Using LINQ Queries with CRM 2013 and CRM 2011

For those of us who have had the unfortunate experience of trying to access CRM data using SOAP queries, it is a luxury to be able to query CRM entities using LINQ.

It is pretty easy to setup your .NET project to use CRM and LINQ. Simply install the CRM SDK. This link is to the CRM 2013 SDK, but you can certainly do this in CRM 2011 as well:  
<http://www.microsoft.com/en-us/download/details.aspx?id=40321>

Once you have extracted the CRM SDK, you can use the CrmSvcUtil command line function to generate an xrm.cs file to include in your project, which generates the CRM entity definitions.  
<http://msdn.microsoft.com/en-us/library/ff681563.aspx>

1. Open the Command Prompt as an Administrator.
2. Change directories to the SDK directory which contains the CrmSvcUtil executable.

cd c:\CRM-2013-SDK\SDK\Bin

1. Type the CrmSvcUtil.exe command with the following parameters:
   1. /out – [name of the file you want generated]
   2. /url – [link to the organization service of your CRM implementation]
   3. /domain – [domain name of the CRM administrator account you will use to generate the file]
   4. /username – [user name of the CRM administrator account you will use to generate the file]
   5. /password – [password of the CRM administrator account you will use to generate the file]

CrmSvcUtil.exe /codeCustomization:"Microsoft.Xrm.Client.CodeGeneration.CodeCustomization, Microsoft.Xrm.Client.CodeGeneration" /out:Xrm.cs /url:https://crm.mycrmserver.com/myorganization/XRMServices/2011/Organization.svc /domain:mydomain /username:mycrmadministrator /password:mycrmadminpassword /namespace:Xrm /serviceContextName:XrmServiceContext

Once the xrm.cs file has been generated, you can include it as a class in your .Net project. Then you can create a context object:

private void RefreshContext()

{

if (context != null)

{

context.Dispose();

}

ClientCredentials credentials = new ClientCredentials();

credentials.UserName.UserName = MscrmWebService.LoginName;

credentials.UserName.Password = MscrmWebService.Password;

string link = isDev ? MscrmWebService.LinkDev : MscrmWebService.Link;

Uri organizationUri = new Uri(MscrmWebService.Link);

Uri homeRealmUri = null;

using (serviceProxy = new OrganizationServiceProxy(organizationUri, homeRealmUri, credentials, null))

{

serviceProxy.ServiceConfiguration.CurrentServiceEndpoint.Behaviors.Add(new ProxyTypesBehavior());

service = (IOrganizationService)serviceProxy;

context = new XrmServiceContext(service);

}

}

And from there you can query the CRM entities like you would any other LINQ entity. Some examples include:

**Retrieve CRM Contact by Id**

public Contact RetrieveContact(Guid contactId)

{

RefreshContext();

return context.ContactSet.Where(e => e.ContactId == contactId).Single();

}

**Retrieve CRM Contact by Name**

public Contact RetrieveContact(string firstName, string lastName)

{

RefreshContext();

return context.ContactSet.Where(e => e.FirstName == firstName && e.LastName == lastName).Single();

}

**Retrieve CRM Account by Id**

public Account RetrieveAccount(Guid accountId)

{

RefreshContext();

return context.AccountSet.Where(e => e.AccountId == accountId).Single();

}

**Retrieve CRM Active Accounts**

public IQueryable<Account> RetrieveActiveAccounts()

{

return

context.AccountSet.Where(

e =>

(e.CustomerTypeCode.Value == (int)ContactType.Customer ||

e.CustomerTypeCode.Value == (int)ContactType.Prospect) && e.StateCode == (int)StateCode.Active)

.OrderBy(e => e.Name);

}

**Retrieve CRM Accounts Last Modified**

public List<Account> RetrieveLastModifiedAccounts(int recordLastModifiedDays, bool activeOnly, ref bool isError, ref string errorDetails)

{

System.DateTime today = System.DateTime.Today;

if (recordLastModifiedDays > 0)

{

today = today.AddDays(recordLastModifiedDays \* -1);

}

if (activeOnly)

{

return context.AccountSet.Where(e => e.StateCode == (int)StateCode.Active && e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.Name).ToList();

}

return context.AccountSet.Where(e => e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.Name).ToList();

}

**Retrieve CRM Contacts Last Modified**

public List<Contact> RetrieveLastModifiedContacts(int recordLastModifiedDays, bool activeOnly, ref bool isError, ref string errorDetails)

{

System.DateTime today = System.DateTime.Today;

if (recordLastModifiedDays > 0)

{

today = today.AddDays(recordLastModifiedDays \* -1);

}

if (activeOnly)

{

return context.ContactSet.Where(e => e.StateCode == (int)StateCode.Active && e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.FullName).ToList();

}

return context.ContactSet.Where(e => e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.FullName).ToList();

}

**Retrieve CRM Opportunities Last Modified**

public List<Opportunity> RetrieveLastModifiedOpportunities(int recordLastModifiedDays, bool activeOnly, ref bool isError, ref string errorDetails)

{

System.DateTime today = System.DateTime.Today;

if (recordLastModifiedDays > 0)

{

today = today.AddDays(recordLastModifiedDays \* -1);

}

if (activeOnly)

{

return context.OpportunitySet.Where(e => e.StateCode == (int)OpportunityState.Open && e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.Name).ToList();

}

return context.OpportunitySet.Where(e => e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.Name).ToList();

}

**Retrieve CRM Opportunities by Status**

public List<Opportunity> RetrieveLastModifiedOpportunitiesByStatus(int status, int recordLastModifiedDays, ref bool isError, ref string errorDetails)

{

System.DateTime today = System.DateTime.Today;

if (recordLastModifiedDays > 0)

{

today = today.AddDays(recordLastModifiedDays \* -1);

}

return context.OpportunitySet.Where(e => e.StateCode == status && e.ModifiedOn.Value >= new DateTime(today.Year, today.Month, today.Day)).OrderBy(e => e.Name).ToList();

}

public List<Opportunity> RetrieveOpportunitiesByStatus(int status, ref bool isError, ref string errorDetails)

{

return context.OpportunitySet.Where(e => e.StateCode == status).OrderBy(e => e.Name).ToList();

}