# Microsoft Dynamics CRM 2013 SDK

Microsoft.Crm.Sdk.Mobile Readme

[Microsoft Dynamics CRM 2013 SDK 1](#_Toc397767022)

[Overview 1](#_Toc397767023)

[Building the Library 2](#_Toc397767024)

[What is included in the library 2](#_Toc397767025)

[What isn’t included in the library 3](#_Toc397767026)

[How to write apps that use this library 3](#_Toc397767027)

[How to update existing code 4](#_Toc397767028)

[Known issues 4](#_Toc397767029)

[Change history 4](#_Toc397767030)

[Copyright 5](#_Toc397767031)

# Overview

The Microsoft.Crm.Sdk.Mobile source code sample is a partial re-implementation of the Microsoft Dynamics CRM SDK classes written as a Portable Class Library to facilitate development of store apps for Windows 8.1 desktop, tablets, and phones. The code makes use of the SOAP and OData protocols to issue web service method calls. An organization web service proxy and most of the message response/request classes in the CRM SDK have been implemented. When writing apps that use this code, you don’t have to link to the CRM SDK assemblies to access the organization web service.

This code can also be used when you develop apps using [Xamarin](http://xamarin.com/). However, you must comment out the **EnableProxyTypes** method in Microsoft.Xrm.Sdk.Samples.cs when developing iOS or Android apps because the method contains code that is specific to Windows Store. An alternative is to comment out the relevant code as shown here.

public async Task EnableProxyTypes()

{

    List<TypeInfo> typeList = new List<TypeInfo>();

    //// Obtain folder of executing application.

    //var folder = Package.Current.InstalledLocation;

    //foreach (var file in await folder.GetFilesAsync())

    //{

    //    // Not only .dll but .exe also contains types.

    //    if (file.FileType == ".dll" || file.FileType == ".exe")

    //    {

    //        var assemblyName = new AssemblyName(file.DisplayName);

    //        var assembly = Assembly.Load(assemblyName);

    //        foreach (TypeInfo type in assembly.DefinedTypes)

    //        {

    //            // Store only CRM entities.

    //            if (type.BaseType == typeof(Entity))

    //                typeList.Add(type);

    //        }

    //    }

    //}

    typeList.Add(typeof(ActivityParty).GetTypeInfo());

    typeList.Add(typeof(SavedQuery).GetTypeInfo());

    typeList.Add(typeof(Privilege).GetTypeInfo());

    typeList.Add(typeof(SystemUser).GetTypeInfo());

    typeList.Add(typeof(Annotation).GetTypeInfo());

    typeList.Add(typeof(TimeZoneDefinition).GetTypeInfo());

    types = typeList.ToArray();

}

# Building the Library

Your development computer must have an internet connection to build the library as a [NuGet](http://nuget.org/) package named Json.NET will be automatically downloaded during the build.

To build the Microsoft.Crm.Sdk.Mobile.dll assembly, follow these steps.

1. In Visual Studio 2013, with update 2 or later installed, load the Microsoft.Crm.Sdk.Mobile.sln file.
2. Press F6.

# What is included in the library

This library contains the following key components:

* **Web Service Proxy**
* This library provides most methods that the [OrganizationServiceProxy](http://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationserviceproxy.aspx) class in the CRM SDK does - Create, Update, Assign, Execute, and so on. The class supports early-bound development by providing an **EnableProxyTypes** method. SOAP implementations of these methods use the same name as are used in the CRM SDK while the OData-based methods have names that begin with “Rest” for example **RestCreate**.
* **SDK classes and enumerations**
* This library supports most common CRM SDK classes and enumerations. With the addition of early-bound type support, you can use Visual Studio IntelliSense when writing your apps.
* **Organization web service messages**
* This library includes request/response classes for over 200 messages. It covers both data operations, for example **Create**, and metadata operations, for example **RetrieveMetadataChanges**.

# What isn’t included in the library

Several functional areas that the library doesn’t cover are as follows:

**Authentication**

Since this sample code library targets multiple platforms, and each platform has its unique way to authenticate the user, the library doesn’t provide any authentication mechanism. You must obtain an OAuth access token and pass it to an **OrganizationDataWebServiceProxy** object to access the organization web service. It’s recommended that you use the [Microsoft Azure Active Directory Authentication Library](http://msdn.microsoft.com/en-us/library/jj573266.aspx) (ADAL) for identity authentication. This library is referenced in the CRM SDK documentation and sample code. There is an open source implementation of ADAL available for [iOS](https://github.com/AzureAD/azure-activedirectory-library-for-objc) and [Android](https://github.com/AzureAD/azure-activedirectory-library-for-android). There is a Windows 8.1 version available for the desktop, tablets, and phones.

For more information on ADAL see <http://www.cloudidentity.com/blog/2014/06/16/adal-for-windows-phone-8-1-deep-dive>.

**DiscoveryService, DeploymentService, OrganizationServiceContext**

This library doesn’t provide support for the discovery or deployment web services. It also doesn’t support the [OrganizationServiceContext](http://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.client.organizationservicecontext.aspx) class.

# How to write apps that use this library

You can build Microsoft.Crm.Sdk.Mobile.dll using the supplied Visual Studio 2013 solution and add a reference for it in your app’s project or you can add the library’s C# sample code files to your project.

In your app, add code to authenticate the user and obtain a security access token. The method used varies depending on platform as mentioned previously. Next, instantiate the **OrganizationDataWebServiceProxy** class.

OrganizationDataWebServiceProxy \_proxy = new OrganizationDataWebServiceProxy();

Set the access token and web server URL in the **OrganizationDataWebServiceProxy** object.

\_proxy.ServiceUrl = "<your CRM server URL goes here>";  
\_proxy.AccessToken = "<access token>";

Call **EnableProxyTypes** to use early-bound types and optionally assign a service time-out interval.

await \_proxy.EnableProxyTypes();

\_proxy.Timeout = 600;

Send a message request to the web service by using an async/await pattern.

WhoAmIResponse whoAmIResponse = (WhoAmIResponse)await \_proxy.Execute(new WhoAmIRequest());

// Retrieve User Info

SystemUser user = (SystemUser)await \_proxy.Retrieve(SystemUser.EntityLogicalName, whoAmIResponse.UserId, new ColumnSet(true));

# How to update existing code

If you want to use existing application code that was written for the CRM SDK with this library, make the following code changes.

1. Add the **async** keyword to methods.

2. Add **await** before the call to a proxy method.

3. When accessing the property of a web service response, do so as shown here.

*Original CRM SDK code*

\_proxy.RetrieveMultiple(new QueryExpression(“account”, new Columns(true)).Entities)

*Modified code*

(await \_proxy.RetrieveMultiple(new QueryExpression(“account”, new Columns(true))).Entities)

#### 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 Dynamics CRM 2013.
* This library was written by Kenichiro Nakamura at Microsoft.
* Source code for the CrmSvcMobileUtil program is provided in a separate project. Use that program to generate any custom or customized entity types in your organization for inclusion in your application.

# Known issues

There are no known issues at this time.

# Change history

1.0: Initial version

1.1: Added support for the new CRM 2015 messages and supporting classes

* All new 2015 message classes added, except for CalculatePrice and RetrieveCurrentOrganization
* New classes – AttributeMapping, AttributeMappingCollection
* LoadFromXml<T> method updated to handle a null string value
* IsLogical and SourceType fields added to the AttributeMetadata class
* FormulaDefinition and SourceTypeMask fields added to some classes derived from the AttributeMetadata class
* IsBaseCurrency field added to the MoneyAttributeMetadata class
* Many classes updated to use the revised LoadFromXml<T> method to handle null strings returned from the web service
* A sample file named CRMHelpers.cs is included in this package, but is not built as part of the library. This code demonstrates universal app authentication.

# 

# 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.

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.

© 2015 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, IntelliSense, Microsoft Dynamics, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.