** group to view the available project templates.

3. Select **Dynamics 365 Workflow Activity Library**.
4. Change the **Name** text box to **RouteLead**.



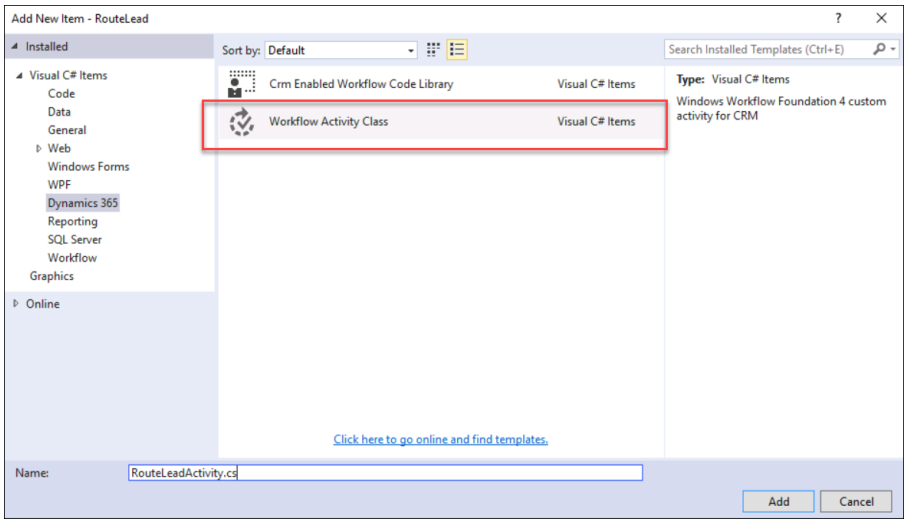
5. Click **OK**.

The workflow library project provides support for implementing custom workflow activities. Workflow activities that are added to the project that are based on the CRM workflow activity template accelerate the starting point for development. The class generated from the template simplifies some of the common code that is required by many workflow activities including:

- Obtaining execution context.
- Obtaining the tracing service.
- Instantiating the Organization Service proxy.

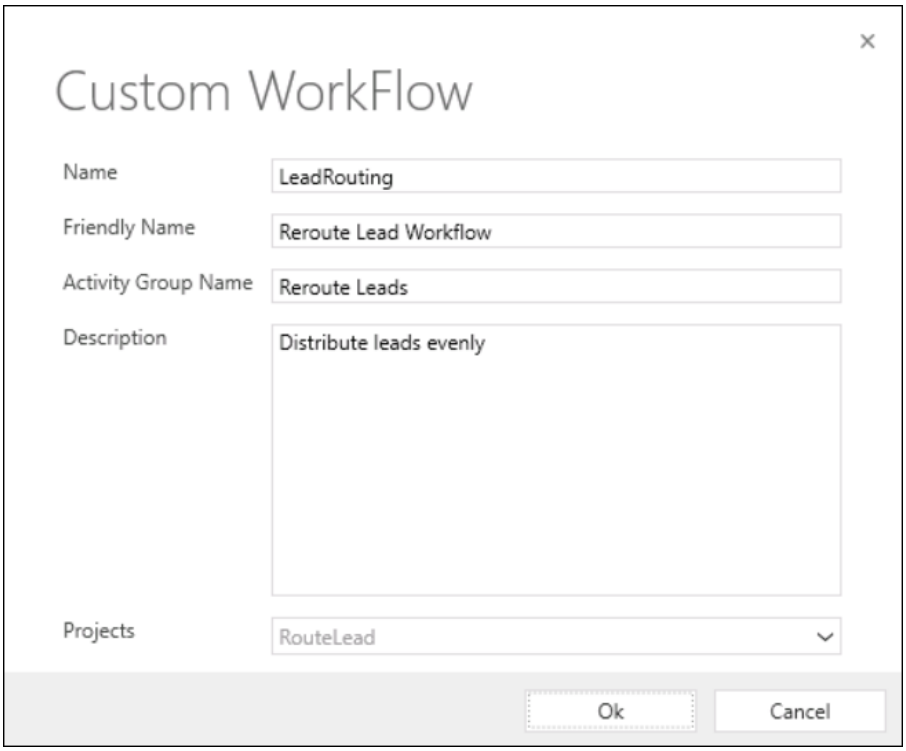
Create a Workflow Activity Class

1. In **Solution Explorer**, right-click the **RouteLead** workflow project node, select **Add** and then click **New Item**.
2. In **Add New Item – RouteLead** dialog box, in the **Dynamics 365** group of installed templates select the **Workflow Activity Class** template.
3. Change the **Name** textbox to **RouteLeadActivity.cs**.



4. Click **Add**.
5. In the **Custom Workflow** dialog box specify field values for your workflow activity according to the descriptions in the following table:

Field	Description
Name	**LeadRouting**
Friendly Name	**Reroute Lead Workflow**
Activity Group Name	**Reroute Leads**
Description	**Distribute leads evenly**

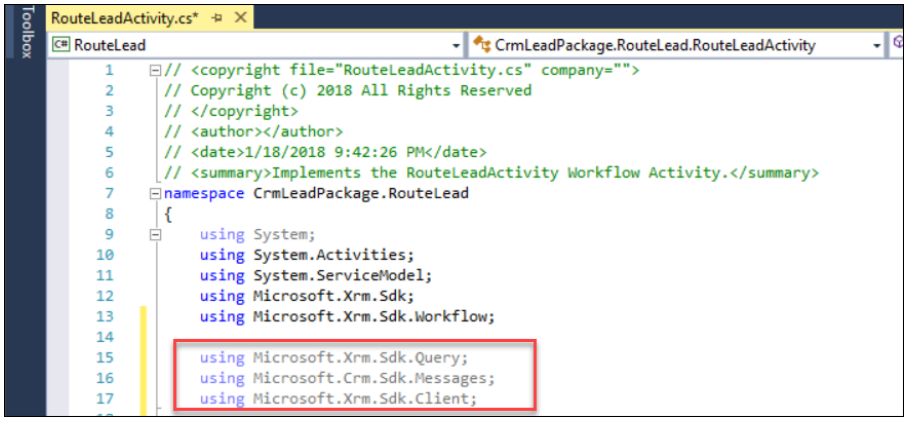


6. Click **OK**.

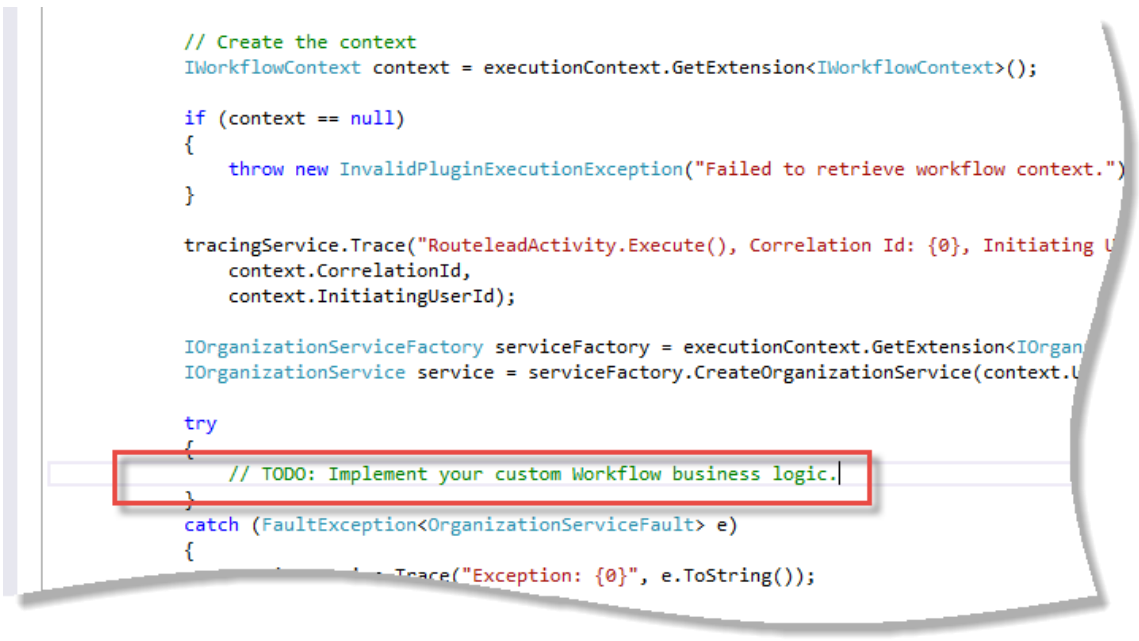
Add code components

1. In the **RouteLeadActivity.cs** file, add the following **using** statements:

```
using Microsoft.Xrm.Sdk.Query;
using Microsoft.Crm.Sdk.Messages;
using Microsoft.Xrm.Sdk.Client;
```



2. Locate the // **TODO: Implement your custom Workflow business logic**. comment.



3. Add the following code below the comment:

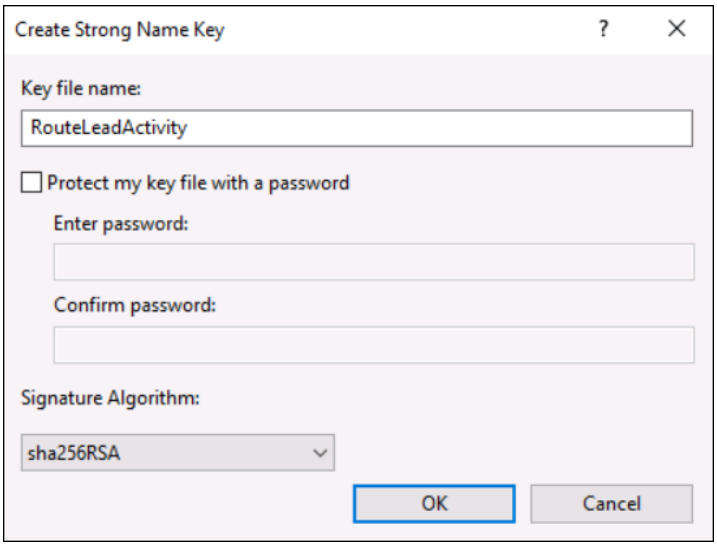
```
// Set the Counters
int currentLeadCount = 0;
int lowLeadCount=-1;
Guid lowUserId = Guid.Empty;
```

```
Guid leadId = context.PrimaryEntityId;

// Create the Query
QueryExpression userQuery = new QueryExpression( ) ;
userQuery.EntityName ="systemuser";
userQuery.ColumnSet = new ColumnSet( );
userQuery.ColumnSet.AllColumns = true;

// Gets the list of users
EntityCollection bec = service.RetrieveMultiple(userQuery);
foreach(Entity e in bec.Entities)
{
    QueryExpression qe = new QueryExpression();
    qe.EntityName="lead";
    qe.ColumnSet = new ColumnSet();
    qe.ColumnSet.AllColumns= true;
    qe.Criteria = new FilterExpression();
    qe.Criteria.FilterOperator = LogicalOperator.And;
    ConditionExpression ce = new ConditionExpression("ownerid",
          ConditionOperator.Equal, e.Attributes["systemuserid"]);
    qe.Criteria.Conditions.Add(ce);
    EntityCollection ecLead = service.RetrieveMultiple(qe);
    currentLeadCount = ecLead.Entities.Count;

    //if the first user, the user is marked the lowest
    if (lowLeadCount == -1)
    {
        lowLeadCount = currentLeadCount;
        lowUserId = new Guid(e.Attributes["systemuserid"].ToString( ));
    }

    // if the number of leads is lowest, the current user is marked lowest
    if (currentLeadCount < lowLeadCount)
    {
      lowLeadCount = currentLeadCount;
      lowUserId = new Guid(e.Attributes["systemuserid"].ToString( ));
    }
}
//Route the lead
if (lowUserId != Guid.Empty)
{
    AssignRequest assignRequest = new AssignRequest()
    {
        Assignee = new EntityReference("systemuser", lowUserId),
        Target = new EntityReference("lead",leadId)
    };

    AssignResponse assignResponse = (AssignResponse) service.Execute(assignRequest);
}
```

```
tracingService.Trace("RouteleadActivity.Execute(), Correlation Id: {0}, Init
    context.CorrelationId,
    context.InitiatingUserId);

IOrganizationServiceFactory serviceFactory = executionContext.GetExtension<IOr
IOrganizationService service = serviceFactory.CreateOrganizationService(contex

try
{
    // TODO: Implement your custom Workflow business logic.
    //Set the counters
    int currentLeadCount = 0;
    int lowLeadCount = -1;

    Guid lowUserId = Guid.Empty;
    Guid leadId = context.PrimaryEntityId;

    //Create the Query
    QueryExpression userQuery = new QueryExpression();
    userQuery.EntityName = "systemuser";
    userQuery.ColumnSet = new ColumnSet();
    userQuery.ColumnSet.AllColumns = true;

    //Gets the list of users
    EntityCollection bec = service.RetrieveMultiple(userQuery);
    foreach (Entity e in bec.Entities)
    {
                                new QueryExpression();
```

Strongly Sign the Assembly

1. Right-click the **RouteLead** project, and then click **Properties**.
2. Click the **Signing** tab.
3. Check the **Sign the assembly** check box.
4. Select **<New...>** in the **Choose a strong name key file** drop-down list.
5. In the **Key File Name** textbox type **RouteLeadActivity**.
6. Clear the **Protect my** key file with a password checkbox.



7. Click **OK**.
8. In **Visual Studio 2015**, click the **Build** menu and then click **Build Solution**.

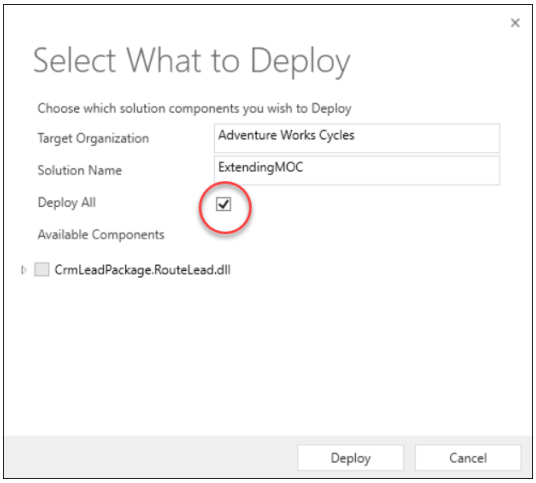
Deploy the solution to the server

The CrmPackage project is a manifest project that contains all assets to be deployed to Microsoft Dynamics 365 combined with their deployment settings. By default, the outputs from each other project are added as references to this project in so that they can be deployed to the server.

1. Right-click the **CrmLeadPackage** project and select **Deploy**.

This builds the dependent projects and deploys the workflow activity to the Microsoft Dynamics 365 server. Any subsequent changes to the workflow activity are pushed to the server during the build and deploy process.

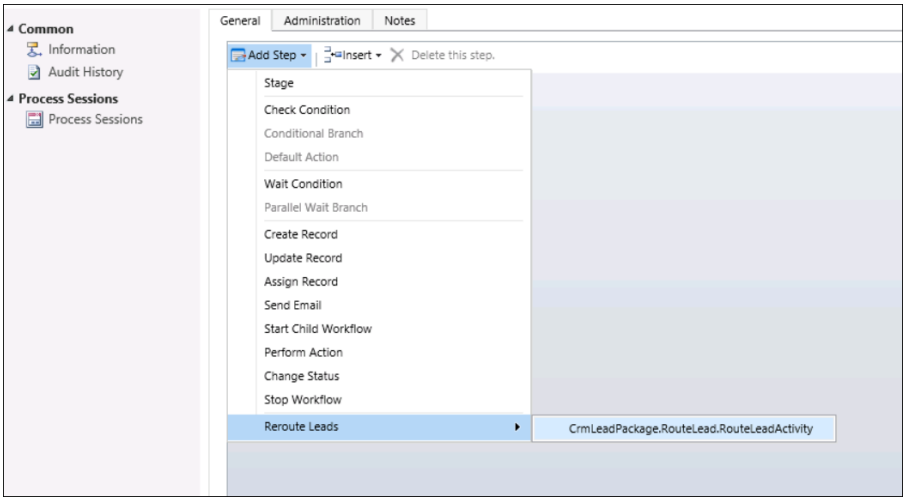
2. In the **Select What to Deploy** dialog box, select **Deploy All**.



3. Click **Deploy**.
4. Verify that the deployment was successful.

Create a new workflow process

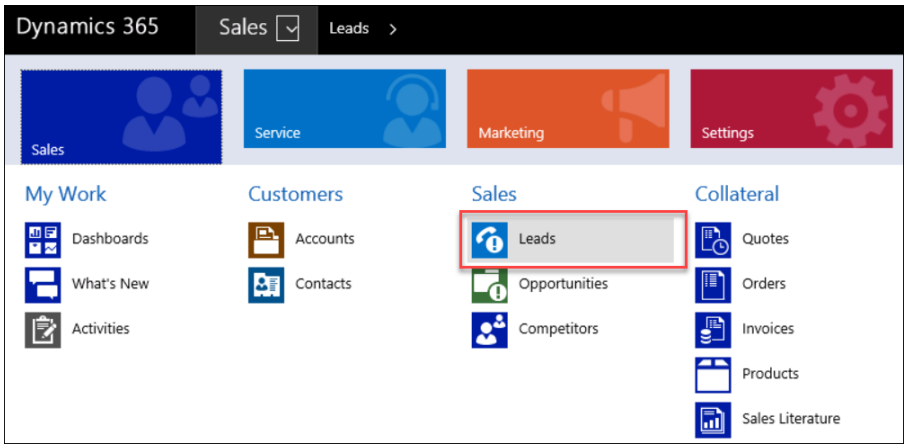
1. Open **Microsoft Dynamics 365** by using Internet Explorer.
2. Go to **Settings** > **Processes**.
3. On the Actions toolbar, click **New**.
4. In the **Process Name** textbox, type **Route Leads**.
5. Select **Lead** in the Entity drop-down list.
6. Select **Workflow** in the category drop-down list and then click **OK**.
7. In the **Process:Route Leads** page, verify that the **Start when Record is Created** check box is selected.
8. Click the **Add Step** button.
9. Click **Reroute Leads** and then click **CrmLeadPackage.RouteLead. RouteLeadActivity**.



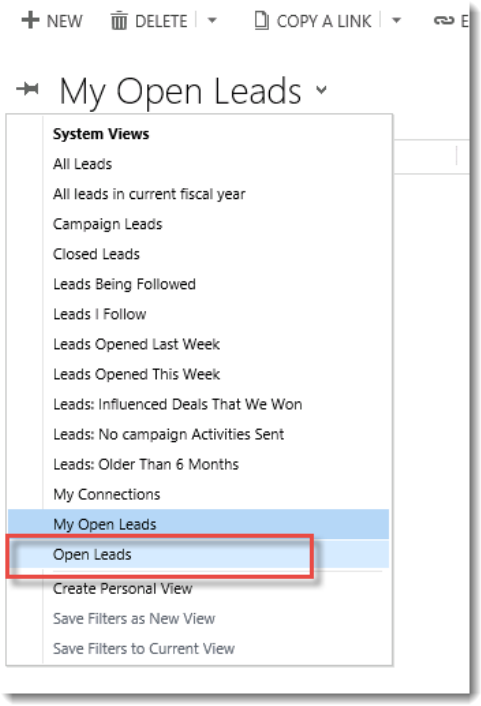
10. On the Actions toolbar, click **Activate**.
11. In the **Process Activate Confirmation** dialog box, click **Activate**.
12. On the Actions toolbar, click **Close**.

Test the Workflow process

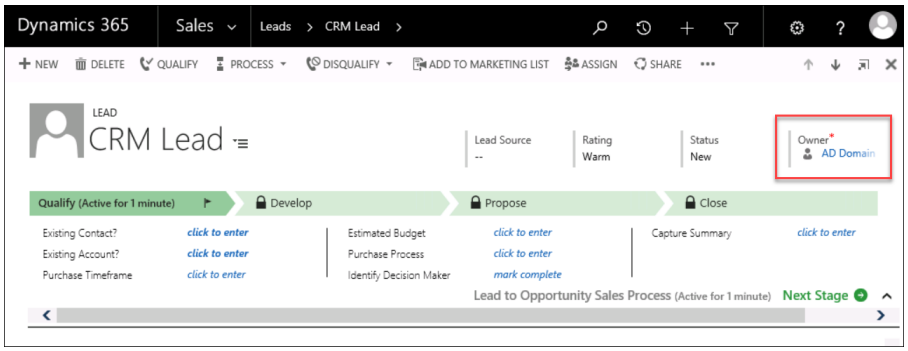
1. In **Microsoft Dynamics 365**, go to **Sales** > **Leads**.



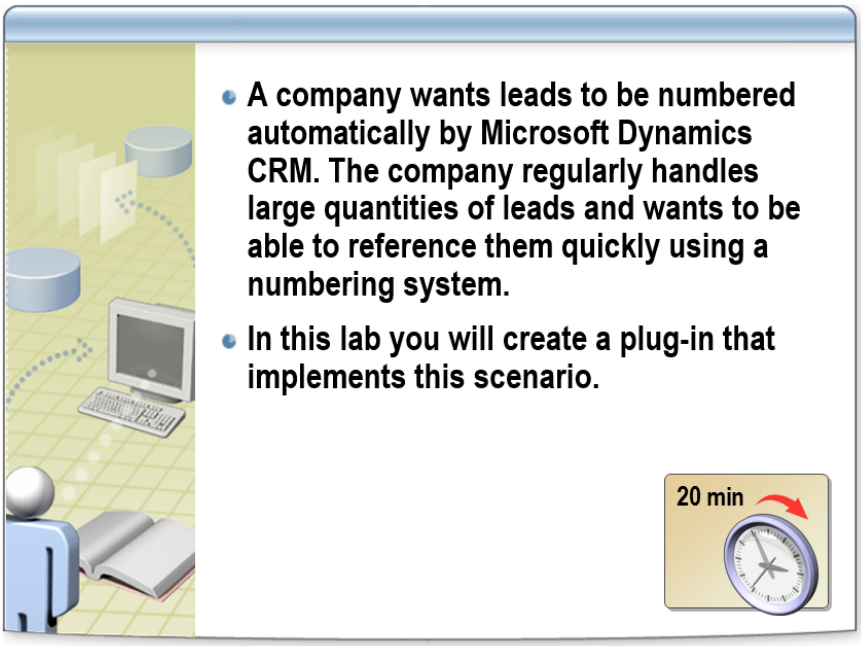
2. Click **New**.
3. In the **Summary** area of the **Lead** screen, enter the required fields under the **Contact** section.
4. Click the **Save and Close** button on the bottom right of the screen.
5. After saving and closing the new lead, select **Open Leads** in the View drop-down list.



6. Open the new lead, and verify that the owner *is not* the CRM Administrator user.


Dynamics 365
Sales
Leads
CRM Lead
NEW
DELETE
QUALIFY
PROCESS
DISQUALIFY
ADD TO MARKETING LIST
ASSIGN
SHARE
LEAD
CRM Lead
Lead Source
--
Rating
Warm
Status
New
Owner*
AD Domain
Qualify (Active for 1 minute)
Develop
Propose
Close
Existing Contact?
click to enter
Existing Account?
click to enter
Purchase Timeframe
click to enter
Estimated Budget
click to enter
Purchase Process
click to enter
Identify Decision Maker
mark complete
Capture Summary
click to enter
Lead to Opportunity Sales Process (Active for 1 minute)
Next Stage

Lab 07: Creating a Plug-in



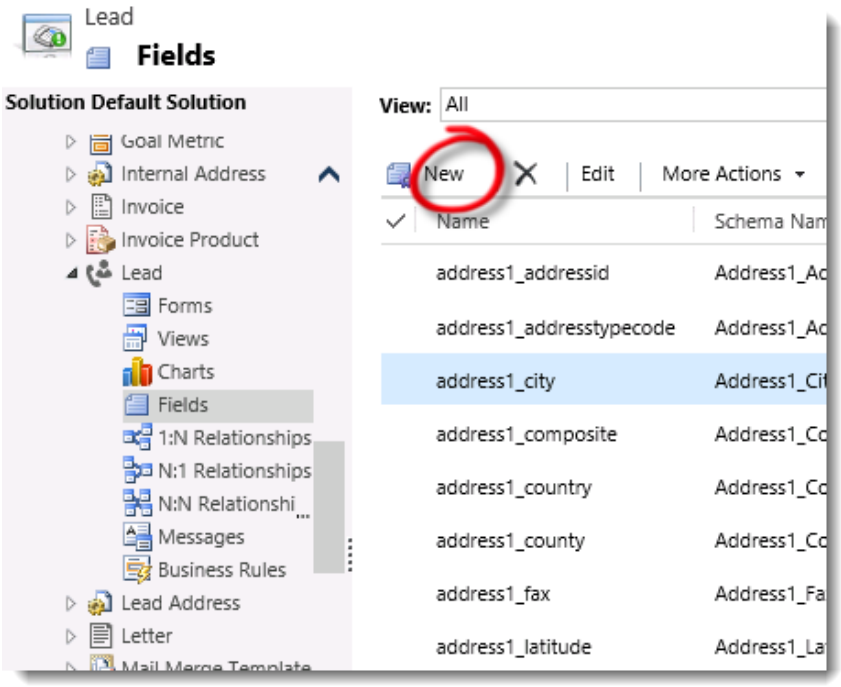
Scenario

A company wants leads to be numbered automatically by Microsoft Dynamics 365. The company regularly handles large quantities of leads and wants to be able to reference them quickly using a numbering system.

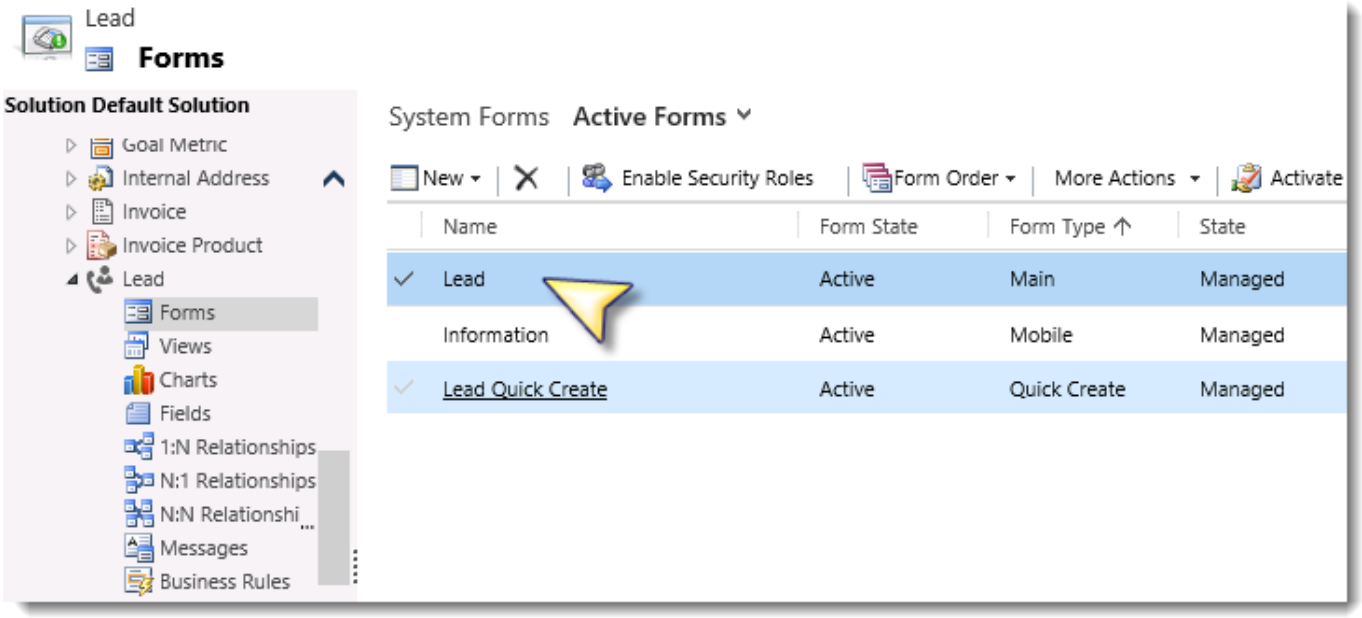
In this lab you will create a plug-in that implements this scenario.

Create customizations

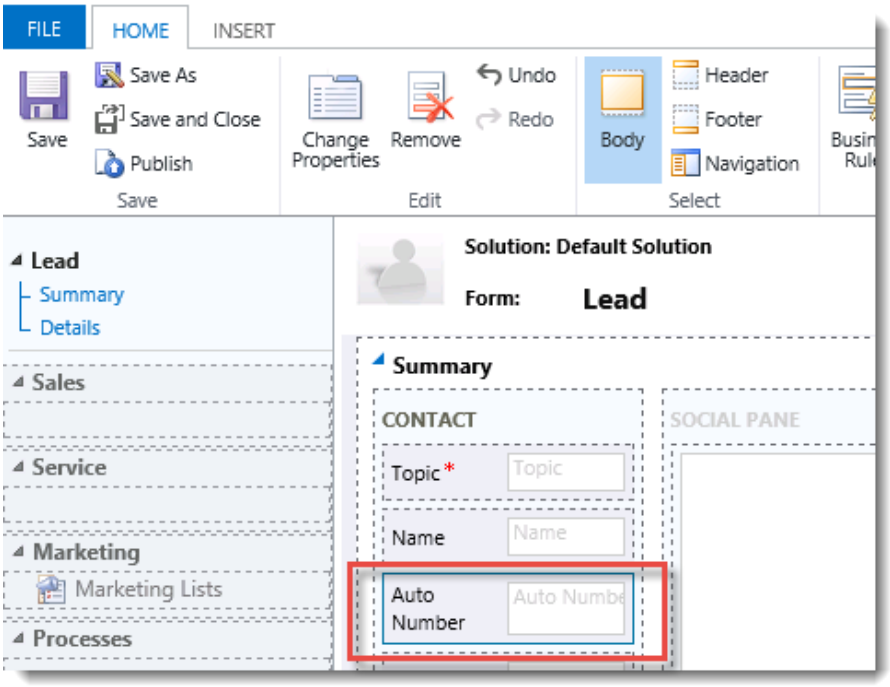
1. In **Microsoft Dynamics 365**, go to **Settings** > **Customizations**, and then click **Customize the System**.
2. In the **Navigation Pane**, expand **Entities**, expand **Lead**, select **Fields**, and then click **New**.



3. In the **Field: New for Lead** form, set the **Display Name** to **Auto Number**, the **Type** to **Whole Number**, and then click **Save and Close**.
4. In the **Navigation Pane**, click **Forms**.
5. Click the **Lead** main form.



6. Drag the **Auto Number** field to the **Contact** section of the **Summary** tab.

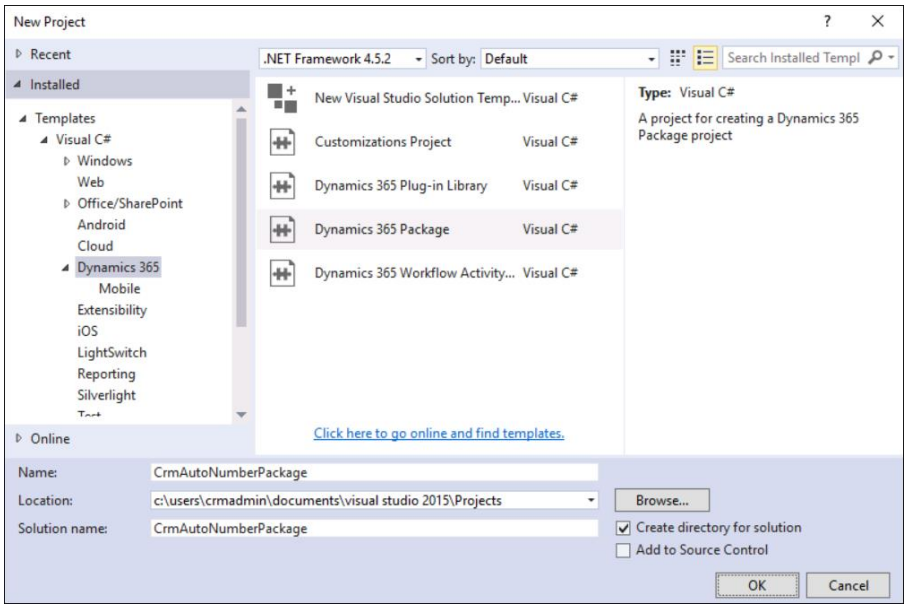


7. Select the **Auto Number** field, click **Change Properties** on the Ribbon, click the **Field is readonly** check box, and then click **OK**.
8. Click **Save and Close**, and then publish the **Lead** entity.

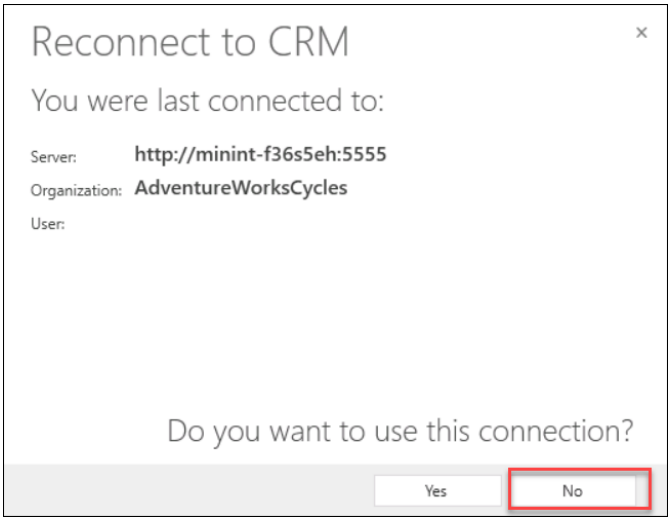
↙ Create a new Dynamics 365 Developer Toolkit project

1. Open **Visual Studio 2015**.
2. Click the **File** menu.
3. Click **New** and then click **Project**.
4. In the **New Project** dialog box under **Templates** > **Visual C#**, select the **Dynamics 365** group to view the available project templates.
5. Select **Dynamics 365 Package**.
6. Change the **Name** text box to **CrmAutoNumberPackage**.

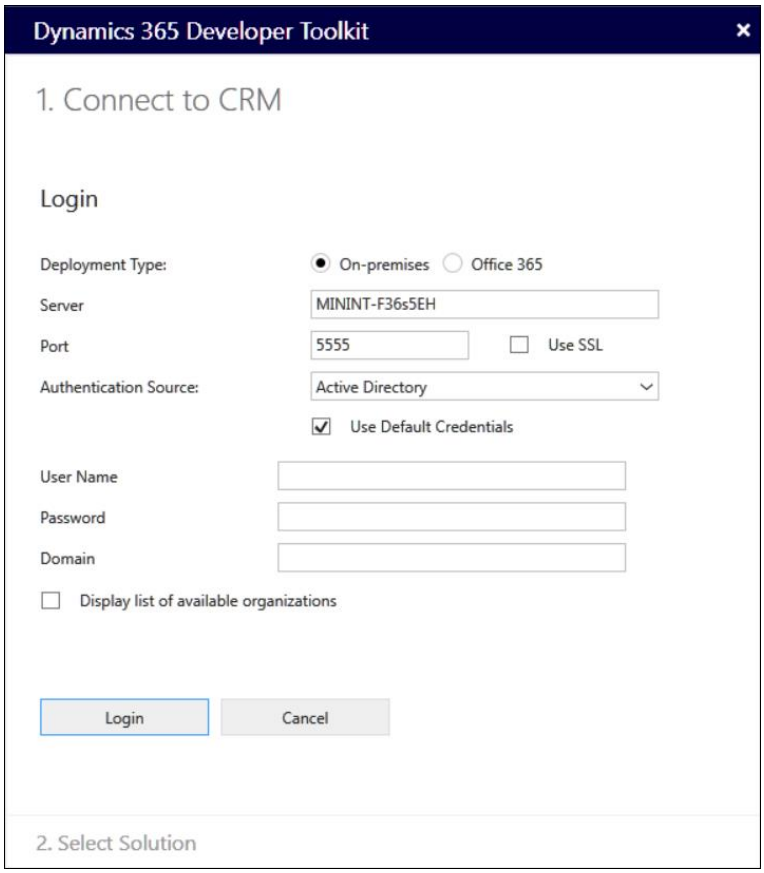
In your new solution, the CrmPackage project is a manifest project that contains all assets to be deployed to Microsoft Dynamics 365 combined with their deployment settings. By default, the outputs from each other project are added as references to this project in so that they can be deployed to the server.



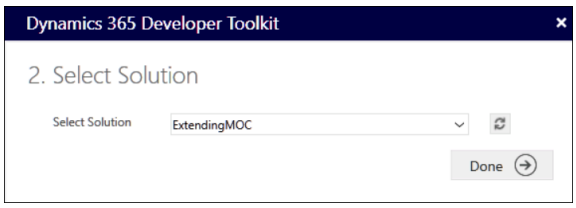
7. Click **OK**.
8. If the **Reconnect to CRM** dialog box is displayed, click **No**.



9. In the **Connect to CRM** dialog, in the **Server** textbox type **MININT-F36s5EH**.
10. In the **Port Number** textbox type **5555**.
11. In the **Authentication Source** list, make sure **Active Directory** is selected.
12. Select **Use Default Credentials**.



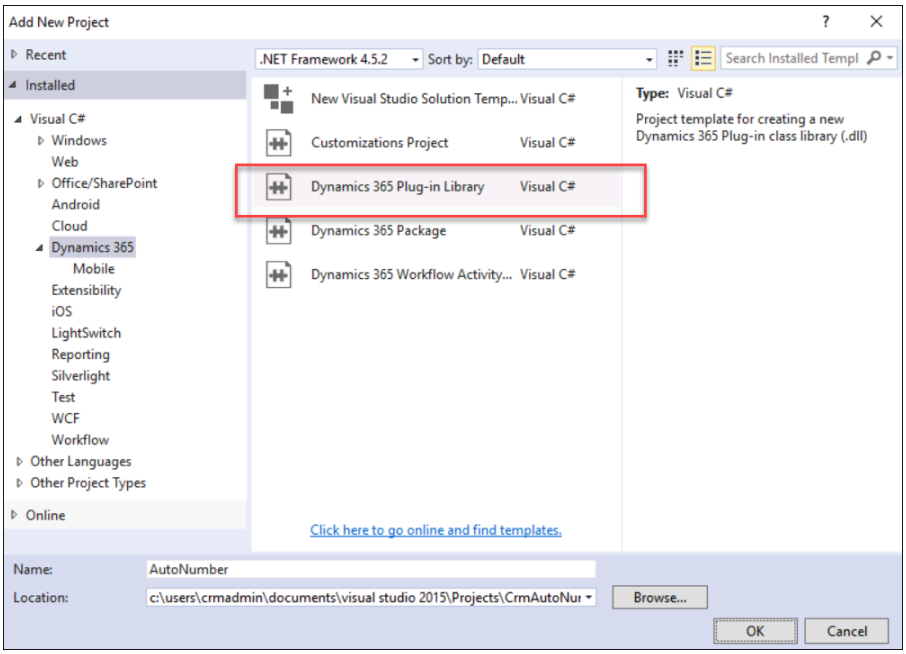
5) Click **Login**.
6) In the **Select Solution** dialog, in the **Select Solution** drop-down list select **ExtendingMOC**.



7) Click **Done**.

Add a new Dynamics 365 Workflow Library project

1. In **Solution Explorer**, right-click the solution node, select **Add** and then click **New Project**.
2. In the **Add New Project** dialog box under **Visual C#**, select the **Dynamics 365** group to view the available project templates.
3. Select **Dynamics 365 Plug-in Library**.
4. Change the **Name** text box to **AutoNumber**.



5. Click **OK**.

   The Plugins project contains a base class named Plugin that implements the Microsoft.Xrm.Sdk.IPlugin interface. All plug-in classes generated through the Create Plug-in option of the CRM Explorer will derive from this class. This class simplifies some of the common code that is required by many plug-ins, including the following:

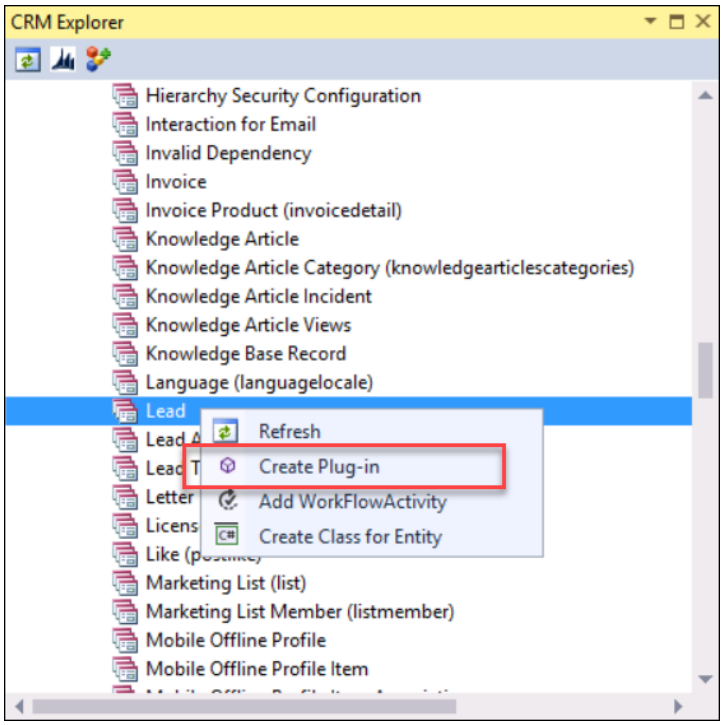
   - Obtaining execution context from the service provider.
   - Obtaining the tracing service from the service provider.
   - Instantiating the Organization Service proxy.

   Additionally, the validation method verifies several typical plug-in events and provides a pattern for the developer to extend.

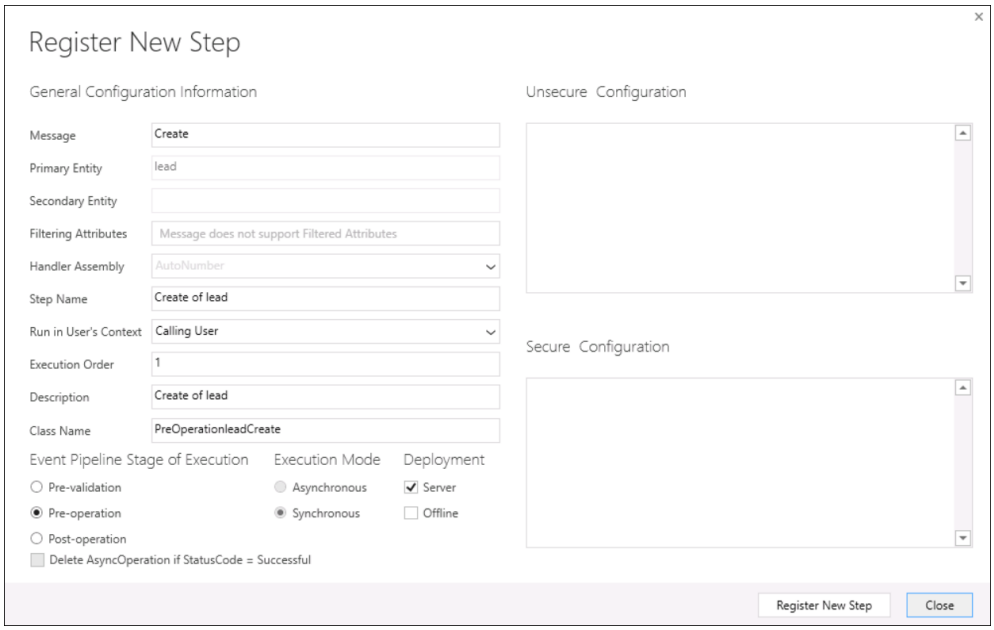
   Initially, the implementation of the derived class calls the implementation of the base class before continuing with its own, more specific implementation. After validating that the plug-in should execute, the base class calls up to the Execute method of the derived class, passing through the execution context and Organization Service proxy.

Create a Plug-in

1. In **CRM Explorer**, locate the **Lead** entity. Right-click the **Lead** entity and then select **Create Plug-in**.



2. The **Register New Step** dialog box appears.
3. In the **Register New Step** dialog box enter field values according to the following illustration:



4. In the **Register New Step** dialog box, click **OK** to generate the plug-in code.

   This action also updates the **RegisterFile.crmregister** in the CRM Package project to store the plug-in registration information.

5. In the **PreoperationleadCreate.cs** file, add the following **using** statement:

```
using Microsoft.Xrm.Sdk.Query;
```

6. Locate this comment // **TODO: Implement your custom Plug-in business logic**.
7. Add the following code below the comment:

```
// Obtain the execution context from the service provider.
IPluginExecutionContext context = localContext.PluginExecutionContext;

// The InputParameters collection contains all the data passed in the message request.
if (context.InputParameters.Contains("Target") &&
                context.InputParameters["Target"] is Entity)
{
    // Obtain the target entity from the input parameters.
    Entity entity = (Entity)context.InputParameters["Target"];

    QueryExpression qe = new QueryExpression("lead");

    // Retrieve new_autonumber field
    qe.ColumnSet = new ColumnSet();
    qe.ColumnSet.Columns.Add("new_autonumber");

     // Order records by autonumber descending (highest autonumber first)
     OrderExpression oe = new OrderExpression();
     oe.AttributeName = "new_autonumber";
     oe.OrderType = OrderType.Descending;
     qe.Orders.Add(oe);

     // Use paging to return only one record
     qe.PageInfo = new PagingInfo();
     qe.PageInfo.Count = 1;
     qe.PageInfo.PageNumber = 1;
     qe.PageInfo.PagingCookie = null;

     // Obtain the organization service reference.
     IOrganizationService service = localContext.OrganizationService;

     // Retrieve first page (one record) of lead records
     EntityCollection ec = service.RetrieveMultiple(qe);

     // Late bound entity
     Entity lead = (Entity)ec[0];

     int nextnumber = 1;  // default initial value

     // Calculate next autonumber value if existing value
     if(lead.Attributes.Contains("new_autonumber"))
             nextnumber = (int)lead.Attributes["new_autonumber"] + 1;

// Add autonumber to entity
     entity.Attributes.Add("new_autonumber", nextnumber);
}
```

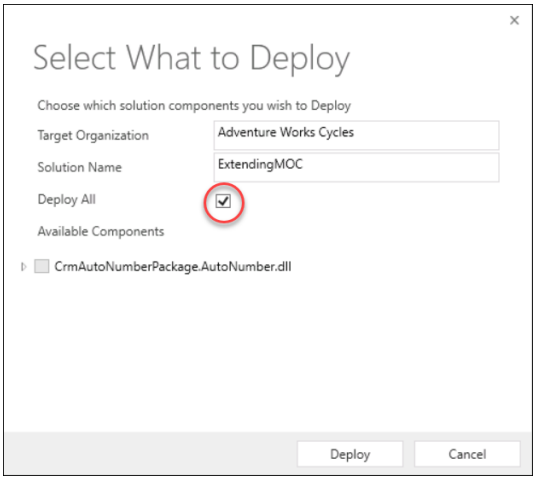
Strongly Sign the Assembly

1. Right-click the **AutoNumber** project, and then click **Properties**.
2. Click the **Signing** tab.
3. Check the **Sign the assembly** check box.

4. Select **<New...>** in the **Choose a strong name key file** drop-down list.
5. In the **Key File Name** textbox type **AutoNumber**.
6. Clear the **Protect my** key file with a password checkbox.
7. Click **OK**.
8. In **Visual Studio 2015**, click the **Build** menu and then click **Build Solution**.

Deploy the solution to the server

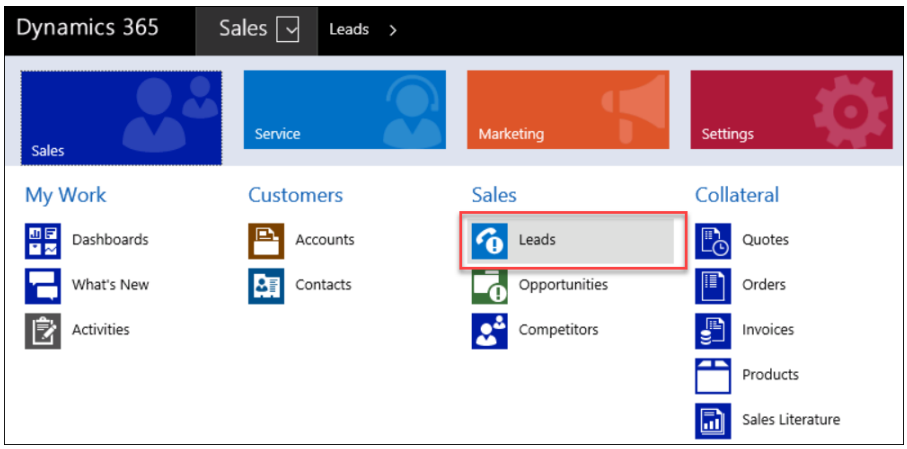
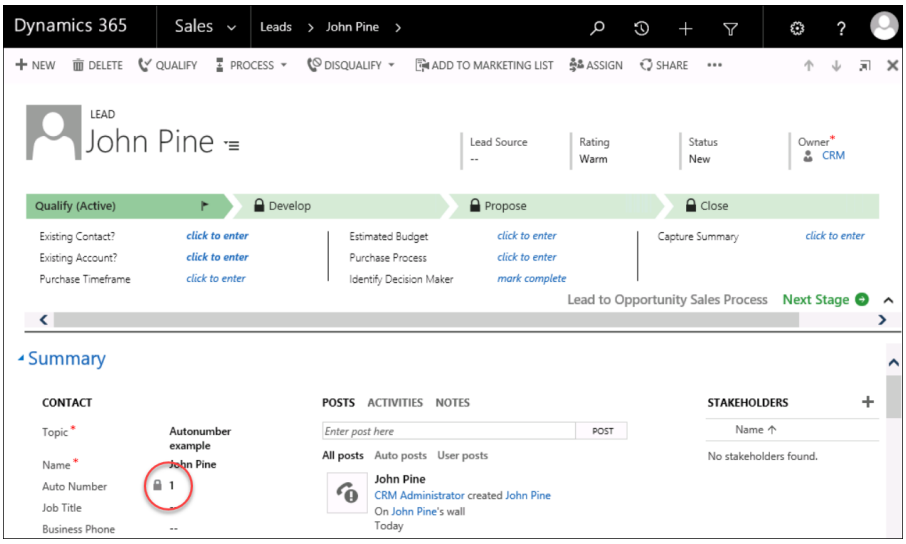
1. Right-click the **CrmAutoNumberPackage** project and select **Deploy**.
2. In the **Select What to Deploy** dialog box, select **Deploy All**.



3. Click **Deploy**.
4. Verify that the deployment was successful.

Create a new lead

1. In **Microsoft Dynamics 365**, go to **Sales** > **Leads**.



2. Click **New**.
3. In the **Summary** area of the **Lead** screen, enter the required fields under the **Contact** section.
4. Click **Save**.
5. After saving the new lead, confirm that a new value is set in the **Auto Number**.