# Microsoft Dynamics CRM SDK

Xrm.Page Script Library Template Readme

The Xrm.Page Script Library Template is a Microsoft Visual Studio 2010 project that provides JScript IntelliSense and capabilities to test functions in JScript libraries that use Xrm.Page objects and methods.

# Summary

The Xrm.Page Script Library Template makes it easier to write and test JScript library functions that work with Xrm.Page objects in Microsoft Dynamics CRM 2011 forms.

This template provides the following advantages:

1. JScript IntelliSense completion and method documentation.
2. You can perform initial testing without creating a Script Web resource.
3. Debugging is simplified.

**Note**: Although the Xrm.Page Script Library Template provides capabilities for initial testing, it should not replace actual testing of JScript libraries in the application.

## Components

The Xrm.Page Script Library Template consists of two components:

#### Xrm.Page JScript Library Project

With this Visual Studio 2010 extension installed on your development environment you can create a new Visual Studio project specifically for the development of JScript libraries that interact with Xrm.Page objects and methods.

#### Xrm.Page Script Development Form Snapshot

With this managed solution installed in your development Microsoft Dynamics CRM organization you can capture a snapshot of all the data available in a form through the Xrm.Page objects and methods. You will copy the output from this solution into your Xrm.Page JScript Library project to create an in-memory representation of the Xrm.Page object in the form when you captured your form snapshot.

# Installation

## Prerequisites

1. You must have privileges to install managed solutions for the Microsoft Dynamics CRM 2011 development organization. The System Customizer or System Administrator security roles should provide these privileges.
2. You must have Visual Studio 2010 or Visual Studio 2012. All versions will work but for best results the Visual Studio 2010 Ultimate, Premium, or Professional versions are recommended.
3. For best results with Visual Studio 2010, install the [JScript Editor Extensions](http://visualstudiogallery.msdn.microsoft.com/872d27ee-38c7-4a97-98dc-0d8a431cc2ed/) for Visual Studio 2010. These free extensions provide enhancements for editing JScript in Visual Studio. The Xrm.Page Script Library Template is designed to work with these extensions for optimal presentation of method documentation in Visual Studio. These extensions are not available for Visual Studio 2010 Express editions.

## Installation Procedure

#### To install the Xrm.Page JScript Library Project

Although a JScript Library project does not contain any managed code, Visual Studio separates all project templates by .NET languages. Therefore, there are two versions of this Visual Studio extension available. Install the one that represents the .NET language available in your installation of Visual Studio 2010 or Visual Studio 2012.

The installation files are available in the download files for the Microsoft Dynamics CRM 2011 SDK at SDK\Templates\Xrm.PageScriptProjectTemplate.

* For Visual Studio 2010:
  + C# use XrmPageScriptDevelopmentProjectCS\_VS2010.vsix.
  + Visual Basic .Net use XrmPageScriptDevelopmentProjectVB\_VS2010.vsix.
* For Visual Studio 2012:
  + C# use XrmPageScriptDevelopmentProjectCS\_VS2012.vsix.
  + Visual Basic .Net use XrmPageScriptDevelopmentProjectVB\_VS2012.vsix.

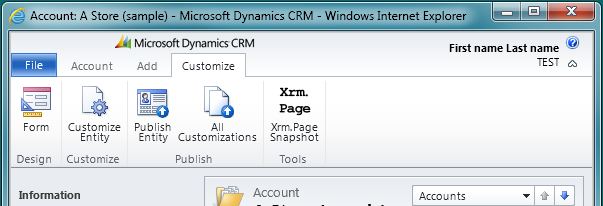
1. Double-click the .vsix file to install it.
2. The Visual Studio Extension Installer dialog box prompts you to install the extension.
3. Close all open instances of Visual Studio for the project to be available to use.

#### To install the Xrm.Page Script Development Form Snapshot

1. In your development Microsoft Dynamics CRM organization, navigate to **Settings** > **Customization** and select the **Solutions** area.
2. On the toolbar select **Import**.
3. In the **Import Solution** dialog box click the **Browse** button and select the XrmPageScriptDevelopmentFormSnapshot\_1\_0\_0\_1\_managed.zip file located in the SDK download files at SDK\Templates\Xrm.PageScriptProjectTemplate
4. Continue selecting **Next** in the **Import Solution** dialog box until the managed solution is installed.
5. Close Internet Explorer and open your development Microsoft Dynamics CRM organization.

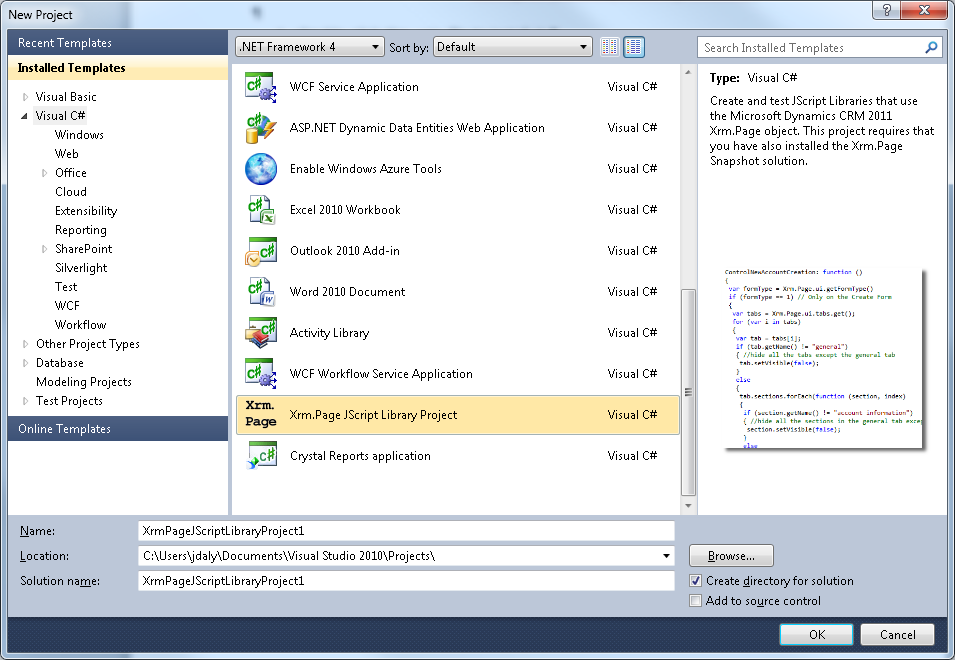
# Using the Template

After you have installed both components of the Xrm.Page Script Library Template you are ready to begin using it.

1. Open Microsoft Dynamics CRM and navigate to an entity record for the form you want to develop your script to work with.
2. On the **Customize** tab click the **Xrm.Page Snapshot** button shown below:
3. When the Event Context Dialog opens you have the option to choose an event.

**Note**: You only need to choose an event if you are creating a JScript function that depends on the Microsoft CRM form event handler to pass event context as the first parameter to the function. If your function does not require this, leave the default option of **None** and click **Get Data**. For more information about writing functions for specific event context, see [Write Functions That Use the Event Context](#_Write_Functions_That).

1. Click **Get Data** and then **Copy to Clipboard**. You will paste this data in step 9.
2. In Visual Studio, select **File** > **New** > **Project**.
3. Select the .NET language option you used to install the Xrm.Page Script Library Template and scroll down until you see the **Xrm.Page JScript Library Project** template as shown below:



1. Click **OK** to open your Visual Studio project.
2. When the project opens, in Solution Explorer expand the Scripts folder and open the PageData.js file.
3. Select all the contents of the PageData.js file and paste the text you copied in step 4.
4. Save the PageData.js file.

You are now ready to begin writing a JScript Library.

# Write Your JScript Library

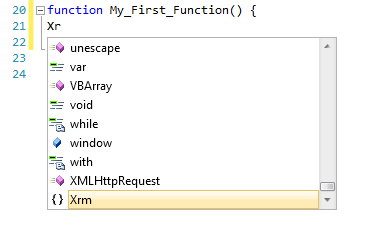
Within the Xrm.Page project you will find a sample JScript library file: My.Library.js

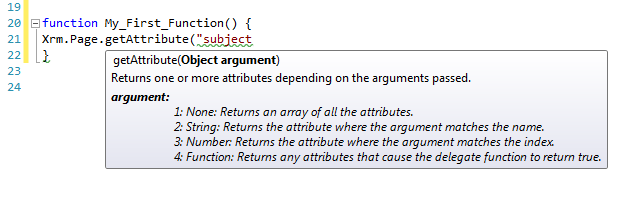
This library contains simple example functions to illustrate a strategy to use to help ensure that the functions in your JScript library have unique names. This will help prevent another JScript library from overwriting your function.

You can edit these example functions or remove them and create your own functions. You can create new JScript library files.

**Important**: If you create new JScript library files you must drag the XrmPageTemplate.js file to the top of your JScript library to create the reference that enables Visual Studio IntelliSense. The following reference must appear at the top of the file:

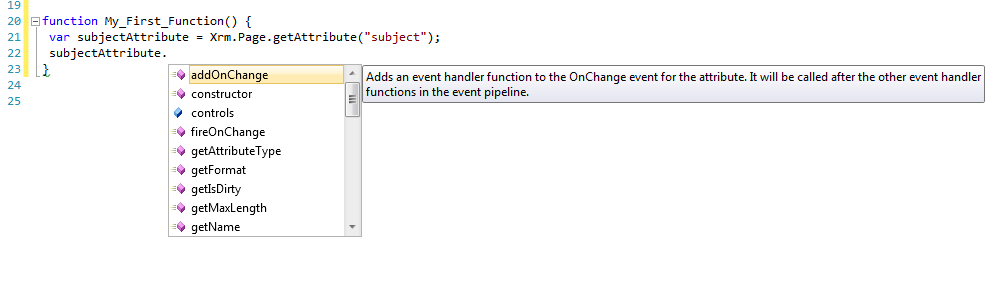
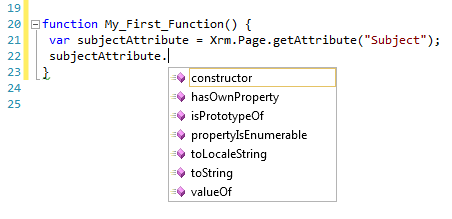
**/// <reference path="XrmPageTemplate.js" />**

With this reference in place you can simply start typing ‘Xrm” and you will see the completion and method documentation that JScript IntelliSense provides as shown in the following illustration.

When you begin to set the parameters of a function you will see the IntelliSense documentation for that function as shown here.

The Xrm.Page Script Library Template does not provide IntelliSense support for the string values that identify Xrm.Page elements such as attributes or controls. You still need to know the names of the items you reference. However, the template will give you a clue when you have typed the name of an attribute incorrectly.

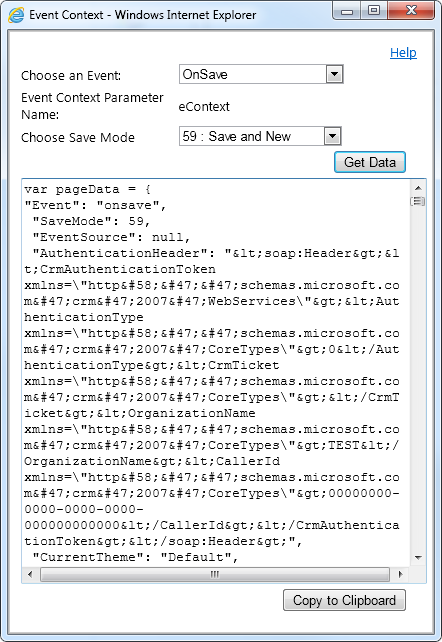
If you have used the correct name you should expect to see IntelliSense completion for any available Xrm.Page children of the object. For example, when referencing the subject attribute of a Lead entity correctly you should see the following.

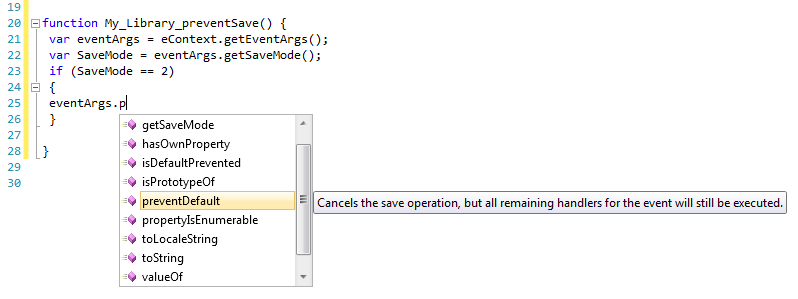
If you use the incorrect name, or the casing is not correct, you will only see the standard JScript object properties with no IntelliSense method documentation.

**Note**: While this IntelliSense clue can be helpful it is not 100% reliable. See [JScript IntelliSense Limitations](#_JScript_IntelliSense_Limitations) for more information.

# Write Functions That Use the Event Context

It is not possible to set the value of a Jscript function parameter at design time so that IntelliSense can be generated for it. Setting the parameter will initialize any variable already defined. To work around this at design time the Xrm.Page Template generates a global eContext object that you can use as long as you do not set a parameter for your function using the same name.

For example, for your function to have the capability to cancel a save event you must pass the event context as the first parameter to your function. When you use the Xrm.Page Form Snapshot Event Context dialog you would select OnSave as the event and a Save Mode if that is relevant to your script.

When you paste the definition into the PageData.js file it will generate a global eContext object that you can use to have IntelliSense while writing your script as shown below.

**Important:** Before you begin to test your script you must add ‘eContext’ as the first parameter to your function in order for it to work property.

# Testing your Functions

The XrmPageTemplate.js library uses the data from PageData.js to create an Xrm.Page object that is almost fully functional. Because of this you can use the TestPage.htm to perform preliminary unit testing on the functions in your JScript library to verify that they work before you upload the files as Script Web Resources and perform final testing in Microsoft Dynamics CRM.

The TestPage.htm is a test harness and does not provide the user interface for the form. With the TestPage.htm you can execute your code and use the **writeMessage** function to display messages to confirm the behavior you expect to see in the application.

For example, if you use the **setVisible** function to hide a form section, you can include code in TestPage.htm using the **getVisible** function to display the current visibility of the section.

For test code to work, the TestPage.htm must include references to the PageData.js, the XrmPageTemplate.js and the library you are testing.

**Note:** The code in the XrmPageTemplate.js library that replicates the Xrm.Page object is reverse engineered based on documented supported methods. It does not share any code with the actual Microsoft Dynamics CRM code that generates the Xrm.Page object in the form. The Xrm.Page objects contain private properties that each begin with the letter ‘z’. You should never reference any of these properties in your code.

## Limitations for Testing

The Xrm.Page Script Library Template is intended to be used for initial tests early in the development process. It is not a substitute for testing within the application. The Xrm.Page Script Library Template is reverse engineered to support a design time experience and does not contain any code from the application. It provides a model of the Xrm.Page object that only contains the supported objects and methods.

Except for the following limitations, the Xrm.Page object provided is fully functional according to the documented methods. Because the TestPage.htm does not represent an actual Microsoft Dynamics CRM form in the application the following methods have no effect during testing with TestPage.htm.

* Xrm.Page.data.entity.save will not save the record and the use of the “saveandclose” and “saveandnew” parameters have no effect.
* Xrm.Page.ui.formSelector item.navigate cannot navigate to a new form.
* Xrm.Page.ui.refreshRibbon cannot refresh the Ribbon.
* The subgrid control refresh button cannot refresh the subgrid.
* It is not valid to do performance testing using the Xrm.Page Script Library Template Xrm.Page object.
* The Xrm.Page Script Library Template cannot be used to interact with the contents of an IFrame or methods exposed by a Silverlight Web resource because these are not part of the Xrm.Page object.
* The IFrame and Web Resource getObject method is not implemented.
* The datetime.format and datetime.localeFormat return values based on the operating system locale, not the crmuser locale preferences.
* You cannot test OData or SOAP XMLHttpRequests using Microsoft Dynamics CRM Online with the Xrm.Page Script Library Template.
  + If you add your local development server (or localhost) to a trusted group within your Internet Explorer security settings and then configure that trusted group to allow cross-domain requests you may be able to test XMLHttpRequests with Microsoft Dynamics CRM On-Premises or other Web sites or Web services that support these requests.

# Known Issues

## JScript IntelliSense Limitations

**Note**: JavaScript IntelliSense in Visual Studio 2012 is much better than Visual Studio 2010. Most of the following applies only to Visual Studio 2010.

JScript IntelliSense in Visual Studio depends on the ability of Visual Studio to infer types across assignments between JScript variables that are inherently type-less. It will try to instantiate each of the variables that it finds and generate IntelliSense using that information. There are limits to how far Visual Studio can go in maintaining this. You will find that in certain circumstances IntelliSense will simply stop working. The fact that IntelliSense stops working does not necessarily mean that your code is incorrect.

Two common cases where IntelliSense will stop working are:

* Using the If statement to test a Boolean value.
* When you write code that uses several methods to assign a variable.

Despite these limitations there are still ways to use IntelliSense to confirm your syntax while you code.

For example, if you are testing whether a record has changes you will use the Xrm.Page.data.entity.getIsDirty method. If you try to write your code in the following way you will not have IntelliSense support:

if (Xrm.Page.data.entity.getIsDirty())

{ }

However, you will have IntelliSense support if you write your code this way:

var isDirty = Xrm.Page.data.entity.getIsDirty()

if (isDirty)

{ }

The following variable assignments represent the types of assignments where you can expect JScript IntelliSense to stop working:

var a = Xrm.Page.data.entity.attributes.get(0).controls.get(0).getAttribute();

var b = Xrm.Page.ui.controls.get(0).getAttribute();

var c = Xrm.Page.ui.tabs.get(0).sections.get(0);

For more information see [JScript IntelliSense Overview](http://msdn.microsoft.com/en-us/library/bb385682.aspx)

## getObject not implemented

IFrames and Web resources have a [getObject](http://msdn.microsoft.com/en-us/library/gg334266.aspx#BKMK_getObject) method that is not included at this time.

# Troubleshooting

## Update JScript IntelliSense

If you are not seeing IntelliSense when you expect to, try using Ctrl+Shift+J to tell Visual Studio to recalculate the in-memory representation of the JScript variables.

# 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.

© 2012 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.