**Generate Early Bound Types**

1. Run the CrmSvcUtil.exe tool, with the **Microsoft.Xrm.Client.CodeGeneration** extension, to generate your entity classes and service contexts. The following is an example command to create a file called Xrm.cs that points at an instance of Microsoft Dynamics CRM. Note that the Microsoft.Xrm.Client.CodeGeneration.dll file must be in the same directory as the CrmSvcUtil.exe file, or in the system GAC, when you run this command.

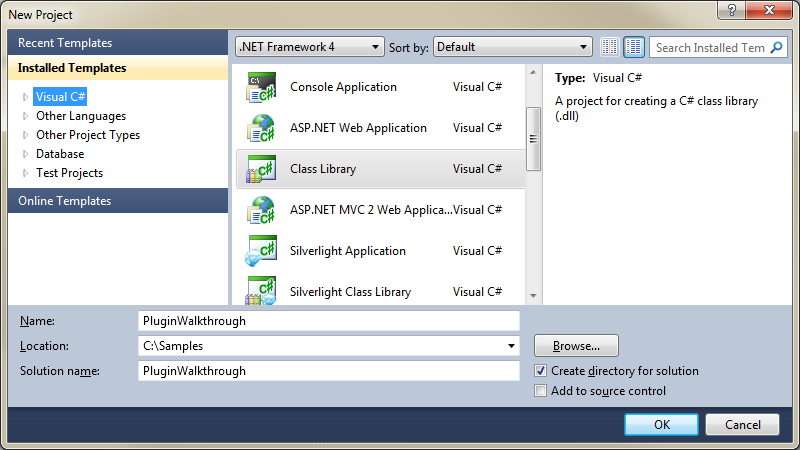
**msdos**

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_f1b2e6e4-da7a-4057-9aee-206af92ec0b5');" \o "Copy to clipboard.)

CrmSvcUtil.exe /codeCustomization:"Microsoft.Xrm.Client.CodeGeneration.CodeCustomization, Microsoft.Xrm.Client.CodeGeneration" /out:Xrm\Xrm.cs /url:http://Crm/Contoso/XRMServices/2011/Organization.svc /domain:CONTOSO /username:administrator /password:pass@word1 /namespace:Xrm /serviceContextName:XrmServiceContext

**Set up your plug-in project in Visual Studio**

1. Create a new class library project in Microsoft Visual Studio as shown here. This sample uses “Plugin” as the project name.



1. Add the following references from the SDK\bin folder.
   * Microsoft.Xrm.Client.dll
   * Microsoft.Xrm.Sdk.dll
2. Add the following .NET references.
   * Microsoft.IdentityModel.dll
   * System.Data.Services
   * System.Data.Services.Client
   * System.Runtime.Serialization
   * System.ServiceModel

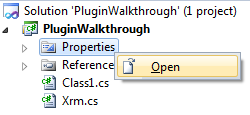
If you do not have the Microsoft.IdentityModel.dll file, you must install [Windows Identity Foundation](http://go.microsoft.com/fwlink/?LinkId=202021).

1. Right-click the project in Visual Studio, click **Add**, and then click **Existing Item**.
2. Select the “xrm.cs” file that you created when you generated the early bound types.
3. Right-click your project again, click **Add**, and then click **New Item**.
4. Select **Application Configuration File** from the options and then click **Add**.
5. Edit the configuration file with your specific connection string. For more information, see [Simplified Connection to Microsoft Dynamics CRM](http://msdn.microsoft.com/en-us/library/gg695810.aspx).

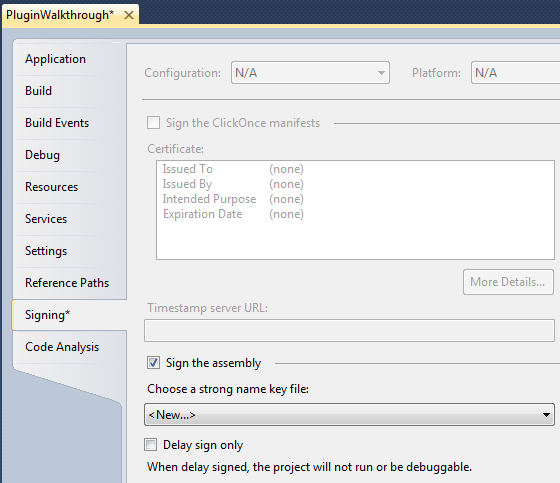
**Sign your plug-in project**

**Add a strong key to your project**

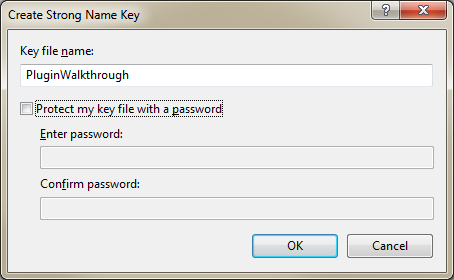
1. Open the properties pane under your plug-in project.



1. Create a new strong key file by clicking the **Signing** tab, select the **Sign the assembly** check box and select **<New…>** in the drop down list.



1. Type a name for your strong key (in this example, it is “PluginWalkthrough”) and clear the ”Protect my key file with a password” check box before clicking **OK**.

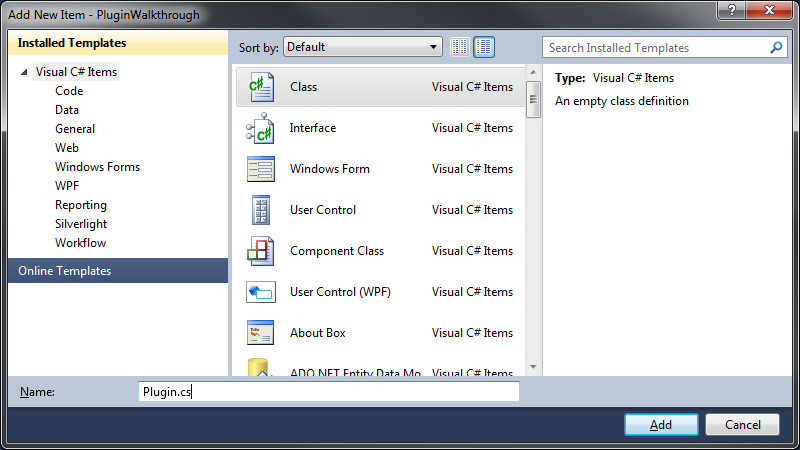


1. Save your changes.

**Create the plug-in that will run your code**

This plug-in will run when a contact is created. The following code shows how to create the plug-in.

1. Right-click your project again, click **Add**, and then click **New Item**.
2. Select **Class** from the options, type the name “Plugin.cs”, and then click **Add**.



1. Add the following code to the Plugins.cs file:

**C#**

[Copy](javascript:CodeSnippet_CopyCode('CodeSnippetContainerCode_a279599a-bbc7-45f6-8c1b-06636161d944');)

using System;

using System.Diagnostics;

using System.Linq;

using System.ServiceModel;

using Microsoft.Xrm.Sdk;

using Xrm;

public class Plugin: IPlugin

{

public void Execute(IServiceProvider serviceProvider)

{

IPluginExecutionContext context = (IPluginExecutionContext)

serviceProvider.GetService(typeof(IPluginExecutionContext));

Entity entity;

// Check if the input parameters property bag contains a target

// of the create operation and that target is of type Entity.

if (context.InputParameters.Contains("Target") &&

context.InputParameters["Target"] is Entity)

{

// Obtain the target business entity from the input parameters.

entity = (Entity)context.InputParameters["Target"];

// Verify that the entity represents a contact.

if (entity.LogicalName != "contact") { return; }

}

else

{

return;

}

try

{

IOrganizationServiceFactory serviceFactory =

(IOrganizationServiceFactory)serviceProvider.GetService(

typeof(IOrganizationServiceFactory));

IOrganizationService service =

serviceFactory.CreateOrganizationService(context.UserId);

var id = (Guid)context.OutputParameters["id"];

AddNoteToContact(service, id);

}

catch (FaultException<OrganizationServiceFault> ex)

{

throw new InvalidPluginExecutionException(

"An error occurred in the plug-in.", ex);

}

}

private static void AddNoteToContact(IOrganizationService service, Guid id)

{

using (var crm = new XrmServiceContext(service))

{

var contact = crm.ContactSet.Where(

c => c.ContactId == id).First();

Debug.Write(contact.FirstName);

var note = new Annotation

{

Subject = "Created with plugin",

NoteText = "This Note was created by the example plug-in",

ObjectId = contact.ToEntityReference(),

ObjectTypeCode = contact.LogicalName

};

crm.AddObject(note);

crm.SaveChanges();

}

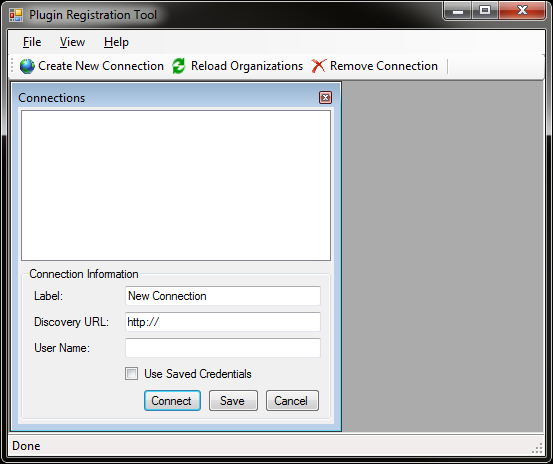
}

}

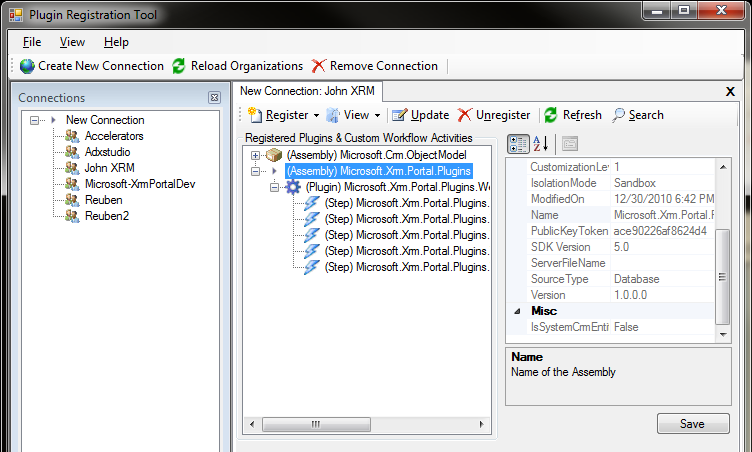
1. Build the solution.

**Register your plug-in with the Plugin Registration Tool**

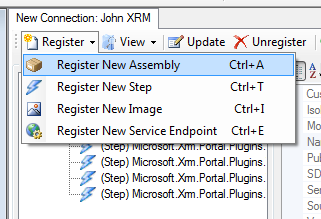
1. Add all dependent assemblies to the GAC on the server. To do this, open your project’s Debug/bin folder and copy all \*.dll files except for the main PluginWalkthrough.dll file to the GAC on the server.
2. Run the Plugin Registration Tool. If you have not already built this tool, build the Plug-in Registration Tool according to the instructions in its Readme file, located at SDK\Tools\PluginRegistration\Readme.docx.
3. Click **Create New Connection**.



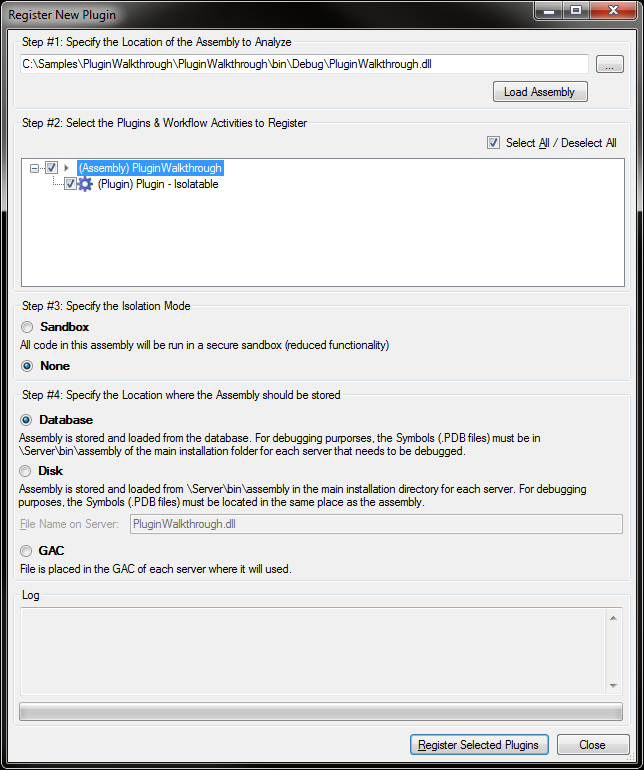
1. In the **Connections** panel, enter a descriptive label for the connection. Fill in the other fields as appropriate for your Microsoft Dynamics CRM server.
2. Click **Connect**. A connection to the server is established and a list of available organizations for the specified system account is displayed.
3. Double-click the desired organization in the connections list. The list of all assemblies, steps, and plug-ins currently registered for the target organization is displayed.



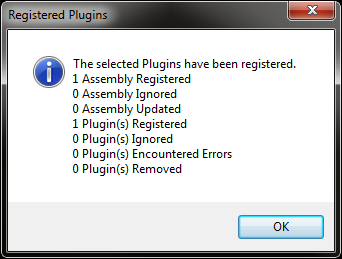
1. Click **Register New Assembly**.



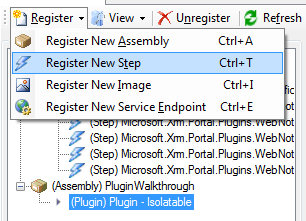
1. In the **Register New Plugin** dialog box, click the ellipsis button **(…)** and navigate to the location of your plug-in assembly. Select the assembly and make sure that all plug-ins under it are selected.
2. Select **None** for the isolation mode. Note that Developer Extensions for Microsoft Dynamics CRM currently does not support the sandbox isolation mode. Verify that the **Database** option is selected so that the plug-in will be stored in your organization’s database.
3. Click **Register Selected Plugins**.



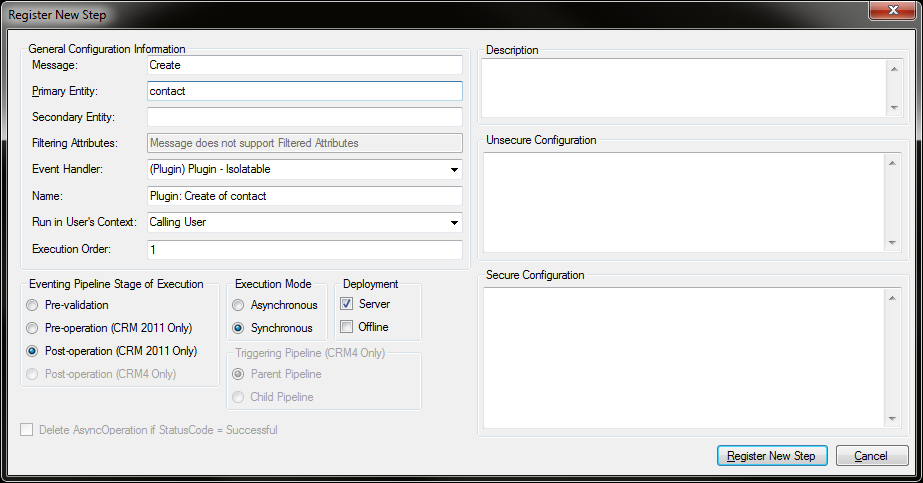
1. Click **OK**.



1. Expand the assembly and select the plug-in that you just registered. Select **Register**, and then click **Register New Step**.



1. In the **Register New Step** dialog box, enter the Microsoft Dynamics CRM message that the step will be registered under. In this example, type “Create”. Under **Primary Entity**, type “contact”. Leave the **Pipeline** stage set to **Post Stage**, execution mode to **Synchronous**, and other options set to default. Click **Register New Step**.



The registered step appears under the plug-in.

Step registered successfully

1. Test the plug-in by creating a new contact in Microsoft Dynamics CRM. After you have saved the entity, a note should be attached.