**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.

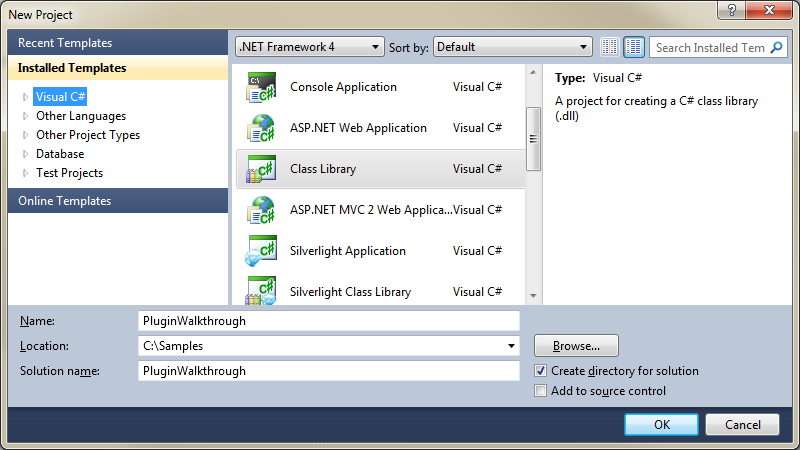
**msdos**

[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

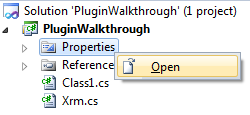
If you do not have the Microsoft.IdentityModel.dll file, you must install [Windows Identity Foundation](http://go.microsoft.com/fwlink/?LinkId=202021).

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).

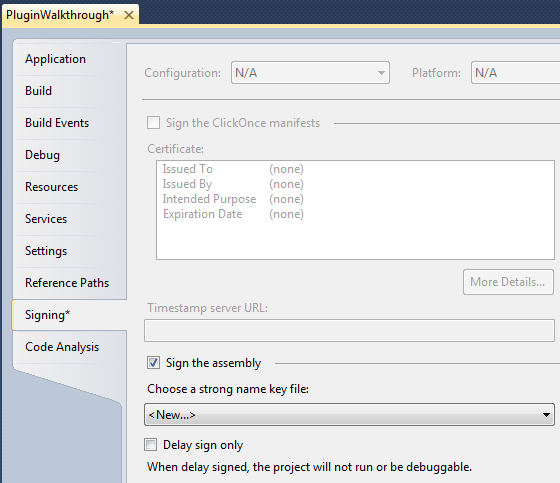
**Sign your plug-in project**

**Add a strong key to your project**

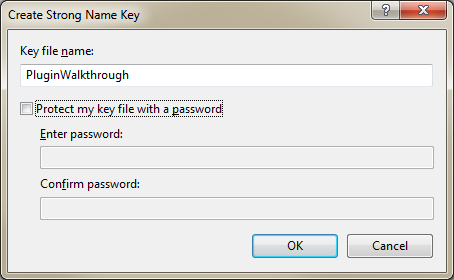
1. Open the properties pane under your plug-in project.



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**.

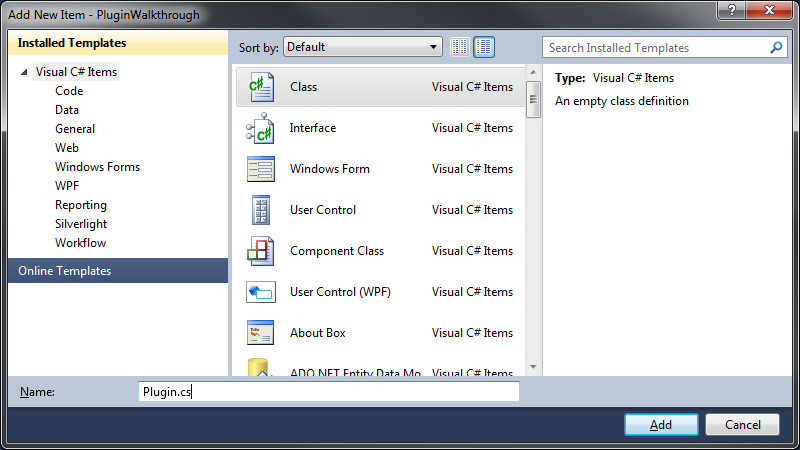


1. Save your changes.

**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;

usingSystem.Diagnostics;

usingSystem.Linq;

usingSystem.ServiceModel;

usingMicrosoft.Xrm.Sdk;

usingXrm;

publicclass Plugin: IPlugin

{

publicvoid Execute(IServiceProviderserviceProvider)

{

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

{

IOrganizationServiceFactoryserviceFactory =

(IOrganizationServiceFactory)serviceProvider.GetService(

typeof(IOrganizationServiceFactory));

IOrganizationService service =

serviceFactory.CreateOrganizationService(context.UserId);

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

AddNoteToContact(service, id);

}

catch (FaultException<OrganizationServiceFault> ex)

{

thrownewInvalidPluginExecutionException(

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

}

}

privatestaticvoidAddNoteToContact(IOrganizationService service, Guid id)

{

using (var crm = newXrmServiceContext(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();

}

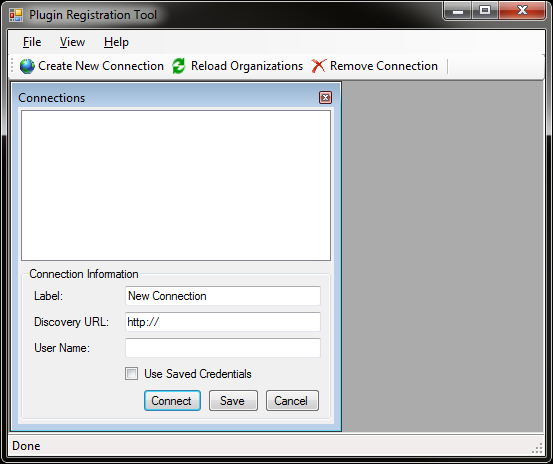
}

}

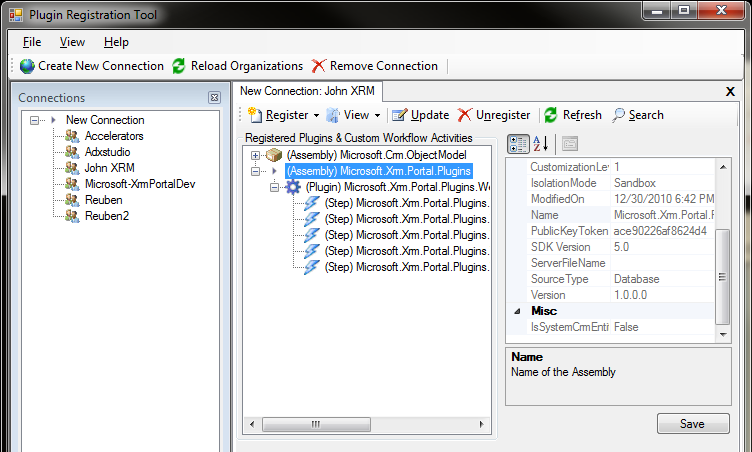
1. Build the solution.

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

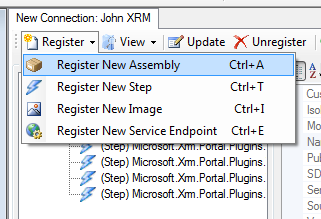
1. Add all dependent assemblies to the GAC on the server. To do this, open your project’s Debug/bin folder and copy all \*.dll files except for the main PluginWalkthrough.dll file to the GAC on the server.
2. Run the Plugin Registration Tool. If you have not already built this tool, build the Plug-in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



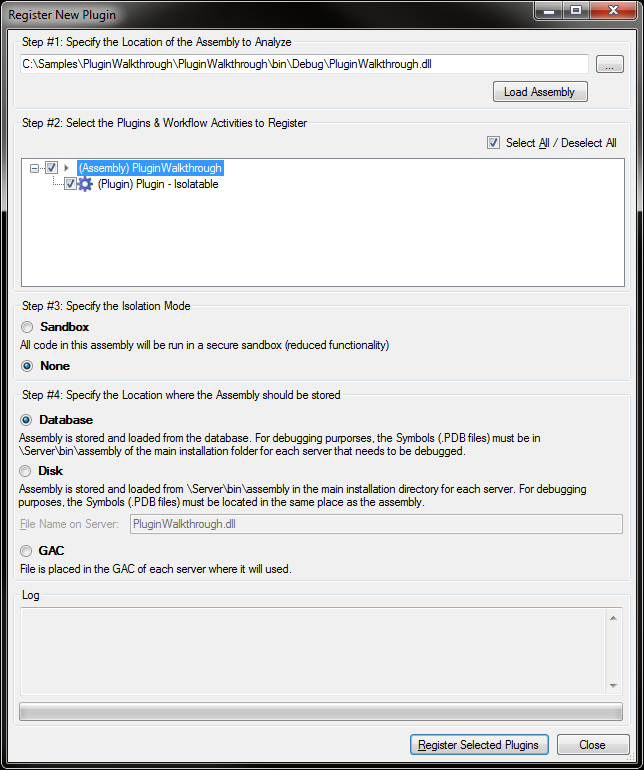
1. In the **Connections** panel, enter a descriptive label for the connection. Fill in the other fields as appropriate for your Microsoft Dynamics CRM server.
2. Click **Connect**. A connection to the server is established and a list of available organizations for the specified system account is displayed.
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.



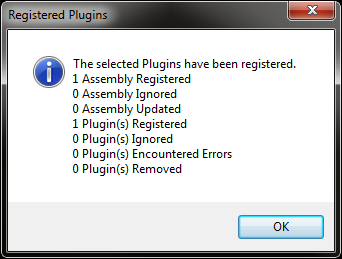
1. Click **Register New Assembly**.



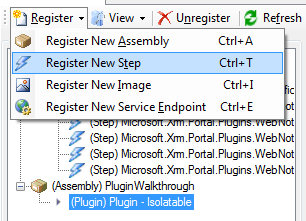
1. In the **Register New Plugin** dialog box, click the ellipsis button **(…)** and navigate to the location of your plug-in assembly. Select the assembly and make sure that all plug-ins under it are selected.
2. Select **None** for the isolation mode. Note that Developer Extensions for Microsoft Dynamics 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**.



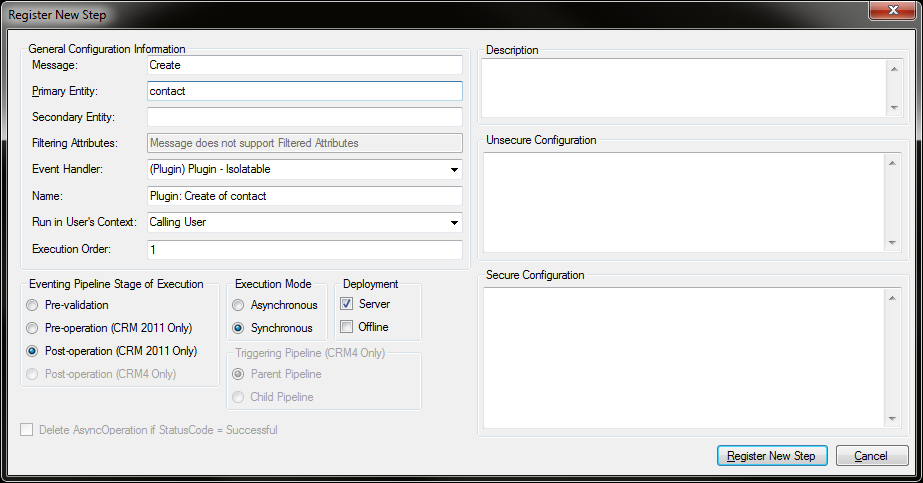
1. Click **OK**.



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.

You can use .NET Language-Integrated Query (LINQ) to write queries in Microsoft Dynamics CRM Online 2016 Update and Microsoft Dynamics CRM 2016 (on-premises). You can use the [OrganizationServiceContext](https://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationservicecontext.aspx) class or a deriving class created by the CrmSvcUtil tool to write [LINQ](https://msdn.microsoft.com/library/bb397897.aspx) queries that access the SOAP endpoint (Organization.svc). The [OrganizationServiceContext](https://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationservicecontext.aspx) class contains an underlying LINQ query provider that translates LINQ queries from Microsoft Visual C# or Microsoft Visual Basic .NET syntax into the query API used by Microsoft Dynamics CRM.

When you use early-bound programming classes you can generate a class derived from the [OrganizationServiceContext](https://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationservicecontext.aspx) class if you specify the name of the class using the servicecontextname parameter when using the Code Generation Tool (CrmSvcUtil.exe). Use of this class allows for referencing an [IQueryable](https://msdn.microsoft.com/library/system.linq.iqueryable.aspx) entity set using the pattern <entity schema name>+Set, for example AccountSet to reference the collection of **Account** entity records.

Examples :

[Use LoadProperty to retrieve related records](https://msdn.microsoft.com/en-us/library/gg509017.aspx#BKMK_UseLoadProperty)

// Retrieve all accounts owned by the user who has read access rights

// to the accounts and where the last name of the user is not Cannon.

var queryAccounts = from a in svcContext.AccountSet

join owner in svcContext.SystemUserSet

on a.OwnerId.Id equals owner.SystemUserId

where owner.LastName != "Cannon"

select new Account

{

Name = a.Name,

Address1\_City = a.Address1\_City

};

## [Simple Where clause](javascript:void(0))

The following sample shows how to retrieve a list of accounts where the Name contains “Contoso”.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_55878245-2a95-4e4f-8d18-0100341205af');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_where1 = from a in svcContext.AccountSet

where a.Name.Contains("Contoso")

select a;

foreach (var a in query\_where1)

{

System.Console.WriteLine(a.Name + " " + a.Address1\_City);

}

}

The following sample shows how to retrieve a list of accounts where the Name contains “Contoso” and Address1\_City is “Redmond”.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_8fc8ee5c-302d-419a-a29d-c671c5ca72b4');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_where2 = from a in svcContext.AccountSet

where a.Name.Contains("Contoso")

where a.Address1\_City == "Redmond"

select a;

foreach (var a in query\_where2)

{

System.Console.WriteLine(a.Name + " " + a.Address1\_City);

}

}

## [Join and simple Where clause](javascript:void(0))

The following sample shows how to retrieve the account Name and the contact LastName where the account Name contains “Contoso” and the contact LastName contains “Smith” and the contact is the Primary Contact for the account.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_65ed4273-3b70-4de4-bfe3-009080c9b4eb');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_where3 = from c in svcContext.ContactSet

join a in svcContext.AccountSet

on c.ContactId equals a.PrimaryContactId.Id

where a.Name.Contains("Contoso")

where c.LastName.Contains("Smith")

select new

{

account\_name = a.Name,

contact\_name = c.LastName

};

foreach (var c in query\_where3)

{

System.Console.WriteLine("acct: " +

c.account\_name +

"\t\t\t" +

"contact: " +

c.contact\_name);

}

}

## [Use the Distinct Operator](javascript:void(0))

The following sample shows how to retrieve a distinct list of contact last names. Although there may be duplicates, each name will be listed only once.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_25fc7e02-d9f2-47da-ab81-bdd1aeb02835');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_distinct = (from c in svcContext.ContactSet

select c.LastName).Distinct();

foreach (var c in query\_distinct)

{

System.Console.WriteLine(c);

}

}

## [Simple inner join](javascript:void(0))

The following sample shows how to retrieve information about an account and the contact listed as the primary contact for the account.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b32eb72c-53ed-42a8-aa97-a234faf61022');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_join1 = from c in svcContext.ContactSet

join a in svcContext.AccountSet

on c.ContactId equals a.PrimaryContactId.Id

select new

{

c.FullName,

c.Address1\_City,

a.Name,

a.Address1\_Name

};

foreach (var c in query\_join1)

{

System.Console.WriteLine("acct: " +

c.Name +

"\t\t\t" +

"contact: " +

c.FullName);

}

}

## [Self-join](javascript:void(0))

The following sample shows how to retrieve information about accounts where an account is the parent account for an account.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_e732349d-0428-4a45-8caf-11e5144e77ba');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_join5 = from a in svcContext.AccountSet

join a2 in svcContext.AccountSet

on a.ParentAccountId.Id equals a2.AccountId

select new

{

account\_name = a.Name,

account\_city = a.Address1\_City

};

foreach (var c in query\_join5)

{

System.Console.WriteLine(c.account\_name + " " + c.account\_city);

}

}

## [Double and multiple joins](javascript:void(0))

The following sample shows how to retrieve information from account, contact and lead where the contact is the primary contact for the account and the lead was the originating lead for the account.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-7)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_16de105d-319a-40fc-8b27-b160e1f75a91');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_join4 = from a in svcContext.AccountSet

join c in svcContext.ContactSet

on a.PrimaryContactId.Id equals c.ContactId

join l in svcContext.LeadSet

on a.OriginatingLeadId.Id equals l.LeadId

select new

{

contact\_name = c.FullName,

account\_name = a.Name,

lead\_name = l.FullName

};

foreach (var c in query\_join4)

{

System.Console.WriteLine(c.contact\_name +

" " +

c.account\_name +

" " +

c.lead\_name);

}

}

The following sample shows how to retrieve account and contact information where an account is the parent account for an account and the contact is the primary contact for the account.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-8)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b6a24ba4-e3ba-42a8-ad95-584555fa3b25');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_join6 = from c in svcContext.ContactSet

join a in svcContext.AccountSet

on c.ContactId equals a.PrimaryContactId.Id

join a2 in svcContext.AccountSet

on a.ParentAccountId.Id equals a2.AccountId

select new

{

contact\_name = c.FullName,

account\_name = a.Name

};

foreach (var c in query\_join6)

{

System.Console.WriteLine(c.contact\_name + " " + c.account\_name);

}

}

## [Join using entity fields](javascript:void(0))

The following sample shows how to retrieve information about accounts from a list

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-9)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_678ccda9-cf21-430d-9199-b20b86af9284');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var list\_join = (from a in svcContext.AccountSet

join c in svcContext.ContactSet

on a.PrimaryContactId.Id equals c.ContactId

where a.Name == "Contoso Ltd" &amp;&amp;

a.Address1\_Name == "Contoso Pharmaceuticals"

select a).ToList();

foreach (var c in list\_join)

{

System.Console.WriteLine("Account " + list\_join[0].Name

+ " and it's primary contact "

+ list\_join[0].PrimaryContactId.Id);

}

}

## [Late-binding left join](javascript:void(0))

The following sample shows a left join. A left join is designed to return parents with and without children from two sources. There is a correlation between parent and child, but no child may actually exist.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-10)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_3bd9cbcf-a2ab-4da7-a58b-139af7ebac12');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_join8 = from a in svcContext.AccountSet

join c in svcContext.ContactSet

on a.PrimaryContactId.Id equals c.ContactId

into gr

from c\_joined in gr.DefaultIfEmpty()

select new

{

contact\_name = c\_joined.FullName,

account\_name = a.Name

};

foreach (var c in query\_join8)

{

System.Console.WriteLine(c.contact\_name + " " + c.account\_name);

}

}

## [Use the Equals operator](javascript:void(0))

The following sample shows how to retrieve a list of contacts where the FirstName is “Colin”.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-11)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_3c8db447-818f-4469-83ed-67e6d7c08f3e');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_equals1 = from c in svcContext.ContactSet

where c.FirstName.Equals("Colin")

select new

{

c.FirstName,

c.LastName,

c.Address1\_City

};

foreach (var c in query\_equals1)

{

System.Console.WriteLine(c.FirstName +

" " + c.LastName +

" " + c.Address1\_City);

}

}

The following sample shows how to retrieve a list of contacts where the FamilyStatusCode is 3. This corresponds to the Marital Status option of Divorced.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-12)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d131f6c1-c04f-4224-ba16-0cd01ed3e3d8');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_equals2 = from c in svcContext.ContactSet

where c.FamilyStatusCode.Equals(3)

select new

{

c.FirstName,

c.LastName,

c.Address1\_City

};

foreach (var c in query\_equals2)

{

System.Console.WriteLine(c.FirstName +

" " + c.LastName +

" " + c.Address1\_City);

}

}

## [Use the Not Equals operator](javascript:void(0))

The following sample shows how to retrieve a list of contacts where the Address1\_City is not “Redmond”.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-13)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1c1908f2-2834-423b-8b8b-0c085f297c89');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_ne1 = from c in svcContext.ContactSet

where c.Address1\_City != "Redmond"

select new

{

c.FirstName,

c.LastName,

c.Address1\_City

};

foreach (var c in query\_ne1)

{

System.Console.WriteLine(c.FirstName + " " +

c.LastName + " " + c.Address1\_City);

}

}

The following sample shows how to retrieve a list of contacts where the FirstName is not “Colin”.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg509017.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-14)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_42efdc92-1602-48f3-9b79-e149405a1c0d');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_ne2 = from c in svcContext.ContactSet

where !c.FirstName.Equals("Colin")

select new

{

c.FirstName,

c.LastName,

c.Address1\_City

};

foreach (var c in query\_ne2)

{

System.Console.WriteLine(c.FirstName + " " +

c.LastName + " " + c.Address1\_City);

}

}

# Use LINQ to construct a query

**Dynamics CRM 2016**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635955913322383505

* [Dynamics CRM 2015](https://msdn.microsoft.com/en-us/library/gg328328(v=crm.7).aspx)
* [Dynamics CRM 2013](https://msdn.microsoft.com/en-us/library/gg328328(v=crm.6).aspx)
* [Dynamics CRM 2011](https://msdn.microsoft.com/en-us/library/gg328328(v=crm.5).aspx)

Applies To: CRM 2016 on-prem, CRM Online

The .NET Language-Integrated Query (LINQ) query provider in Microsoft Dynamics CRM Online 2016 Update and Microsoft Dynamics CRM 2016 (on-premises) uses standard LINQ syntax. The first step in creating a LINQ query is to identify the relevant entity types and the relationships between them. You can then specify the data source and the other query parameters.

The **from** clause is used to return a single “root” entity. The query provider can only return entities of a single entity type. The **orderby** and **select** clauses must reference this root entity. You can use **join** clauses to add entities with a relationship to the “root” entity.

## [In This Topic](javascript:void(0))

[LINQ operators](https://msdn.microsoft.com/en-us/library/gg328328.aspx#bkmk_operators)

[LINQ limitations](https://msdn.microsoft.com/en-us/library/gg328328.aspx#limitations)

[Filter multiple entities](https://msdn.microsoft.com/en-us/library/gg328328.aspx#filter)

## [LINQ operators](javascript:void(0))

All LINQ query expressions have a similar format. The following table shows the most common clauses in a LINQ query expression when using the Microsoft Dynamics CRM LINQ query provider.

|  |  |  |
| --- | --- | --- |
| **LINQ Operator** | **Description** | **Example** |
| Obtain a data source (the **from** clause) | When using the generated service context and early binding, use the **IQueryable** entity set, such as **AccountSet**, in the generated context.  When not using the generated context, the **CreateQuery** method on the organization service context object gives you access to Microsoft Dynamics CRM entities. | Using the generated service context:  C#  [Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f463f57a-6e05-47a5-a336-46784df6f728');)  var query1 = from c in context.ContactSet  select c;  Using the **CreateQuery** method:  C#  [Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_a7751939-86ce-4585-9d3a-59a5eaa84fa9');)  var query1 = from c in context.CreateQuery<Contact>()  select c; |
| **join** clause | The **join** clause represents an inner join. You use the clause to work with two or more entities that can be joined with a common attribute value. | C#  [Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b5ac3204-e86b-4892-b47e-45351f5a5ecf');)  from c in context.ContactSet  join a in context.AccountSet on c.ContactId equals a.PrimaryContactId.Id |
| Filtering (the **where** clause) | The **where** clause applies a filter to the results, often using a Boolean expression. The filter specifies which elements to exclude from the source sequence. Each **where** clause can only contain conditions against a single entity type. A composite condition involving multiple entities is not valid. Instead, each entity should be filtered in separate **where** clauses. | C#  [Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0ce5a2fb-c377-4968-9cf5-0f69b7b9c817');)  from a in context.AccountSet  where (a.Name.StartsWith("Contoso") && a.Address1\_StateOrProvince == "WA") |
| **orderby** | The **orderby** operator puts the returned query attributes in a specified order. | C#  [Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_d83911c0-6ff7-4ff4-9fd9-70d9430d3a1c');)  var query1 = from c in context.CreateQuery<Contact>()  orderby c.FullName ascending  select c;  foreach ( var q in query1)  {  Console.WriteLine(q.FirstName + " " + q.LastName);  } |
| **select** clause | The **select** clause defines the form of the data returned. The clause creates a column set based on the query expression results. You can also define an instance of a new object to work with. The newly created object using the **select** clause is not created on the server, but is a local instance. | C#  [Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_bca5f435-6e08-4ca6-85a1-1e96b79e2cf6');)  select new Contact  {  ContactId = c.ContactId,  FirstName = c.FirstName,  LastName = c.LastName,  Address1\_Telephone1 = c.Address1\_Telephone1  }; |

## [LINQ limitations](javascript:void(0))

The LINQ query provider supports a subset of the LINQ operators. Not all conditions that can be expressed in LINQ are supported. The following table shows some of the limitations of the basic LINQ operators.

|  |  |
| --- | --- |
| **LINQ Operator** | **Limitations** |
| **join** | Represents an inner or outer join. Only left outer joins are supported. |
| **from** | Supports one **from** clause per query. |
| **where** | The left side of the clause must be an attribute name and the right side of the clause must be a value. You cannot set the left side to a constant. Both the sides of the clause cannot be constants.  Supports the **String** functions **Contains**, **StartsWith**, **EndsWith**, and **Equals**. |
| **groupBy** | Not supported. FetchXML supports grouping options that are not available with the LINQ query provider. More information: [Use FetchXML aggregation](https://msdn.microsoft.com/en-us/library/gg309565.aspx) |
| **orderBy** | Supports ordering by entity attributes, such as **Contact.FullName**. |
| **select** | Supports anonymous types, constructors, and initializers. |
| **last** | The **last** operator is not supported. |
| **skip** and **take** | Supports **skip** and **take** using server-side paging. The **skip** value must be greater than or equal to the **take** value. |
| **aggregate** | Not supported. FetchXML supports aggregation options that are not available with the LINQ query provider. More information: [Use FetchXML aggregation](https://msdn.microsoft.com/en-us/library/gg309565.aspx) |

## [Filter multiple entities](javascript:void(0))

You can create complex .NET Language-Integrated Query (LINQ) queries in Microsoft Dynamics CRM 2016 and Microsoft Dynamics CRM Online. You use multiple **Join** clauses with filter clauses to create a result that is filtered on attributes from several entities.

The following sample shows how to create a LINQ query that works with two entities and filters the result based on values from each of the entities.

C#

[VB](https://msdn.microsoft.com/en-us/library/gg328328.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-7)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_1b132782-3ded-4476-8287-068755899a90');)

using (ServiceContext svcContext = new ServiceContext(\_serviceProxy))

{

var query\_where3 = from c in svcContext.ContactSet

join a in svcContext.AccountSet

on c.ContactId equals a.PrimaryContactId.Id

where a.Name.Contains("Contoso")

where c.LastName.Contains("Smith")

select new

{

account\_name = a.Name,

contact\_name = c.LastName

};

foreach (var c in query\_where3)

{

System.Console.WriteLine("acct: " +

c.account\_name +

"\t\t\t" +

"contact: " +

c.contact\_name);

}

}