Windows Media Center Application

Step by Step

Microsoft Corporation

Applies To:

* Microsoft® Windows Media® Center
* Microsoft® Windows Media® Center Software Development Kit 6.0

# Summary

This document takes you step by step through creating a simple Windows Media Center Application with Media Center Markup Language (MCML) for the visuals leveraging managed code and the Windows Media Center object model.

This article assumes a clean install of Windows and software outlined in the Preparation section.

**The code and markup provided here may have formatting applied which may interrupt direct copy + paste operations.** It is assumed the developer understands the basics of hand coding both C# and XML. At the end of this walkthrough the developer will have an application nearly identical to the result of the Windows Media Center Application Project Template for Visual C# 2008 Express Edition and Visual Studio 2008. Comparing the Step By Step project with the template can be an aid in debugging.

# Legal Notice

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, Windows, Windows NT, Windows Server, Windows Vista, Active Directory, ActiveSync, ActiveX, Direct3D, DirectDraw, DirectInput, DirectMusic, DirectPlay, DirectShow, DirectSound, DirectX, Expression, FrontPage, HighMAT, Internet Explorer, JScript, Microsoft Press, MSN, Outlook, PlaysForSure logo, PowerPoint, SideShow, Visual Basic, Visual C++, Visual InterDev, Visual J++, Visual Studio, WebTV, Windows Media, Win32, and Win32s are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

All other trademarks are property of their respective owners.

Some of the links in this document might let you leave Microsoft's site. The linked sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement by Microsoft of the site.

Table of Contents

[Summary 1](#_Toc228944145)

[Legal Notice 2](#_Toc228944146)

[Preparation 4](#_Toc228944147)

[Project Setup 5](#_Toc228944148)

[Create and Save the Project 5](#_Toc228944149)

[Set the Properties 5](#_Toc228944150)

[Add References 6](#_Toc228944151)

[Add the Code 6](#_Toc228944152)

[Create the Code Folder 6](#_Toc228944153)

[Add Launch.cs 6](#_Toc228944154)

[Add Application.cs 7](#_Toc228944155)

[Add the Markup 8](#_Toc228944156)

[Create the Markup Folder 9](#_Toc228944157)

[Add Styles.mcml 9](#_Toc228944158)

[Add Button.mcml 9](#_Toc228944159)

[Add Controls.mcml 10](#_Toc228944160)

[Add Menu.mcml 10](#_Toc228944161)

[Add RepeatItem.mcml 11](#_Toc228944162)

[Add Test.mcml 11](#_Toc228944163)

[Add the Resources 12](#_Toc228944164)

[Add Resources.resx 12](#_Toc228944165)

[Add Strings 12](#_Toc228944166)

[Add Files 12](#_Toc228944167)

[Build the Solution 13](#_Toc228944168)

[Create the Registration XML 13](#_Toc228944169)

[Determine the Public Key Token 13](#_Toc228944170)

[Add The Setup Folder 13](#_Toc228944171)

[Add Registration.xml 13](#_Toc228944172)

[Add The Images Folder 14](#_Toc228944173)

[Add The Program Image 14](#_Toc228944174)

[Enable UI Testing and MCML Verification 14](#_Toc228944175)

[Verify the MCML 16](#_Toc228944176)

[Test the UI 16](#_Toc228944177)

[Create the Development Install Script 17](#_Toc228944178)

[Development Install 18](#_Toc228944179)

[Launch the Application 19](#_Toc228944180)

[Development Uninstall 19](#_Toc228944181)

[Create the Setup Program Using WiX 20](#_Toc228944182)

[Add License.rtf 20](#_Toc228944183)

[Add License.txt 20](#_Toc228944184)

[Add LicenseAccessible.txt 21](#_Toc228944185)

[Add Build.cmd 21](#_Toc228944186)

[Add Setup.wxs 22](#_Toc228944187)

[Add Setup-en-us.wxl 25](#_Toc228944188)

[Add WiX Post-build Event 26](#_Toc228944189)

[Build the Solution 26](#_Toc228944190)

[Install the Application 26](#_Toc228944191)

[Test the Application 26](#_Toc228944192)

[Congratulations! 27](#_Toc228944193)

[Need Help? 27](#_Toc228944194)

# Preparation

1. Download and Install Microsoft Visual C# Express Edition 2008 from <http://msdn.microsoft.com/vstudio/express/downloads/>.
2. Download and Install the Windows Media Center Software Development Kit from <http://www.microsoft.com/downloads/details.aspx?familyid=a43ea0b7-b85f-4612-aa08-3bf128c5873e&displaylang=en>.
3. Download and Install GUIDGEN.EXE from <http://www.microsoft.com/downloads/details.aspx?familyid=94551f58-484f-4a8c-bb39-adb270833afc&displaylang=en>.
4. Download and install the latest build of WiX v3.0 (Wix3.msi) from <http://wix.sourceforge.net/downloadv3.html>.

# Project Setup

Setting up the Microsoft Visual C# Express Edition 2008 project prepares all of the infrastructure needed for our managed code assembly, including creation of a strong name and references to the Windows Media Center assemblies necessary to call the managed code object model.

## Create and Save the Project

1. Launch Microsoft Visual C# 2008 Express Edition
2. Select File > New Project from the menu.
3. In the New Project dialog:
   1. Select the Empty Project template.
   2. Type MyProject in the Name field.
   3. Click the OK button.
4. Select File > Save All from the menu.
5. In the Save Project dialog:
   1. Type MyProject in the Name field.
   2. Select a location for the project.
   3. Clear the check box next to Create directory for solution.
   4. Click the Save button.

## Set the Properties

1. Select View > Solution Explorer from the menu.
2. In Solution Explorer select the MyProject project.
3. Select Project > MyProject Properties from the menu.
4. In the Signing tab of the MyProject Properties:
   1. Check the box next to Sign the assembly.
   2. Select <New> from the Choose a strong name key file drop down list.
   3. In the Create Strong Name Key dialog:
      1. Type MyProject in the Key file name field.
      2. Clear the check box next to Protect my key file with a password.
      3. Click the OK button.
5. In the Application tab of the MyProject Properties:
   1. Select Class Library from the Output Type drop down list.
   2. Click the Assembly Information button.
   3. In the Assembly Information dialog:
      1. Type MyProject in the Title field.
      2. Type 1.0.0.0 in the Assembly Version field.
      3. Type 1.0.0.0 in the File Version field.
      4. Generate a Registry Format GUID using Guidgen.exe and enter it in the GUID field without curly braces. **Note:** Each GUID referenced in this document should be absolutely unique and not shared or used elsewhere. To aid the reader, this is represented in the whitepaper as GUID-X where X is an incremented number. This step is GUID-1.
      5. Clear the check box next to Make assembly COM-Visible
      6. Click the OK button.
   4. Select .NET Framework 2.0 from the Target Framework drop down list.
   5. Select the Yes button in the Target Framework Change dialog box. The project will save, close and reopen.

## Add References

1. In Solution Explorer select References.
2. Select Project > Add Reference from the menu.
3. In the Add Reference dialog:
   1. Select the Browse tab.
   2. Navigate to c:\windows\ehome\.
   3. Hold down the control key and select **Microsoft.MediaCenter.dll** and **Microsoft.MediaCenter.UI.dll**.
   4. Click the OK button.

# Add the Code

Code in a Windows Media Center application provides the logic and data for the experience. In the context of this whitepaper it provides several pieces of functionality:

1. Implements the required Initialize and Uninitialize via IAddInModule and the Launch method via IAddInEntryPoint. The launch method provides the opportunity to hook into an instance of the AddInHost which provides access to the managed code object model.
2. Inherits from ModelItem to create a custom class.
3. Creates a HistoryOrientedPageSession which allows for an application page stack and navigating to pages defined using Media Center Markup Language (MCML).
4. Defines a local application method which calls the managed code API MediaCenterEnvironment.Dialog Method to display a Media Center styled dialog box.
5. Provides a simple string array providing the data for the application.

## Create the Code Folder

1. In Solution Explorer select the MyProject project.
2. Select Project > New Folder from the menu.
3. In Solution Explorer rename NewFolder1 to Code.

## Add Launch.cs

1. In Solution Explorer select the Code folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the Code File template.
   2. Type Launch.cs in the Name field.
   3. Click the Add button.
   4. Type the following in the Launch.cs editing window:

using System.Collections.Generic;

using Microsoft.MediaCenter.Hosting;

namespace MyProject

{

public class MyAddIn : IAddInModule, IAddInEntryPoint

{

private static HistoryOrientedPageSession s\_session;

public void Initialize(Dictionary<string, object> appInfo, Dictionary<string, object> entryPointInfo)

{

}

public void Uninitialize()

{

}

public void Launch(AddInHost host)

{

if (host != null && host.ApplicationContext != null)

{

host.ApplicationContext.SingleInstance = true;

}

s\_session = new HistoryOrientedPageSession();

Application app = new Application(s\_session, host);

app.GoToMenu();

}

}

}

## Add Application.cs

1. In Solution Explorer select the Code folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the Code File template.
   2. Type Application.cs in the Name field.
   3. Click the Add button.
4. Type the following in the Application.cs editing window:

using System.Collections.Generic;

using System.Diagnostics;

using Microsoft.MediaCenter;

using Microsoft.MediaCenter.Hosting;

using Microsoft.MediaCenter.UI;

namespace MyProject

{

public class Application : ModelItem

{

private AddInHost host;

private HistoryOrientedPageSession session;

public Application() : this(null, null)

{

}

public Application(HistoryOrientedPageSession session, AddInHost host)

{

this.session = session;

this.host = host;

}

public MediaCenterEnvironment MediaCenterEnvironment

{

get

{

if (host == null) return null;

return host.MediaCenterEnvironment;

}

}

public void GoToMenu()

{

Dictionary<string, object> properties = new Dictionary<string, object>();

properties["Application"] = this;

if (session != null)

{

session.GoToPage("resx://MyProject/MyProject.Resources/Menu", properties);

}

else

{

Debug.WriteLine("GoToMenu");

}

}

public void DialogTest(string strClickedText)

{

int timeout = 5;

bool modal = true;

string caption = Resources.DialogCaption;

if (session != null)

{

MediaCenterEnvironment.Dialog(strClickedText,

caption,

new object[] { DialogButtons.Ok },

timeout,

modal,

null,

delegate(DialogResult dialogResult) { });

}

else

{

Debug.WriteLine("DialogTest");

}

}

public string[] MyData

{

get

{

return new string[4] { "Alpha", "Bravo", "Charlie", "Delta" };

}

}

}

}

# Add the Markup

The Media Center Markup Language is used to define all of the visuals for our application. This whitepaper puts forth a very, very simple user interface: flat color fills for buttons. These can be greatly enhanced using the powerful rendering capabilities provided by MCML. The files added in this section are as follows:

* Styles.mcml provides a location for application wide shared styling resources.
* Button.mcml defines the main visual for our application (a button) which can be altered at any time without having to resort to changing our managed code. This also introduces the concept of Properties.
* Controls.mcml aggregates a family of controls (for example: a button, spinner, checkbox) to be available from a single namespace. In the context of this whitepaper there is only a single control so this resource provides ‘growing room’ when additional controls are added later.
* Menu.mcml defines the single page in the application and serves as the container for the individual buttons. It also shows how to use a couple of layout types at both a parent and child level. Data binding in the MCML also occurs in this resource.
* RepeatItem.mcml takes the data provided by the managed code via Menu.mcml and wires it up to our button control defined in Button.mcml. It also provides the path by which Button.mcml calls our method for displaying the Windows Media Center dialog box.
* Test.mcml is a container used for testing the buttons (or any other defined UI) outside of the confines of our managed code assembly in the Media Center Markup Language Preview Tool. This allows for testing of the UI without needing to actually install the assembly and launching Windows Media Center.

## Create the Markup Folder

1. In Solution Explorer select the MyProject project.
2. Select Project > New Folder from the menu.
3. In Solution Explorer rename NewFolder1 to Markup.

## Add Styles.mcml

1. In Solution Explorer select the Markup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the MCML File template.
   2. Type Styles.mcml in the name field.
   3. Click the Add button.
4. Type the following in the Styles.mcml editing window:

<Mcml xmlns="http://schemas.microsoft.com/2008/mcml">

<Color Name="FocusColor"

Color="Yellow"/>

<Color Name="NonFocusColor"

Color="Firebrick"/>

<Color Name="SelectedColor"

Color="Green"/>

</Mcml>

## Add Button.mcml

1. In Solution Explorer select the Markup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the MCML File template.
   2. Type Button.mcml in the name field.
   3. Click the Add button.
4. Type the following in the Button.mcml editing window:

<Mcml xmlns="http://schemas.microsoft.com/2008/mcml"

xmlns:s="resx://MyProject/MyProject.Resources/Styles">

<UI Name="Button">

<Properties>

<ICommand Name="Model"

ICommand="$Required"/>

<Color Name="MyNonFocusColor"

Color="color://s:NonFocusColor"/>

<Color Name="MyFocusColor"

Color="color://s:FocusColor"/>

<Color Name="MySelectedColor"

Color="color://s:SelectedColor"/>

</Properties>

<Locals>

<ClickHandler Name="Clicker"/>

</Locals>

<Rules>

<Binding Source="[Model.Description]"

Target="[Label.Content]"/>

<Binding Source="[Model]"

Target="[Clicker.Command]"/>

<Condition Source="[Clicker.Clicking]"

SourceValue="true">

<Actions>

<Set Target="[Background.Content]"

Value="[MySelectedColor]"/>

</Actions>

</Condition>

<Condition Source="[Input.KeyFocus]"

SourceValue="true">

<Actions>

<Set Target="[Background.Content]"

Value="[MyFocusColor]"/>

</Actions>

</Condition>

</Rules>

<Content>

<ColorFill Name="Background"

Content="[MyNonFocusColor]"

Padding="5,5,5,5"

MinimumSize="300,50"

MaximumSize="300,50">

<Children>

<Text Name="Label"

Color="Black"

Font="Calibri,24"/>

</Children>

</ColorFill>

</Content>

</UI>

</Mcml>

## Add Controls.mcml

1. In Solution Explorer select the Markup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the MCML File template.
   2. Type Controls.mcml in the name field.
   3. Click the Add button.
4. Type the following in the Controls.mcml editing window:

<Mcml xmlns="http://schemas.microsoft.com/2008/mcml">

<Aggregate Source="resx://MyProject/MyProject.Resources/Button"/>

</Mcml>

## Add Menu.mcml

1. In Solution Explorer select the Markup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the MCML File template.
   2. Type Menu.mcml in the name field.
   3. Click the Add button.
4. Type the following in the Menu.mcml editing window:

<Mcml xmlns="http://schemas.microsoft.com/2008/mcml"

xmlns:cor="assembly://MsCorLib/System"

xmlns:a="assembly://MyProject/MyProject"

xmlns:s="resx://MyProject/MyProject.Resources/Styles"

xmlns:r="resx://MyProject/MyProject.Resources/RepeatItem">

<UI Name="Default">

<Properties>

<a:Application Name="Application"

Application="$Required"/>

</Properties>

<Content>

<Panel Layout="Center">

<Children>

<Repeater Source="[Application.MyData]" >

<Layout>

<FlowLayout Orientation="Vertical"

Spacing="4,0"/>

</Layout>

<Content>

<r:RepeatItem Text="[RepeatedItem!cor:String]"

Application="[Application]"/>

</Content>

</Repeater>

</Children>

</Panel>

</Content>

</UI>

</Mcml>

## Add RepeatItem.mcml

1. In Solution Explorer select the Markup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the MCML File template.
   2. Type RepeatItem.mcml in the name field.
   3. Click the Add button.
4. Type the following in the RepeatItem.mcml editing window:

<Mcml xmlns="http://schemas.microsoft.com/2008/mcml"

xmlns:cor="assembly://MsCorLib/System"

xmlns:a="assembly://MyProject/MyProject"

xmlns:c="resx://MyProject/MyProject.Resources/Controls">

<UI Name="RepeatItem">

<Properties>

<a:Application Name="Application"

Application="$Required"/>

<cor:String Name="Text"

String="$Required"/>

</Properties>

<Locals>

<Command Name="MyCommand"/>

</Locals>

<Rules>

<Binding Target="[MyCommand.Description]"

Source="[Text]"/>

<Changed Source="[MyCommand.Invoked]">

<Actions>

<Invoke Target="[Application.DialogTest]"

strClickedText="[Text]"/>

</Actions>

</Changed>

</Rules>

<Content>

<c:Button Name="Button"

Model="[MyCommand]" />

</Content>

</UI>

</Mcml>

## Add Test.mcml

1. In Solution Explorer select the Markup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the MCML File template.
   2. Type Test.mcml in the name field.
   3. Click the Add button.
4. Type the following in the Test.mcml editing window:

<Mcml xmlns="http://schemas.microsoft.com/2008/mcml"

xmlns:c="resx://MyProject/MyProject.Resources/Controls">

<UI Name="Test">

<Locals>

<Command Name="Command"

Description="Button Test"/>

</Locals>

<Content>

<Panel>

<Layout>

<FlowLayout Orientation="Vertical"

Spacing="4,0"/>

</Layout>

<Children>

<c:Button Model="[Command]"/>

<c:Button Model="[Command]"/>

<c:Button Model="[Command]"/>

<c:Button Model="[Command]"/>

</Children>

</Panel>

</Content>

</UI>

</Mcml>

# Add the Resources

Resources are a standard part of any managed code assembly designed for worldwide use. In addition to the string used for the dialog box the MCML files are embedded in the assembly for ease of deployment.

## Add Resources.resx

1. In Solution Explorer select the MyProject project.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the Resources File template.
   2. Type Resources.resx in the Name field.
   3. Click the Add button.

## Add Strings

1. Press Control+1 on the keyboard.
2. Add the following String to Resources.resx:

Name: DialogCaption

Value: Button Selected

## Add Files

1. Press Control+5 on the keyboard.
2. Add the following Files to Resources.resx with drag and drop from Solution Explorer (note the file extensions do not appear in the Resources.resx user interface):

Button.mcml

Controls.mcml

Menu.mcml

RepeatItem.mcml

Styles.mcml

# Build the Solution

1. Select View > Output from the menu.
2. Select Build > Build Solution from the menu.
3. Observe something similar to the following in the Output window:

------ Build started: Project: MyProject, Configuration: Release Any CPU ------

C:\Windows\Microsoft.NET\Framework\v2.0.50727\Csc.exe /noconfig /nowarn:1701,1702 /errorreport:prompt /warn:4 /define:TRACE /reference:..\..\..\..\..\..\Windows\ehome\Microsoft.MediaCenter.dll /reference:..\..\..\..\..\..\Windows\ehome\Microsoft.MediaCenter.UI.dll /reference:C:\Windows\Microsoft.NET\Framework\v2.0.50727\System.dll /debug:pdbonly /keyfile:MyProject.snk /optimize+ /out:obj\Release\MyProject.dll /resource:obj\Release\MyProject.Resources.resources /target:library Code\Application.cs Code\Launch.cs Properties\AssemblyInfo.cs Resources.Designer.cs

Compile complete -- 0 errors, 0 warnings

MyProject -> C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\bin\Release\MyProject.dll

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

# Create the Registration XML

The registration XML allows us to define the application and how it appears to end users in the Windows Media Center user interface.

## Determine the Public Key Token

1. Select Tools -> External Tools from the menu.
2. In the External Tools dialog:
   1. If there are already existing tools click Add to create a new one. Otherwise, if this is the first external tool [New Tool 1] is already selected
   2. Enter Get Public Key in the Title text box.
   3. Enter the following in the Command text box: C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\sn.exe
   4. Type the following in the Arguments text box: -Tp "$(TargetPath)"
   5. Uncheck all options except Use Output window.
   6. Click the OK button.
3. Select Tools > Get Public Key from the menu.
4. Note the Public Key Token from the output window.

## Add The Setup Folder

1. In Solution Explorer select the MyProject project.
2. Select Project > New Folder from the menu.
3. In Solution Explorer rename NewFolder1 to Setup.

## Add Registration.xml

1. In Solution Explorer select the Setup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the XML File template.
   2. Type Registration.xml in the Name field.
   3. Click the Add button.
4. Type the following in the Registration.xml editing window, replacing the placeholders in red with Registry Format GUIDs created with GUIDGEN.EXE and the previously determined Public Key Token.

<application title="Windows Media Center Step By Step"

id="PUT\_GUID\_2\_HERE">

<entrypoint id="PUT\_GUID\_3\_HERE"

addin="MyProject.MyAddIn, MyProject,Culture=Neutral,Version=1.0.0.0,PublicKeyToken=insert\_public\_key\_token\_here"

title="Windows Media Center Step By Step"

description="A Windows Media Center Application"

ImageUrl=".\Application.png">

<category category="More Programs"/>

</entrypoint>

</application>

## Add The Images Folder

1. In Solution Explorer select the MyProject project.
2. Select Project > New Folder from the menu.
3. In Solution Explorer rename NewFolder1 to Images.

## Add The Program Image

1. Using the image editor of your choice (Start > All Programs > Accessories > Paint works fine) create a square image and save it as Application.png.
2. Using Windows Explorer copy Application.png into the project folder named Images. **Note**: For more information on creating the application image see the Windows Media Center Software Development Kit topic titled ‘entrypoint Element’ under the ‘imageURL’ section.
3. In Solution Explorer select the MyProject project.
4. Select Project > Add Existing Item from the menu.
5. In the Add Existing Item dialog:
   1. Select Image Files in the Files of Type drop down list.
   2. Navigate to and select Application.png.
   3. Click the Add button.

# Enable UI Testing and MCML Verification

For this section we leave Microsoft Visual C# Express Edition 2008 to make some changes to the C# project file for which there is no UI provided in the Integrated Design Environment (IDE). This accomplishes two tasks: Add the MCML Preview Tool (MCMLPad.exe) as the program launched when you press F5 in Microsoft Visual C# Express Edition 2008 and which opens Test.mcml. In addition, the MCML Verifier tool is run against the MCML each time the solution is built to check the syntax of our MCML.

1. In Microsoft Visual C# Express Edition 2008 select File > Save All from the menu.
2. Select File > Close Solution from the menu and leave Visual C# Express Edition running.
3. Launch Notepad (Start > All Programs > Accessories > Notepad).
4. Select File > Open from the menu.
5. In the Open dialog:
   1. Select All Files in the Files of Type drop down list.
   2. Browse to the project folder in the Windows file system.
   3. Select MyProject.csproj.
   4. Click the Open button.
6. Add the following after all other Property Groups. The <StartArguments> element information should all be on a single line:

<PropertyGroup>

<UseVSHostingProcess>true</UseVSHostingProcess>

<StartWorkingDirectory>$(windir)\eHome</StartWorkingDirectory>

<StartArguments>-load:"file://$(MSBuildProjectDirectory)\Markup\Test.mcml" -assemblyredirect:"$(FullyQualifiedOutputPath)" -markupredirect:"resx://MyProject/MyProject.Resources/,file://$(MSBuildProjectDirectory)\Markup\,. mcml"</StartArguments>

<StartAction>Program</StartAction>

<StartProgram>$(windir)\eHome\McmlPad.exe</StartProgram>

</PropertyGroup>

1. Add the following line in red to the identified property group:

<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == '**Debug**|AnyCPU' ">

<DebugSymbols>true</DebugSymbols>

<DebugType>full</DebugType>

<Optimize>false</Optimize>

<OutputPath>bin\Debug\</OutputPath>

<FullyQualifiedOutputPath>$(MSBuildProjectDirectory)\bin\Debug</FullyQualifiedOutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants>

<ErrorReport>prompt</ErrorReport>

<WarningLevel>4</WarningLevel>

</PropertyGroup>

1. Add the following line in red to the identified property group:

<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == '**Release**|AnyCPU' ">

<DebugType>pdbonly</DebugType>

<Optimize>true</Optimize>

<OutputPath>bin\Release\</OutputPath>

<FullyQualifiedOutputPath>$(MSBuildProjectDirectory)\bin\Release</FullyQualifiedOutputPath>

<DefineConstants>TRACE</DefineConstants>

<ErrorReport>prompt</ErrorReport>

<WarningLevel>4</WarningLevel>

</PropertyGroup>

1. Select File > Save from the menu.
2. Close Notepad.
3. Return to Visual C# Express Edition 2008.
4. Select File > Open Project from the menu.
5. In the Open Project dialog:
   1. Browse to the project folder.
   2. Select MyProject.sln.
   3. Click the Open button.
6. In Solution Explorer select the MyProject project.
7. Select Project > MyProject Properties from the menu.
8. Select the Build Events tab in the project properties.
9. Add the following to the Post-build event command line.

%windir%\eHome\McmlVerifier.exe -verbose -assemblyredirect:"$(FullyQualifiedOutputPath)" -directory:"$(ProjectDir)Markup"

# Verify the MCML

1. Select View > Output from the menu.
2. Select Build > Rebuild Solution from the menu.
3. Observe something similar to the following in the output window in addition to the previous build output:

%windir%\eHome\McmlVerifier.exe -verbose -assemblyredirect:"C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\bin\Release" -directory:"C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup"

Microsoft (R) MediaCenter Markup Language (MCML) Verifier version 1.00

Copyright (C) Microsoft Corporation 2006. All rights reserved.

file://C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup\Button.mcml - OK!

file://C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup\Controls.mcml - OK!

file://C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup\Menu.mcml - OK!

file://C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup\RepeatItem.mcml - OK!

file://C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup\Styles.mcml - OK!

file://C:\Users\Windows User\Documents\Visual Studio 2008\Projects\MyProject\Markup\Test.mcml - OK!

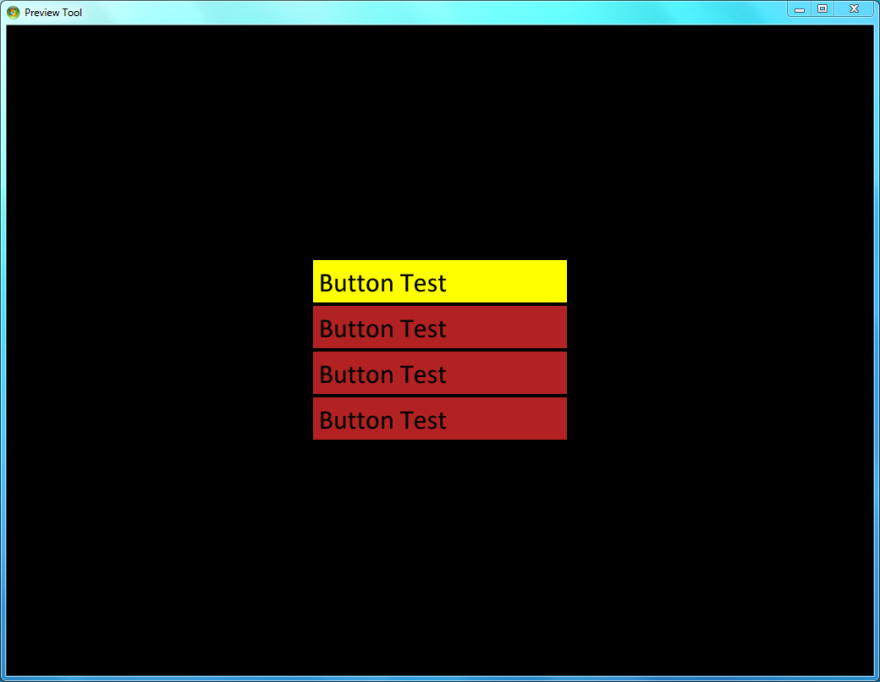
No errors detected.

========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========

# Test the UI

1. Select Debug > Start Debugging from the menu.
2. The Preview Tool will launch and you should see four buttons.
3. Test the buttons.

Note: At this point you are not actually running the application. The Preview Tool has loaded Test.mcml which provides a framework for testing the behavior of our visual design of user interface elements (in this case, Button.mcml). The buttons you see on screen behave as they will within the actual application (focus, non focus and selected colors will change) but do not call any Windows Media Center APIs.



# Create the Development Install Script

The development install script allows for full testing of the managed code assembly within Windows Media Center, simulates the results of a full installer program (\*.msi) and provides some modicum of testing of our registration XML created earlier.

Note: Notepad must be used because the Text file template in Visual Studio does not create an ANSI file which can cause issues when the script is running. Once you’ve created this file and incorporated into the project you can freely edit from within Visual C# 2008 Express Edition.

1. Launch Notepad (Start > All Programs > Accessories > Notepad).
2. Type the following.

@ECHO Off

ECHO.Usage: DevInstall.cmd [/u][/debug]

set CompanyName=MyCompany

set AssemblyName=MyProject

set RegistrationName=Registration

set ProgramImage=Application.png

set GacUtilPath=%ProgramFiles%

ECHO Determine whether we are on an 32 or 64 bit machine

if "%PROCESSOR\_ARCHITECTURE%"=="x86" if "%PROCESSOR\_ARCHITEW6432%"=="" goto x86

ECHO.On an x64 machine

set ProgramFilesPath=%ProgramFiles(x86)%

goto unregister

:x86

ECHO.On an x86 machine

set ProgramFilesPath=%ProgramFiles%

:unregister

ECHO.\*\*\* Unregistering and deleting assemblies \*\*\*

ECHO.Unregister and delete previously installed files (which may fail if nothing is registered)

ECHO.Unregister the application entry points

%windir%\ehome\RegisterMCEApp.exe /allusers "%ProgramFilesPath%\%CompanyName%\%AssemblyName%\%RegistrationName%.xml" /u

ECHO.Remove the DLL from the Global Assembly cache

"%GacUtilPath%\Microsoft SDKs\Windows\v6.0A\bin\gacutil.exe" /u "%AssemblyName%"

ECHO.Delete the folder containing the DLLs and supporting files (silent if successful)

rd /s /q "%ProgramFilesPath%\%CompanyName%\%AssemblyName%"

REM Exit out if the /u uninstall argument is provided, leaving no trace of program files.

if "%1"=="/u" goto exit

:releasetype

REM evaluate the second argument

if "%1"=="/debug" goto debug

ECHO.Using the release version of the binaries

set ReleaseType=Release

goto checkbin

:debug

ECHO.Using the Debug version of the binaries

set ReleaseType=Debug

:checkbin

if exist ".\bin\%ReleaseType%\%AssemblyName%.dll" goto register

ECHO.Cannot find %ReleaseType% binaries.

ECHO.Build solution as %ReleaseType% and run script again.

goto exit

:register

ECHO.\*\*\* Copying and registering assemblies \*\*\*

ECHO.Create the path for the binaries and supporting files (silent if successful)

md "%ProgramFilesPath%\%CompanyName%\%AssemblyName%"

ECHO.Copy the binaries to program files

copy /y ".\bin\%ReleaseType%\%AssemblyName%.dll" "%ProgramFilesPath%\%CompanyName%\%AssemblyName%\"

ECHO.Copy the registration XML to program files

copy /y ".\Setup\%RegistrationName%.xml" "%ProgramFilesPath%\%CompanyName%\%AssemblyName%\"

ECHO.Copy the program image to program files

copy /y ".\Images\%ProgramImage%" "%ProgramFilesPath%\%CompanyName%\%AssemblyName%\"

ECHO.Register the DLL with the global assembly cache

"%GacUtilPath%\Microsoft SDKs\Windows\v6.0A\bin\gacutil.exe" /if "%ProgramFilesPath%\%CompanyName%\%AssemblyName%\%AssemblyName%.dll"

ECHO.Register the application with Windows Media Center

%windir%\ehome\RegisterMCEApp.exe /allusers "%ProgramFilesPath%\%CompanyName%\%AssemblyName%\%RegistrationName%.xml"

:exit

1. Select File > Save from the menu.
2. Navigate to the project folder.
3. Type DevInstall.cmd in the File name field.
4. Select ANSI in the Encoding drop down list.
5. Click the Save Button.
6. Return to Visual C# Express Edition 2008.
7. In Solution Explorer select the MyProject project.
8. Select Project > Add Existing Item from the menu.
9. In the Add Existing Item dialog:
   1. Select All Files in the Files of Type drop down list.
   2. Navigate to and select DevInstall.cmd.
   3. Click the Add button.

# Development Install

1. Launch a command prompt with Administrator privileges (Start > All Programs > Accessories > Right-Click Command Prompt and select Run as administrator).
2. In the command prompt, navigate to the project path where DevInstall.cmd is located.
3. Type ‘DevInstall.cmd /debug’ without the quotes. **Note**: If you built the Release version type ‘DevInstall.cmd’ without the quotes.
4. Press the Enter key.
5. Observe something similar to the following in the command prompt window:

Usage: DevInstall.cmd [/u][/debug]

Determine whether we are on an 32 or 64 bit machine

On an x64 machine

\*\*\* Unregistering and deleting assemblies \*\*\*

Unregister and delete previously installed files (which may fail if nothing is registered)

Unregister the application entry points

Success

Remove the DLL from the Global Assembly cache

Microsoft (R) .NET Global Assembly Cache Utility. Version 3.5.30729.1

Copyright (c) Microsoft Corporation. All rights reserved.

Assembly: MyProject, Version=1.0.1.0, Culture=neutral, PublicKeyToken=e1b966ed354f6943, processorArchitecture=MSIL

Uninstalled: MyProject, Version=1.0.1.0, Culture=neutral, PublicKeyToken=e1b966ed354f6943, processorArchitecture=MSIL

Number of assemblies uninstalled = 1

Number of failures = 0

Delete the folder containing the DLLs and supporting files (silent if successful)

Using the Debug version of the binaries

\*\*\* Copying and registering assemblies \*\*\*

Create the path for the binaries and supporting files (silent if successful)

Copy the binaries to program files

1 file(s) copied.

Copy the registration XML to program files

1 file(s) copied.

Copy the program image to program files

1 file(s) copied.

Register the DLL with the global assembly cache

Microsoft (R) .NET Global Assembly Cache Utility. Version 3.5.30729.1

Copyright (c) Microsoft Corporation. All rights reserved.

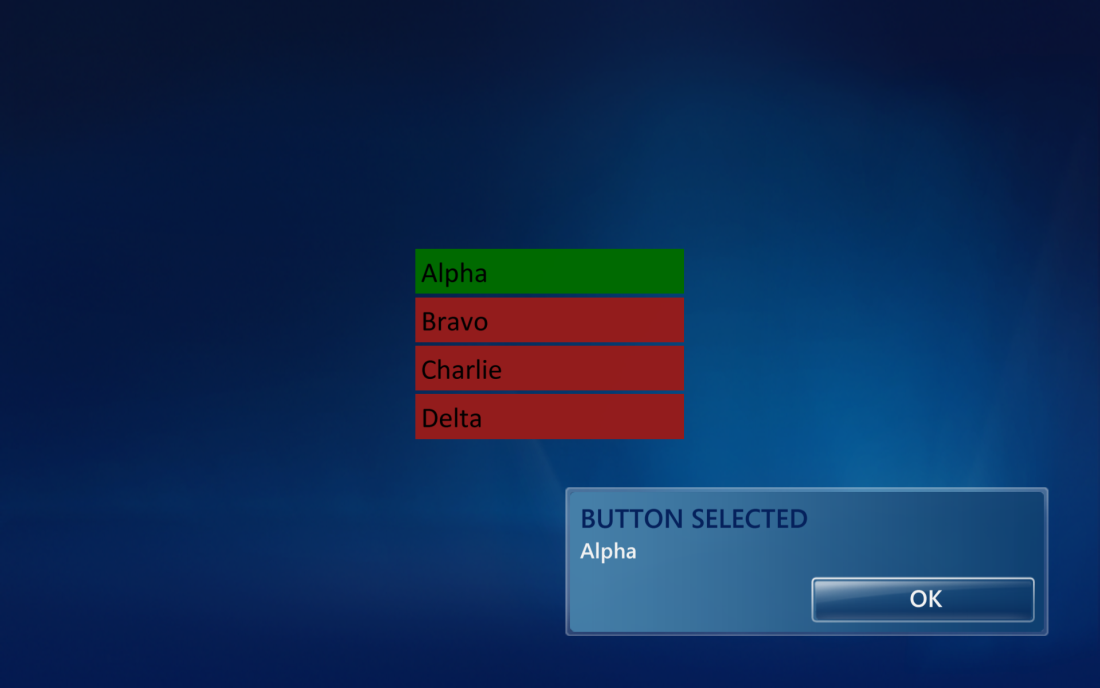
**Assembly successfully added to the cache**

**Register the application with Windows Media Center**

**Success**

# Launch the Application

1. Launch Windows Media Center.
2. Navigate to Extras > Extras Library.
3. Select the Windows Media Center Step By Step tile.
4. Selecting one of the buttons will result in a Windows Media Center dialog:



# Development Uninstall

1. Return to the command prompt with Administrator privileges.
2. Type ‘DevInstall.cmd /u /debug’ without the quotes. **Note**: If you built the Release version type ‘DevInstall.cmd /u’ without the quotes.
3. Press the Enter key.
4. Observe the following in the command prompt window:

\*\*\* Unregistering and deleting assemblies \*\*\*

Unregister and delete previously installed files (which may fail if nothing is registered)

Unregister the application entry points

**Success**

Remove the DLL from the Global Assembly cache

Microsoft (R) .NET Global Assembly Cache Utility. Version 3.5.21022.8

Copyright (c) Microsoft Corporation. All rights reserved.

Assembly: MyProject, Version=1.0.0.0, Culture=neutral, PublicKeyToken=2310e393a6000afe, processorArchitecture=MSIL

Uninstalled: MyProject, Version=1.0.0.0, Culture=neutral, PublicKeyToken=2310e393a6000afe, processorArchitecture=MSIL

**Number of assemblies uninstalled = 1**

Number of failures = 0

Delete the folder containing the DLLs and supporting files (silent if successful)

# Create the Setup Program Using WiX

Creating and testing a setup program allows for full testing of the functionality of the application in the exact same configuration that the end user will have when they install and use the application. Windows Installer XML (WiX) provides a declarative language to describe the setup functionality needed for the application and generates clean, reliable, serviceable and Windows Logo-compliant MSI files. In addition, WiX-based MSIs can be created within the Visual Studio 2008 Express Editions using a free toolset.

## Add License.rtf

License.rtf is used by the standard installer interface the user interacts with using mouse and keyboard.

1. Launch WordPad (Start > All Programs > Accessories > WordPad).
2. Type the following:

End User License Agreement

This is a placeholder End User License Agreement (EULA). It should be replaced with the final license for public distribution.

1. Select File > Save or Save As from the menu.
2. In the Save As dialog:
   1. Type License.rtf in the File name field.
   2. Select Rich Text Format (RTF) from the Save as type drop down list.
   3. Click Browse Folders.
   4. Navigate to the Setup folder for the project.
   5. Click the Save button.
3. In Solution Explorer select the Setup folder.
4. Select Project > Add Existing Item from the menu.
5. In the Add Existing Item dialog:
   1. Select All Files in the Files of Type drop down list.
   2. Navigate to and select License.rtf.
   3. Click the Add button.

## Add License.txt

License.txt is used by the InstallApplication Method within Windows Media Center to display the license agreement.

1. Launch Notepad (Start > All Programs > Accessories > Notepad).
2. Type the following:

End User License Agreement

This is a placeholder End User License Agreement (EULA). It should be replaced with the final license for public distribution.

1. Select File > Save or Save As from the menu.
2. In the Save As dialog:
   1. Type License.txt in the File name field.
   2. Select Text Documents (\*.txt) from the Save as type drop down list.
   3. Click Browse Folders.
   4. Navigate to the Setup folder for the project.
   5. Click the Save button.
3. In Solution Explorer select the Setup folder.
4. Select Project > Add Existing Item from the menu.
5. In the Add Existing Item dialog:
   1. Select All Files in the Files of Type drop down list.
   2. Navigate to and select License.txt.
   3. Click the Add button.

## Add LicenseAccessible.txt

LicenseAccessible.txt is used by accessibility tools when the InstallApplication Method is called so the vision impaired consumer can hear the license if it is over 8,000 characters. A pointer to the license online (at an HTTP address) should be provided in this file. If the license is shorter than 8,000 characters this file can be omitted.

1. Launch Notepad (Start > All Programs > Accessories > Notepad).
2. Type the following:

End User License Agreement

Please visit http://www.yourdomainhere.com/ to hear the End User License Agreement.

1. Select File > Save or Save As from the menu.
2. In the Save As dialog:
   1. Type LicenseAccessible.txt in the File name field.
   2. Select Text Documents (\*.txt) from the Save as type drop down list.
   3. Click Browse Folders.
   4. Navigate to the Setup folder for the project.
   5. Click the Save button.
3. In Solution Explorer select the Setup folder.
4. Select Project > Add Existing Item from the menu.
5. In the Add Existing Item dialog:
   1. Select All Files in the Files of Type drop down list.
   2. Navigate to and select LicenseAccessible.txt.
   3. Click the Add button.

## Add Build.cmd

1. Launch Notepad (Start > All Programs > Accessories > Notepad).
2. Type the following.

SET BUILD\_TYPE=Release

if "%1" == "Debug" set BUILD\_TYPE=Debug

REM Determine whether we are on an 32 or 64 bit machine

if "%PROCESSOR\_ARCHITECTURE%"=="x86" if "%PROCESSOR\_ARCHITEW6432%"=="" goto x86

set ProgramFilesPath=%ProgramFiles(x86)%

goto startInstall

:x86

set ProgramFilesPath=%ProgramFiles%

:startInstall

pushd "%~dp0"

SET WIX\_BUILD\_LOCATION=%ProgramFilesPath%\Windows Installer XML v3\bin

SET INTERMEDIATE\_PATH=..\obj\%BUILD\_TYPE%

SET OUTPUTNAME=..\bin\%BUILD\_TYPE%\Setup.msi

REM Cleanup leftover intermediate files

del /f /q "%INTERMEDIATE\_PATH%\\*.wixobj"

del /f /q "%OUTPUTNAME%"

REM Build the MSI for the setup package

REM This next part looks like 2 separate lines but should be typed in as a single line.

"%WIX\_BUILD\_LOCATION%\candle.exe" Setup.wxs -dBuildType=%BUILD\_TYPE% -out "%INTERMEDIATE\_PATH%\Setup.wixobj"

REM This next part looks like 4 separate lines but should be typed in as a single line.

"%WIX\_BUILD\_LOCATION%\light.exe" "%INTERMEDIATE\_PATH%\setup.wixobj" -cultures:en-US -ext "%ProgramFilesPath%\Windows Installer XML v3\bin\WixUIExtension.dll" -ext "%ProgramFilesPath%\Windows Installer XML v3\bin\WixUtilExtension.dll" -loc setup-en-us.wxl -out "%OUTPUTNAME%"

Popd

1. Select File > Save from the menu.
2. Navigate to the projects Setup folder.
3. Type Build.cmd in the File name field.
4. Select ANSI in the Encoding drop down list.
5. Click the Save Button.
6. Return to Visual C# Express Edition 2008.
7. In Solution Explorer select the Setup folder in the MyProject project.
8. Select Project > Add Existing Item from the menu.
9. In the Add Existing Item dialog:
   1. Select All Files in the Files of Type drop down list.
   2. Navigate to and select Build.cmd.
   3. Click the Add button.

## Add Setup.wxs

1. In Solution Explorer select the Setup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the XML File template.
   2. Type Setup.wxs in the name field.
   3. Click the Add button.
4. Type the following in the Setup.wxs editing window, replacing each instance of PUT\_GUID\_X\_HERE with a unique GUID without the curly braces. There are a total of five (5) GUIDs in Setup.wxs.

<?xml version="1.0" encoding="UTF-8"?>

<?define Property\_ProductVersion = "1.0.0.0" ?>

<?define Property\_ProductCode = "PUT\_GUID\_4\_HERE" ?>

<?define Property\_UpgradeCode = "PUT\_GUID\_5\_HERE" ?>

<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">

<Product Id="$(var.Property\_ProductCode)"

UpgradeCode="$(var.Property\_UpgradeCode)"

Name="!(loc.Property\_ProductName)"

Language="!(loc.Property\_ProductLanguage)"

Version="$(var.Property\_ProductVersion)"

Manufacturer="!(loc.Property\_CompanyName)">

<Package Description="!(loc.Package\_Description)"

Comments="!(loc.Package\_Comments)"

InstallerVersion="200"

Compressed="yes" />

<Upgrade Id="$(var.Property\_UpgradeCode)">

<UpgradeVersion Minimum="$(var.Property\_ProductVersion)"

OnlyDetect="yes"

Property="NEWERVERSIONDETECTED" />

<UpgradeVersion Minimum="0.0.0.0"

IncludeMinimum="yes"

Maximum="$(var.Property\_ProductVersion)"

IncludeMaximum="no"

Property="OLDERVERSIONBEINGUPGRADED" />

</Upgrade>

<Binary Id="License.txt"

SourceFile="..\setup\License.txt" />

<Binary Id="LicenseAccessible.txt"

SourceFile="..\setup\LicenseAccessible.txt" />

<CustomAction Id="CA\_BlockOlderVersionInstall"

Error="!(loc.LaunchCondition\_LaterVersion)" />

<CustomAction Id="CA\_ErrWrongWindowsVersion"

Error="!(loc.LaunchCondition\_WrongOSVersion)" />

<Condition Message="!(loc.LaunchCondition\_AdminPrivs)">

<![CDATA[Privileged]]>

</Condition>

<Directory Id="TARGETDIR"

Name="SourceDir">

<Directory Id="WindowsFolder"

Name="WINDOWS">

<Directory Id="EhomeFolder"

Name="eHome"/>

</Directory>

<Directory Id="ProgramFilesFolder"

Name="Program Files">

<Directory Id="MyCompany"

Name="MyCompany">

<Directory Id="APPROOTDIRECTORY"

Name="MyProject"/>

</Directory>

</Directory>

</Directory>

<DirectoryRef Id="APPROOTDIRECTORY">

<Component Id="MyProject.dll"

Guid="PUT\_GUID\_6\_HERE"

DiskId="1">

<File Id="MyProject.dll"

Name="MyProject.dll"

KeyPath="yes"

Source="..\bin\$(var.BuildType)\MyProject.dll"

Checksum="yes"

Vital="yes"

Assembly=".net"/>

</Component>

<Component Id="Registration.xml"

Guid="PUT\_GUID\_7\_HERE"

DiskId="1">

<File Id="Registration.xml"

Name="Registration.xml"

Source="..\Setup\Registration.xml"

Checksum="no" />

</Component>

<Component Id="Application.png"

Guid="PUT\_GUID\_8\_HERE"

DiskId="1">

<File Id="Application.png"

Name="Application.png"

Source="..\Images\Application.png"

Checksum="no" />

</Component>

</DirectoryRef>

<Feature Id="APPROOTFEATURE"

Title="MyProject"

Level="1">

<ComponentRef Id="MyProject.dll" />

<ComponentRef Id="Registration.xml" />

<ComponentRef Id="Application.png" />

</Feature>

<Media Id="1"

Cabinet="MyProject.cab"

EmbedCab="yes" />

<CustomAction Id="CA\_RegisterMceApp\_Unregister\_Install\_Cmd"

Property="CA\_RegisterMceApp\_Unregister\_Install"

Value="&quot;[REGISTERMCEAPP]&quot; /u /allusers &quot;[#Registration.xml]&quot;"/>

<CustomAction Id="CA\_RegisterMceApp\_Unregister\_Uninstall\_Cmd"

Property="CA\_RegisterMceApp\_Unregister\_Uninstall"

Value="&quot;[REGISTERMCEAPP]&quot; /u /allusers &quot;[#Registration.xml]&quot;"/>

<CustomAction Id="CA\_RegisterMceApp\_Register\_Cmd"

Property="CA\_RegisterMceApp\_Register"

Value="&quot;[REGISTERMCEAPP]&quot; /allusers &quot;[#Registration.xml]&quot;"/>

<CustomAction Id="CA\_RegisterMceApp\_Rollback\_Cmd"

Property="CA\_RegisterMceApp\_Rollback"

Value="&quot;[REGISTERMCEAPP]&quot; /u /allusers &quot;[#Registration.xml]&quot;"/>

<CustomAction Id="CA\_RegisterMceApp\_Unregister\_Install"

BinaryKey="WixCA"

DllEntry="CAQuietExec"

Execute="deferred"

Return="ignore"

Impersonate="no"/>

<CustomAction Id="CA\_RegisterMceApp\_Unregister\_Uninstall"

BinaryKey="WixCA"

DllEntry="CAQuietExec"

Execute="deferred"

Return="ignore"

Impersonate="no"/>

<CustomAction Id="CA\_RegisterMceApp\_Register"

BinaryKey="WixCA"

DllEntry="CAQuietExec"

Execute="deferred"

Return="check"

Impersonate="no"/>

<CustomAction Id="CA\_RegisterMceApp\_Rollback"

BinaryKey="WixCA"

DllEntry="CAQuietExec"

Execute="rollback"

Return="ignore"

Impersonate="no"/>

<InstallExecuteSequence>

<Custom Action="CA\_BlockOlderVersionInstall"

After="FindRelatedProducts">

<![CDATA[NEWERVERSIONDETECTED]]>

</Custom>

<RemoveExistingProducts After="InstallFinalize" />

<LaunchConditions After="AppSearch"/>

<Custom Action="CA\_ErrWrongWindowsVersion"

Before="CostInitialize"><![CDATA[(NOT VersionNT >= 601 OR NOT REGISTERMCEAPP) AND NOT Installed]]></Custom>

<Custom Action="CA\_RegisterMceApp\_Unregister\_Install\_Cmd"

After="CostFinalize">

<![CDATA[NOT REMOVE]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Unregister\_Uninstall\_Cmd"

After="CA\_RegisterMceApp\_Unregister\_Install\_Cmd">

<![CDATA[REMOVE AND ($Registration.xml = 2)]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Register\_Cmd"

After="CA\_RegisterMceApp\_Unregister\_Uninstall\_Cmd">

<![CDATA[NOT REMOVE]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Rollback\_Cmd"

After="CA\_RegisterMceApp\_Register\_Cmd">

<![CDATA[NOT REMOVE]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Unregister\_Uninstall"

Before="RemoveFiles">

<![CDATA[REMOVE AND ($Registration.xml = 2)]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Rollback"

After="InstallFiles">

<![CDATA[NOT REMOVE]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Unregister\_Install"

After="CA\_RegisterMceApp\_Rollback">

<![CDATA[NOT REMOVE]]>

</Custom>

<Custom Action="CA\_RegisterMceApp\_Register"

After="CA\_RegisterMceApp\_Unregister\_Install">

<![CDATA[NOT REMOVE]]>

</Custom>

</InstallExecuteSequence>

<InstallUISequence>

<Custom Action="CA\_BlockOlderVersionInstall"

After="FindRelatedProducts">

<![CDATA[NEWERVERSIONDETECTED]]>

</Custom>

<Custom Action="CA\_ErrWrongWindowsVersion"

Before="CostInitialize"><![CDATA[(NOT VersionNT >= 601 OR NOT REGISTERMCEAPP) AND NOT Installed]]></Custom>

</InstallUISequence>

<Property Id="ARPHELPLINK"

Value="!(loc.Property\_ArpHelpLink)" />

<Property Id="ARPURLINFOABOUT"

Value="!(loc.Property\_ArpUrlInfoAbout)" />

<Property Id="ALLUSERS">

<![CDATA[1]]>

</Property>

<Property Id="MCEINSTALLVERSION"

Secure="yes">

<RegistrySearch Id="MceInstallRegKey"

Root="HKLM"

Key="SOFTWARE\Microsoft\Windows\CurrentVersion\Media Center"

Name="Ident"

Type="raw"/>

</Property>

<Property Id="REGISTERMCEAPP"

Secure="yes">

<DirectorySearch Id="EhomeDirectory"

Path="[WindowsFolder]\ehome">

<FileSearch Id="RegisterMceAppExe"

Name="RegisterMceApp.exe" />

</DirectorySearch>

</Property>

<Property Id="WIXUI\_INSTALLDIR"

Value="APPROOTDIRECTORY"/>

<Property Id="d4276fcf18d5470b97f3fc6c36408694"

Value="License.txt"/>

<Property Id="aefe520afb0145c1bc61ca0463a29e3c"

Value="LicenseAccessible.txt"/>

<UIRef Id="WixUI\_InstallDir"/>

<WixVariable Id="WixUILicenseRtf"

Value="License.rtf" />

</Product>

</Wix>

## Add Setup-en-us.wxl

1. In Solution Explorer select the Setup folder.
2. Select Project > Add New Item from the menu.
3. In the Add New Item dialog:
   1. Select the XML File template.
   2. Type Setup-en-us.wxl in the name field.
   3. Click the Add button.
4. Type the following in the Setup-en-us.wxl editing window:

<?xml version="1.0" encoding="utf-8"?>

<WixLocalization Culture="en-us"

xmlns="http://schemas.microsoft.com/wix/2006/localization">

<String Id="LaunchCondition\_AdminPrivs">You must have Administrative rights on this machine to install [ProductName].</String>

<String Id="LaunchCondition\_LaterVersion">A later version of [ProductName] is already installed.</String>

<String Id="LaunchCondition\_WrongOSVersion">[ProductName] can only be installed on a system with Windows Media Center.</String>

<String Id="Property\_ArpHelpLink">http://msdn.microsoft.com/mce/</String>

<String Id="Property\_ArpUrlInfoAbout">http://msdn.microsoft.com/mce/</String>

<String Id="Property\_ProductLanguage">1033</String>

<String Id="Property\_ProductName">Windows Media Center Step By Step</String>

<String Id="Property\_CompanyName">MyCompany</String>

<String Id="Package\_Description">MyProject Description</String>

<String Id="Package\_Comments">MyProject Comments</String>

</WixLocalization>

## Add WiX Post-build Event

1. In Solution Explorer select the MyProject project.
2. Select Project > MyProject Properties from the menu.
3. Select the Build Events tab.
4. Add the following in the Post-build event command line field:

"$(ProjectDir)Setup\build.cmd" $(ConfigurationName)

## Build the Solution

1. Select View > Output from the menu.
2. Select Build > Rebuild Solution from the menu.
3. Observe something similar to the following in the output window:\

Microsoft (R) Windows Installer Xml Compiler version 3.0.5224.0

Copyright (C) Microsoft Corporation. All rights reserved.

Setup.wxs

Microsoft (R) Windows Installer Xml Linker version 3.0.5224.0

Copyright (C) Microsoft Corporation. All rights reserved.

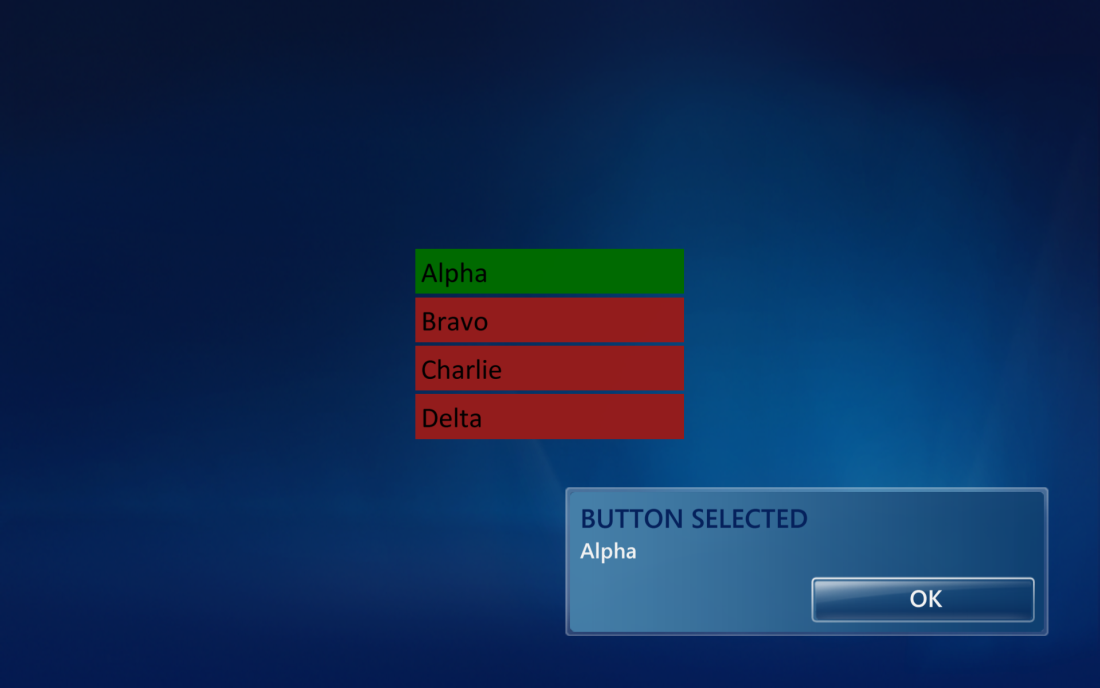
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

## Install the Application

1. Navigate to the project’s \bin\Release\ or \bin\Debug folder.
2. Launch Setup.msi
3. Step through the installation wizard.

# Test the Application

1. Launch Windows Media Center.
2. Navigate to Extras > Extras Library.
3. Select the Step By Step tile.
4. Selecting one of the buttons will result in a Windows Media Center dialog:



# Congratulations!

You’ve built and installed your first Windows Media Center application.

# Need Help?

Visit the discussion forums at <http://discuss.mediacentersandbox.com>.