# Microsoft Dynamics CRM SDK

RESTSilverlightContactEditor Readme

This folder contains sample code that demonstrates how to connect and utilize the Microsoft Dynamics CRM OData endpoint from a Silverlight 4.0 application using WCF Data Services. The sample provides a user interface with a search area to allow the user to search for Contacts by name. When Contacts are found they will populate an editable DataGrid control. The user can perform any CRUD operation, such as creates, retrieves, updates, and deletes.

Microsoft Dynamics CRM supports Silverlight XAP files as Web Resources. A XAP file is simply an archive folder containing a compiled Silverlight application and any files that the Silverlight application depends upon. This sample uses Web Resources to host the Silverlight application to make it available to Microsoft Dynamics CRM.

An important caveat of connecting to the OData endpoint is that it currently can only be done from within the Microsoft Dynamics CRM application. If you try to “Debug” the Silverlight application from directly within Visual Studio it will fail upon the first OData service call.

This sample introduces usage of the MVVM design pattern which stands for Model-View-ViewModel. It’s a pattern that helps to separate the different concerns when building a user interface, such as managing the behavior of the UI, managing the state of the UI, etc. Silverlight’s powerful DataBinding architecture lends itself well to this design pattern.

A ViewModel is a class created to contain all the data that the UI will display as public properties. When a property of the ViewModel changes, it should raise a PropertyChanged event which notifies any UI controls that are bound to the property to use the new value. The View, which is a XAML UserControl, will use the DataBinding architecture to bind to the properties of the corresponding ViewModel class. Typically, you will find a 1-to-1 relationship between a View and its corresponding ViewModel. In this case, the Model is simply the type of data coming from the server - Contact objects.

For more information about MVVM visit sites like <http://msdn.microsoft.com/en-us/magazine/dd458800.aspx> or do a search for “MVVM” at [www.bing.com](http://www.bing.com).

# Requirements

This sample comes ready to build without any compilation needed in Visual Studio 2010. However, if you choose to modify the Silverlight application you will need to recompile the XAP file by rebuilding the Visual Studio solution – see “[Rebuilding the XAP](#_Rebuilding_the_XAP).”

The *AdventureWorksCycleContext* class (used in MainPage.xaml.cs) is generated from referencing the Dynamics CRM OData endpoint and contains strongly typed classes for all out-of-the-box entities and attributes. If you want strong-type access to any custom entities or attributes from your environment, you’ll have to regenerate the service reference by following the steps outlined under “[Regenerate the OData Service Reference](#_Regenerate_the_OData).”

## View the Source Code

Before doing anything, you should view the source code to get a feel for the files included. Open the Visual Studio 2010 solution by double-clicking the “RESTSilverlightContactEditor.sln” file in the sample directory.

##### RESTSilverlightContactEditor project

This is the Silverlight application source code. The following folders and files were created, edited or moved. Other folders or files are part of the default Silverlight Application project created by Visual Studio.

* **Utilities folder** – created to hold the ServerUtility.cs file and other helper classes you may build on your own.
* **ServerUtility.cs** – The ServerUtility class includes methods to retrieve the Microsoft Dynamics CRM URL by invoking the Xrm.Page.context.getServerUrl javascript function. The Xrm javascript object is accessible when hosted inside an entity form or when the hosting HTML Web Resource includes a valid script reference to ClientGlobalContext.js.aspx. The utility is built not to throw an exception should it fail to find the Xrm object so the application can fail gracefully.
* **Views folder** – created to hold the XAML UserControls. It is common practice to separate your views into their own directory.
* **MainPage.xaml** – edited to contain a single StackPanel that will be used to display output to the user as the result of CRUD operations and moved to the Views folder.
* **MainPage.xaml.cs** – edited to contain the majority of the logic, reference the *AdventureWorksCycleContext* class, and moved to the Views folder.
* **ViewModels folder** – created to hold any ViewModel classes. In this sample there is only one named MainViewModel.cs.
* **MainViewModel.cs** – created as part of the MVVM design pattern. The MainViewModel contains all the data that the MainView.xaml will bind to.
* **Models folder** – created to hold any classes representing business objects. In this sample, we extend the Contact class that is generated from the Odata reference in order to add custom validation which gets used natively by the DataGrid in MainPage.xaml
* **Contact.cs** – created as a partial class to extend the OData Contact object. Custom validation attributes are used on the Contact object which then get picked up by the DataGrid’s Validation architecture to verify Last Name and valid Email address.
* **DataServiceContextExtensions.cs –** Added to provide a base ODataEntity class and extensions to the AdventureWorksCycleContext class so that only changed property values are saved.

##### RESTSilverlightContactEditor.Web project

When creating a Silverlight application in Visual Studio 2010 you are prompted with the choice to automatically create an accompanying Web Application to host the compiled Silverlight application (ClientBin\XAP). The following folders and files were created, edited or moved.

* **RESTSilverlightContactEditorTestPage.html** – edited to contain a valid script reference to ClientGlobalContext.js.aspx, deleted a script reference to Silverlight.js because it is already supplied by Microsoft Dynamics CRM, and edited the #silverlightControlHost CSS to a width: 100%.

## Create Web Resources and Preview

For this sample to work, you will create two Web Resources and upload the corresponding file from the sample. Each of these Web Resources needs to be named the following:

* **“/RESTSilverlightContactEditorTestPage.htm”** – An HTML Web resource can be used to view the Silverlight application outside of an entity form. The only purpose of this Web Resource is to provide the URL of the server because Silverlight can’t access the Xrm.Page.context.getServerUrl function when not viewed in an entity form.
* **“/ClientBin/RESTSilverlightContactEditor.xap”** – This Web Resource contains the XAP file which is a compiled Silverlight application. The name of this Web Resource simply reflects the relative output location of the XAP file within Visual Studio 2010.

**NOTE:** The names of the web resources start with “/” to simulate the same structure the files are organized in the Visual Studio solution. Using this naming convention you will not have to change the path of your XAP object reference in your html file before uploading them as Web Resources.

Once you have created the two properly named Web Resources and uploaded the corresponding files from the Visual Studio solution, be sure to Save and Publish both Web Resources.

Navigate to the **“/RESTSilverlightContactEditorTestPage.htm”** Web Resource and click the **Preview** button. You should see the Silverlight application displaying output within a few seconds.

## Rebuilding the XAP (Optional)

You only need to rebuild the XAP if you:

1. make a modification to the Silverlight application
2. regenerate the OData Service Reference.

Open the Visual Studio 2010 solution by double-clicking the **“RESTSilverlightContactEditor.sln”** file in the sample directory to see the files included.

Make any modifications you desire to the application. Perhaps you’d like to work with Contacts instead of Accounts or perhaps you’d like to add some visual elements.

In **Solution Explorer**, right-click the top node “RESTSilverlightContactEditor” and choose **Rebuild Solution**. A new XAP file will be created in the RESTSilverlightContactEditor.Web\ClientBin folder provided there are no build errors.

## Regenerate the OData Service Reference (Optional)

You only need to regenerate the OData Service Reference if you are making modifications to the Silverlight application and want strong-type access to custom entities or attributes. The existing reference was created from a non-customized Microsoft Dynamics CRM environment where the organization name was *AdventureWorksCycle*. It contains all of the out-of-box entities and attributes and the generated System.Data.Services.Client context class is named “*AdventureWorksCycleContext*.” The sample context class should still work within your application even if your organization name is different.

1. **Login** to your Microsoft Dynamics CRM application.
2. **Navigate** to **Settings** -> **Customization** -> **Developer Resources**.
3. You will see a list of services.
4. **Click** the **Download CSDL** link for the **Organization Data Service**.
5. Save the **OrganizationData.csdl** file.
6. In Visual Studio 2010, in the **CrmODataSilverlight** project, navigate to **Service References**.
7. **Delete** the **CrmODataService** service reference. (This will immediately cause errors to display in the Error List window)
8. Right-click **Service References** and choose **Add Service Reference**.
9. In Add Service Reference dialog, type in the **Address** of your **OrganizationData.csdl** file you saved in Step 5.
10. Click **Go** and wait for it to recognize the service.
11. Change the **Namespace** to **“CrmODataService.”**
12. Click **OK**.
13. Open **MainPage.xaml.cs** and identify lines 31 and 50 that reference the *AdventureWorksCycleContext* class.
14. Replace this class with the context class generated in your CrmODataService which should be named <*YourOrganizationName>Context.*
15. Open the **DataServiceContextExtensions.cs** and verify that the namespace is the same you set in step 11.
16. In the **DataServiceContextExtensions.cs** file, verify that the first partial class is named the same as the context class referenced by step 14.
17. In the Visual Studio **Solution Explorer**, click the icon with the tooltip **Show All Files**. Locate the **Reference.cs** file located in your **CrmODataService** service Reference. In the **Reference.cs** file, use **Find and Replace** to change all instances of “: global::System.ComponentModel.INotifyPropertyChanged” to “: ODataEntity, global::System.ComponentModel.INotifyPropertyChanged”.
18. Rebuild the solution and fix any build errors.

You have now regenerated your OData context class. You will be able to access any custom entities and attributes. Follow the steps in “Create Web Resources and Preview” to upload your new XAP file as a Web Resource.

## Quick Setup

This sample comes ready with a Microsoft Dynamics CRM Managed Solution that contains the Web Resources you would’ve created above. If you are having trouble getting your Web Resources to operate correctly or simply want to run the sample as quickly as possible, follow the following steps.

1. Open Microsoft Dynamics CRM.
2. Navigate to **Settings -> Solutions**.
3. Click **“Import.”**
4. Click **“Browse.”**
5. In the File Upload window, navigate to the root of the sample and choose the **“RESTSilverlightContactEditor\_1\_0\_0\_1\_managed.zip”** and click **Open.**
6. Click **Next**, **Next**, **OK** to finish.
7. Click **“Publish All Customizations.”**

#### Expected Results

Using the default solution, locate and open the HTML Web Resource named **‘sample\_/RESTSilverlightContactEditorTestPage.htm’** and click **Preview**.

## Troubleshooting

#### Build error(s)

If you get a build error when trying to build the sample, check to see that you are using the appropriate OgranizationContext (AdventureWorksCycleContext in the sample.) Update any using statements that need resolving.

#### Run-time errors

If you are experiencing errors when running the samples, check the following.

* You have Saved and Published all Customizations before viewing this sample on an Account form.
* If you are trying to “Preview” from a Web Resource, be sure to preview the HTML Web Resource, not the XAP resource. Only the HTML Web Resource will Preview correctly as it can extract the Server URL.

# Important Notes

The sample files are not intended to be used in a production environment. You should deploy this sample to a test environment and examine it for interaction or interference with other parts of the system.

Before you deploy this sample to a production environment, make sure that you consider the existing customizations you may have implemented in Microsoft CRM 2011.

# Copyright

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. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, ActiveX, BizTalk, Excel, Great Plains, Internet Explorer, JScript, Microsoft Dynamics, MSN, Outlook, PivotTable, PivotChart, Visual Basic, Visual Studio, Windows, Windows Live, Windows Server, and Windows Vista are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.