Creating an Application Launcher by Using the Sample Home Screen in Silverlight for Windows Embedded

Windows Embedded Compact 7 Technical Article

Published: March 2011

Applies To: Windows Embedded Compact 7

Abstract

In Windows Embedded Compact 7, you can design and implement a Microsoft Silverlight-based application launcher by using Microsoft Silverlight for Windows Embedded. A Silverlight-based application launcher starts at device startup and provides a main menu of items that each launch an application installed on the device. It also provides a status bar that contains widgets for displaying status of values, such as volume. Users can use a Silverlight-based application launcher to start applications or to open the Control Panel and configure device settings.

Building a Silverlight-based application launcher from the ground up can be time-consuming, especially if you are new to developing applications by using the Silverlight for Windows Embedded C++ class library. This paper describes how to reduce development time by starting from the sample home screen available in Windows Embedded Compact 7.

By creating a Silverlight-based application launcher, you can create a UI for your device that reflects the functionality that you added to your device, and that has a visual design that you selected and tailored to meet your users' needs.

You can then redesign the UI of the sample home screen to fit your needs.

Introduction

By starting with the sample home screen, you can reduce your development time by using the predesigned XAML projects and tested source code in Windows Embedded Compact 7. Then, you can customize the XAML, so that the Silverlight-based application launcher UI meets the needs of your users and reflects the functionality that you will include on your device.

The sample home screen provides a main menu that contains items that launch the Silverlight-based applications on a device. The sample home screen also provides a set of status widgets, which resemble the status icons in the notification tray on the Windows desktop. The default includes widgets that display volume, Bluetooth status, network status, and battery status. When you create a Silverlight-based application launcher by using the sample home screen, you will need to redesign the UI, add a battery status settings application, modify the main menu, and add or modify menu items. You can also choose to add additional custom widgets or custom settings applications to the application launcher. The development process generally requires from five to nine months.

In addition, you can create more than one UI design and implement theme switching to allow your users to change the UI design for the Silverlight-based application launcher by selecting a different theme. Note that theme switching affects only the Silverlight-based application launcher. For information about changing the UI for individual Silverlight-based applications, see Editing the XAML Projects for Silverlight-Based Functionality.

Sample Home Screen Overview

Device users can browse and start applications on your consumer device through a unique sample home screen available in Windows Embedded Compact 7. The sample home screen differs from the traditional Windows desktop shell and is customizable through XAML.

The sample home screen provides three different screen resolutions and four different visual themes for each screen resolution for you to choose from.

The sample home screen provides support for the following optional Windows Embedded Compact applications and functionality:

- **Music Player** a Silverlight-based application that plays music files on a Windows Embedded Compact device.
- **Photo Viewer** a Silverlight-based application for users to view photographs on a Windows Embedded Compact device.
- **Video Player** a Silverlight-based application for users to view video files on a Windows Embedded Compact device.
- **Web Browser** a browser application that supports Internet Explorer 7 for users to browse the web from a Windows Embedded Compact device.
- **Control Panel** a set of control panel applications for users to configure device settings.
- **Bluetooth Settings** a settings application for users to pair with other Bluetooth wireless technology-enabled devices.

- **Software Input Panel** an on-screen keyboard for users to input text by interacting with the UI.

Prerequisites

To create a Silverlight-based application launcher by following the steps in this tutorial, you must install the following items in order:

1. Microsoft Expression Blend 3, downloaded from Microsoft.com (http://go.microsoft.com/fwlink/?LinkId=204751).
2. Visual Studio 2008 with SP1.
3. Windows Embedded Compact 7 and Windows Embedded Silverlight Tools.
4. Virtual CEPC. For more information, see Getting Started with Virtual CEPC (http://go.microsoft.com/fwlink/?LinkId=199788).

The Expression Blend templates are installed at **%ProgramFiles%\Microsoft Expression\Blend 3\ProjectTemplates\en\Windows Embedded Silverlight Tools\Application**.

If you are using a non-U.S. English language version of Expression Blend 3, you must manually copy the template files from **%ProgramFiles%\Microsoft Expression\Blend 3\ProjectTemplates\en\Windows Embedded Silverlight Tools\Application** to a directory you create in your local user directory: **%Users%***user-name***\Documents\Expression\Blend 3\ProjectTemplates\Windows Embedded Silverlight Tools\Applications**.

After you manually copy the files to the new directory path, Expression Blend can display the templates in the **New Project** dialog box.

Creating a Silverlight-Based Application Launcher

Follow the steps in this tutorial to create a Silverlight-based application launcher by starting with the source code and XAML projects for the sample home screen in Windows Embedded Compact 7.

1. Select a Display Screen Size and Preliminary Theme Design
2. Change the UI Design of the Sample Home Screen XAML Project
3. Customize the Background Image
4. Add a Battery Status Settings Application to the Application Launcher
5. Configure Items in the Menus for Launching Applications
6. Localize, Build, and Run the Silverlight-Based Application Launcher
7. (Optional) Create Functionality for Users to Switch the Theme

Step 1: Select a Display Screen Size and Preliminary Theme Design

You can choose from three different screen resolutions and four different visual themes for each screen resolution.

Each display screen size supports a specific layout, which determines how elements are sized, positioned, and arranged on-screen, that is appropriate for a display screen that has a specific screen resolution.

Each theme design for a display screen size provides a visual appearance for the UI elements, with customized colors, textures, and icons.

Before you create a Silverlight-based application launcher by using the sample home screen, you must select which screen size and which theme design you want to start with. Then you can modify the sample home screen to meet your needs.

Determine the Display Screen Size

Use the following table to determine which screen size to use.

Display Screen Size	Characteristics
Small	• Screen resolution of 480 × 640 pixels • Portrait orientation
Medium	• Screen resolution of 800 × 480 pixels • Landscape orientation
Large	• Screen resolution of 1920 × 1080 pixels • Landscape orientation

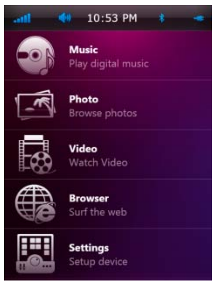
Select the Preliminary Theme Design

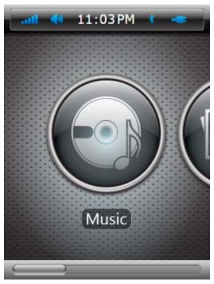
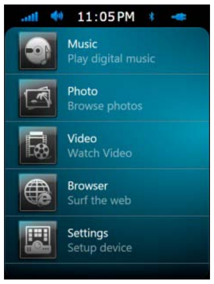
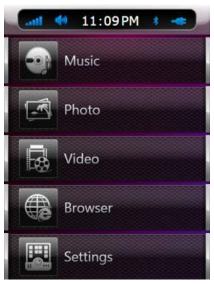
After you select a display screen size for your theme, select the preliminary theme design to work from.

The theme designs available for each display screen size are described below.

Theme Designs for Small Display Screens

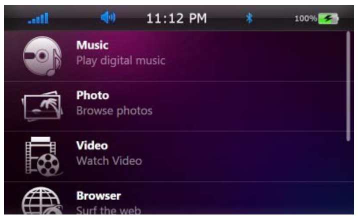
The following table shows the theme designs available for the small display screen size:

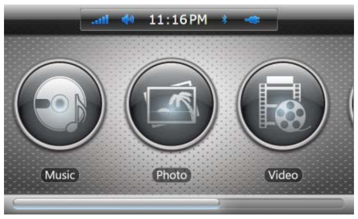
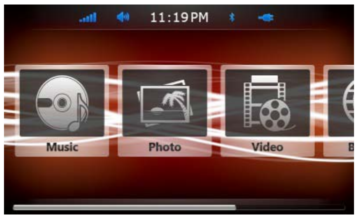
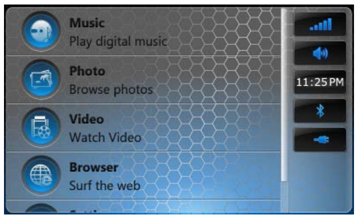
Theme	Appearance
Default Theme	 10:53 PM Music Play digital music Photo Browse photos Video Watch Video Browser Surf the web Settings Setup device

Theme	**Appearance**
Theme 1	
Theme 2	 11:05PM Music Play digital music Photo Browse photos Video Watch Video Browser Surf the web Settings Setup device
Theme 3	

Theme Designs for Medium Display Screens

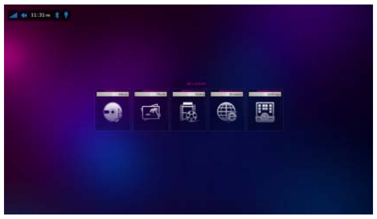
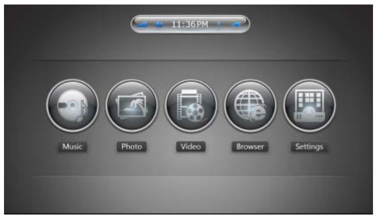
The following table shows the theme designs available for the medium display screen size:

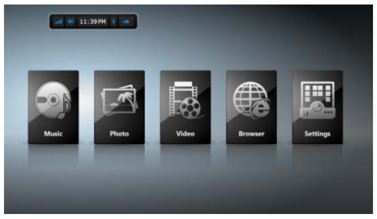
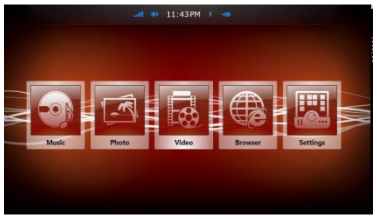
Theme	**Appearance**
Default Theme	 11:12 PM 100% Music Play digital music Photo Browse photos Video Watch Video Browser Surf the web

Theme	Appearance
Theme 1	
Theme 2	
Theme 3	 Music Play digital music Photo Browse photos Video Watch Video Browser Surf the web 11:25PM

Theme Designs for Large Display Screens

The following table shows the theme designs available for the large display screen size:

Theme	Appearance
Default Theme	
Theme 1	

Theme	**Appearance**
Theme 2	
Theme 3	

Create an OS Design that Uses the Preliminary Theme and Display Screen Size

After you decide which theme and display screen size to start with, create an OS design by using the Embedded Device with Silverlight XAML design template. When you create the OS design, select the display screen size and preliminary theme, and then customize the theme to meet the needs of your users.

If you decide to implement theme switching, the preliminary theme you choose will be the default theme for your Windows Embedded Compact device.

Create an OS Design that uses your preliminary theme and display screen size

1. In Microsoft Platform Builder, on the **File** menu, point to **New**, and then select **New Project**.
2. Select **Platform Builder**, type a name for the project, and then click **OK**.
3. Click **Next**.
4. On the **Board Support Packages** page, click **Virtual PC: x86**, and then click **Next**.
5. On the **Design Templates** page, select **Embedded Device with Silverlight XAML,** and then click **Next**.
6. On the **UI Screen Size** page, select one of the display screen sizes:
   - For a small display screen, select 640 × 480 Portrait (IMG_SCREEN_SMALL) for a small display screen
   - For a medium display screen, select 800 × 480 Landscape (IMG_SCREEN_MEDIUM) for a medium display screen
   - For a large display screen, select 1920 × 1080p Landscape (IMG_SCREEN_LARGE) for a large display screen
7. Click **Next**.
8. On the **Applications** page, expand **Sample Home Screen** and select one of the following preliminary themes:

- Default Theme (IMG_XRST_THEME_0)
- Theme 1 (IMG_XRST_THEME_1)
- Theme 2 (IMG_XRST_THEME_2)
- Theme 3 (IMG_XRST_THEME_3)

9. Select the Silverlight-based applications that you want to include in the OS design and then click **Next**.
10. Click **Finish**.
11. In the **Catalog Item Notification** dialog box, click **Acknowledge**.
12. In **Catalog Items View**, browse to **<OSDesignName>\Core OS\Windows Embedded Compact\Core OS Services\Kernel Functionality** and then select the Target Control Support catalog item (SYSGEN_SHELL) to use the Target Control command-line prompt window in Platform Builder.
13. On the **File** menu, click **Save All**.
14. On the **Build** menu, click **Build Solution**.

Step 2: Change the UI Design of the Sample Home Screen XAML Project

After you determine the display screen size and preliminary theme design and create an OS design, use Expression Blend 3 to start applying changes to the foundational XAML for the sample home screen.

Note Alternatively, you can use Expression Blend 4; however, you must use an Expression Blend project template that is in Microsoft Silverlight 3 format.

Each preliminary theme design includes the base elements of a Silverlight-based application launcher for a Windows Embedded Compact device. You can use Expression Blend to change the visual appearance or the layout of the elements, replace the elements, and add additional elements to the theme design.

For more information about editing a XAML project, see Editing XAML at MSDN.

Editing the Expression Blend XAML Project for the Sample Home Screen

To create a custom theme by starting with a preliminary theme for the sample home screen, you must copy the XAML project to a new folder, edit the XAML by using Expression Blend 3, and then add the updated XAML project to the OS design. Then, you must rebuild the theme DLL by using the command-prompt window in Platform Builder before you rebuild your run-time image.

You can implement a development process with both a UI designer and developer in which the UI designer edits the XAML project and sends updated versions of the XAML UI to the developer at different stages of the development process. Then, the developer adds the updated XAML UI to the run-time image, and tests functionality and graphics performance. This process is sometimes referred to as the designer/developer workflow.

Caution Elements that have **x:Name** attribute values prefixed with an underscore (_) might have C++ code-behind attached to them. If you modify the **x:Name**

attribute value or delete the element, the sample home screen might generate a run-time exception, unless you modify the C++ source code in %_WINCEROOT%\Public\Shell\Oak\XRShell.

To prepare to edit the XAML project for a preliminary theme

1. Browse to the directory location of the Expression Blend XAML project that corresponds to the display screen size and theme you identified in Step 1 of this tutorial.
2. Copy the entire directory to a new working folder. For example, copy C:\WINCE700\Public\Shell\Oak\XRShell\Assets\Medium_Theme1 to C:\MyTheme.
3. Create an archived version of the directory that includes the original XAML project that you will edit. For example, archive the contents of the Medium directory, or check the contents of the Medium directory into a source control application.

To edit the XAML project for a preliminary theme

1. Open Microsoft Expression Blend 3.
2. On the **File** menu, click **Open Project/Solution**.
3. Browse to the XAML project file (.csproj) that you copied to your working folder.
4. Click **Open**.
5. In the **Projects** tab, double-click a XAML file to edit.
6. Edit the XAML for the file by using the Expression Blend development environment. For tutorials on editing XAML, see Learn Expression Blend.
7. On the **File** menu, click **Save All**.

After the XAML project is updated with the redesigned UI, rebuild the .dll file for the corresponding theme, copy the updated files to the release directory, and rebuild the run-time image.

To add the updated XAML UI for the Silverlight-based application launcher to the run-time image

1. Copy the directory and its contents from your working folder to its original location in the %_WINCEROOT% directory, in %_WINCEROOT%\Public\Shell\Oak\XRShell\Assets.

   Note When you copy an updated XAML project to the Public directory folder, Windows Embedded Compact adds it to the run-time image for each OS design that you build on your computer.

2. Copy the updated directory and its contents to its original location.
3. If the XAML UI uses source graphic files (for example, .gif, .bmp, or .png files), do the following:
   a. Add the graphic files to a directory. For example, C:\ImageFiles.
   b. In Solution Explorer, select the OS Design name.
   c. On the **Project** menu, click **Properties**.
   d. Expand **Configuration Properties**.
   e. Click **Custom Build Actions**.
   f. In **Build Step**, select **Pre-Make Image**.
   g. Click **New**.
   h. Type a build step to copy the graphic files to the release directory. For example:

```
copy C:\ImageFiles\Theme1.png $(_FLATRELEASEDIR)\Theme1.png
copy C:\ImageFiles\Theme2.png $(_FLATRELEASEDIR)\Theme2.png
copy C:\ImageFiles\Theme3.png $(_FLATRELEASEDIR)\Theme3.png
```

i. Click **OK**.

4. In Microsoft Platform Builder, open a build window.
   - In Solution Explorer, right-click the OS Design name, and select **Open Release Directory in Build Window**.
5. At the command prompt, browse to %_WINCEROOT%\Public\Shell\Oak\XRShell\Themes\<*Directory_Name*>, where <*Directory_Name*> is the name of the directory that contains the updated XAML project.
6. At the command prompt, type **build**.
7. In Microsoft Platform Builder, on the **Build** menu, click **Copy Files to Release Directory**.
8. On the **Build** menu, click **Make Run-Time Image**.
9. Start the Virtual CEPC that you have preconfigured and connected to Platform Builder, as described in Getting Started with Virtual CEPC (http://go.microsoft.com/fwlink/?LinkId=199788). For example, to start Virtual CEPC on a Windows 7 computer:
   - On the **Start** menu, point to **All Programs**, point to **Windows Virtual PC**, and then click **Windows Virtual PC**.
   - Double-click the Virtual CEPC you have preconfigured.
10. On the **Target** menu, click **Attach Device**.

The run-time image loads on Virtual CEPC and displays the updated XAML UI.

Next, you can test and optimize the graphics performance of the updated XAML UI. For more information, see Performance Tuning Guide for Silverlight for Windows Embedded (http://go.microsoft.com/fwlink/?LinkId=205643).

Editing the XAML Projects for Silverlight-Based Functionality

You can also edit the default XAML for Silverlight-based functionality, such as the sample applications that are launched by the sample home screen. By doing this, you can ensure that the UI design for the functionality is consistent with the UI design for the application launcher. The process for this task is similar to the procedure outlined in Editing the Expression Blend XAML Project for the Sample Home Screen.

In Windows Embedded Compact 7, the Expression Blend XAML projects that you can edit are the Bluetooth Settings, Music Player, Video Player, Photo Viewer, Software-based input panel, and Silverlight-based Message Boxes applications. However, you cannot edit projects for Music Player, Video Player, or Photo Viewer for large display screens.

Step 3: Customize the Background Image

You can provide a custom background image for the Silverlight-based application launcher. For example, you can create a design that uses your company's logo to create a brand identity for your device.

The image files for the sample home screen are located in the Images subfolder for each theme directory in %_WINCEROOT%\public\shell\oak\XRShell\Assets. You can replace the default image files with your own images.

You can replace the default background image Backgroundimage.png with your own custom image.

To change the background image for the application launcher

1. Design a background image for the application launcher.
2. Do one of the following:
   - Save the image as a file in the Images subfolder for your theme, and name the file Backgroundimage.png
   - In Expression Blend, open Mainpage.xaml and, for the Grid element named "DesktopBackground," replace the **Source** attribute value with the path to the new image file.

Step 4: Add a Battery Status Settings Application to the Application Launcher

By default, the sample home screen provides a status widget that displays battery status of your device. For example, the battery status widget can indicate power status to the user by displaying the percentage of battery power remaining.

By default, the battery status widget on the main page does not open a settings application when a user clicks on it. You can create a battery status settings application that allows the user to configure settings such as the power management plan, the brightness of the backlight, or the timeout period before a device shuts off due to inactivity.

Implement a Battery Settings Application

To implement a battery settings application, provide the following functionality:

1. An application that provides UI for users to view and configure power settings.
2. Functionality that monitors battery power and updates the battery widget's UI.
3. Functionality that connects the battery widget to the application.

Create the Application

You can create a custom settings application that you can launch when the user clicks the battery status widget in the status bar of a sample home screen. For more information, see A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink/?LinkID=189508) and Silverlight for Windows Embedded Developer's Guide (http://go.microsoft.com/fwlink/?LinkId=191039).

Configure the Battery Status Widget

Configure the battery status widget so that it monitors battery power status when XRShell.exe is loaded, and so that the battery settings application opens when a user clicks on the widget.

For an example of a status widget implementation, see the source code for the volume status widget, located at %_WINCEROOT%\Public\Shell\Oak\XRShell\Controls\Volume\Status_itemvolume.cpp.

To implement code for the battery status widget

1. In %_WINCEROOT%\Public\Shell\Oak\XRShell\Controls\Power, implement a C++ class for the battery status widget. For an example, you can use the **Status_ItemVolume** class implemented in %_WINCEROOT%\public\shell\oak\XRShell\controls\volume\Status_itemvolume.cpp.
2. In the C++ class you create for the battery status widget, provide a custom implementation of the following methods at minimum:
   - **OnClick method**

     Your method implementation of **OnClick** must handle the **Click** event, which occurs when the user clicks the status widget on the main page of the sample home screen. In the method implementation, you can launch the .exe for the battery status settings application, for users to view or configure power management settings.
   - **Start method**

     Starts the widget when the main page starts. In your method implementation, you can add code to start monitoring the status of a setting. Then, you can use status monitoring information to update the UI of a status widget, to communicate status of a setting. For example, the battery UI could display an updated percentage (%) value to indicate current power status.
   - **Stop method**

     Stops the status widget. In your method implementation, you can add code to stop activity to monitor status of a setting, because the application launcher is exiting and you must minimize memory and CPU usage for your status widget.

   You can also implement the following optional method:
   - **OnLoaded method**

     In your method implementation, you can add code that indicates how you want the status widget to respond to user interaction. For example, if it is a button, you can attach an **OnClick** event handler to the button object.

Connect the Application to the Widget

Connect the application to the widget's Click event so that it opens when the user clicks on the widget.

To connect the custom settings application to the battery widget

1. Create a GUID for the .exe file for the settings application. For example, use the Guidgen Tool.

2. Add a registry subkey that has the GUID for your settings application, and add registry settings for your settings application. For more information, see Executable Control Panel Registry Settings in Control Panel Registry Settings in the Windows Embedded Compact 7 Documentation.
3. Open Status_itempower.cpp, located in %_WINCEROOT%\Public\Shell\Oak\XRShell\Controls\Power\Status_itempower.cpp.
4. Locate the **LaunchApplication** function call, in **Status_ItemPower::OnClick** event handler.

```
CHR( m_pShell->LaunchApplication(L"\\windows\\CtlPnl2.exe",
        L"", &AppHandle, &AppID) );
```

5. Set the value of the second parameter of **LaunchApplication** to a string that contains the GUID that you created.

Step 5: Configure Items in the Menus for Launching Applications

In the menu for the sample home screen, menu items and settings items will appear by default for Windows Embedded Compact applications and control panel settings applications that you specifically include in the run-time image.

You can modify the default settings for the items in the menus. For example, you can change the order of menu items, or edit or remove the text descriptions that accompany each menu item.

You can also add menu items to launch custom Silverlight-based applications that you develop, build, and include in the run-time image.

Change the Order of Menu Items

You can reorganize items in the main menu or control panel menu.

To change the order of menu items displayed in the main menu

1. Open Main_Default.xml, located in %_WINCEROOT%\Public\Shell\Oak\XRShell\Src.
2. Rearrange the order of Item elements, including @CESYSGEN conditional statements that enclose elements that are dependent on build variables, as follows:
   - Item elements defined at the top of the XML element tree appear at the beginning of the menu.
   - Item elements defined at the bottom of the XML element tree appear at the end of the menu.

To change the order of menu items displayed in the control panel menu

1. Open Setting_Default.xml, located in %_WINCEROOT%\Public\Shell\Oak\XRShell\Src.
2. Rearrange the order of SettingItem elements, including @CESYSGEN conditional statements that enclose elements that are dependent on build variables, as follows:
   - SettingItem elements defined at the top of the XML element tree appear at the beginning of the menu.

- SettingItem elements defined at the bottom of the XML element tree appear at the end of the menu.

Delete Text in a Menu Item

You can delete the title or descriptive text that accompanies each Path element in a menu.

To delete text in a menu item

1. Open the .xrsl file that defines the menu item. It can be one of the following:

File	Menu item
Main_browser.xrsl	The Browser item in the main menu.
Main_music.xrsl	The Music item in the main menu.
Main_photo.xrsl	The Photo item in the main menu.
Main_settings.xrsl	The Settings item in the main menu.
Main_video.xrsl	The Video item in the main menu.
Setting_audio.xrsl	The Audio item in the control panel menu.
Setting_bluetooth.xrsl	The Bluetooth item in the control panel menu.
Setting_datetime.xrsl	The Date and Time item in the control panel menu.
Setting_display.xrsl	The Display item in the control panel menu.
Setting_ethernet.xrsl	The Ethernet item in the control panel menu.
Setting_networking.xrsl	The Networking item in the control panel menu.
Setting_regional.xrsl	The Regional Settings item in the control panel menu.
Setting_wifi.xrsl	The Wi-Fi item in the control panel menu.

Sample home screen .xrsl files are located at %_WINCEROOT%\public\shell\oak\XRShell\src.

2. Remove the text that identifies the menu item:
   a. In the .xrsl file, locate the **Name** element.
   b. Delete the content in the **Name** element start and end tags, convert the start and end tags to an empty element, and then add a **Text** attribute that is set to an empty string as follows:

      ```
      <Name _locId="IDName" Text=" " />
      ```

3. Remove the text that describes the menu item:
   a. In the .xrsl file, locate the **Description** element.
   b. Delete the content in the **Description** element start and end tags, convert the start and end tags to an empty element, and then add a **Text** attribute that is set to an empty string as follows:

      ```
      <Description _locId="IDDescription" Text=" " />
      ```

Add a New Menu Item

You can add new items to the menu that launch new Silverlight-based applications. Menu items can include both text and a XAML Path element that resembles an icon.

To create a custom Path element that resembles an icon for your menu item

1. In Expression Blend, on the **File** menu, click **New Project**.
2. In **Project types**, select **Windows Embedded**.
3. Type a project name, and then click **OK**.
4. Paste the following XAML, which contains a sample home screen Path element, into Expression Blend:

   ```
   <UserControl
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      x:Class="TestPathElement2.MainPage"
      Width="640" Height="480">

   <Grid x:Name="LayoutRoot" Background="Gray">
   <Path
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   Data="F1M168.556,119.535L168.804,120.336C169.268,122.352,169.536,124
   .303,169.571,126.264L169.519,127.503
   167.379,131.946C164.669,136.934,161.492,141.63,157.905,145.976L153,1
   51.373 153,148.227 153,145.433
   154.819,143.432C159.954,137.209,164.211,130.234,167.399,122.697z
   M153,101.029C153,101.029,163.571,105.171,166.857,114.028L167.569,116
   .336
   165.53,121.907C162.419,129.262,158.265,136.069,153.254,142.142L153,1
   ```

```
42.421 153,130.948C153,115.406,153,101.029,153,101.029z
M90,77.5C83.096,77.5 77.5,83.096 77.5,90 77.5,96.904 83.096,102.5
90,102.5 96.904,102.5 102.5,96.904 102.5,90 102.5,83.096 96.904,77.5
90,77.5z M90,76C97.732,76 104,82.268 104,90 104,97.732 97.732,104
90,104 82.268,104 76,97.732 76,90 76,82.268 82.268,76 90,76z
M90,73.996C81.161,73.996 73.996,81.161 73.996,90 73.996,98.839
81.161,106.004 90,106.004 98.839,106.004 106.004,98.839 106.004,90
106.004,81.161 98.839,73.996 90,73.996z
M145.118,63.013C145.118,63.013 148.096,63.013 148.096,63.013
148.929,68.966 149.048,76.47 155.241,83.973 160.004,89.689
165.364,94.809 169.056,99.453 173.938,105.528 177.391,113.15
177.391,120.891 177.391,133.277 171.438,145.541 167.387,152.33
167.387,152.33 165.482,152.33 165.482,152.33 168.341,145.9
174.057,135.3 173.579,123.988 173.342,117.676 170.96,110.887
166.912,105.886 162.386,100.05 154.763,95.405 148.096,94.93
148.096,94.93 148.096,162.453 148.096,162.453 148.096,167.813
144.88,172.695 140.236,176.268 135.71,179.84 129.754,181.984
124.516,181.984 121.18,181.984 118.083,180.913 115.941,179.125
113.678,177.339 112.368,174.72 112.368,171.624 112.368,166.742
115.702,161.857 120.228,158.286 124.754,154.593 130.469,152.212
135.233,152.212 139.402,152.212 142.974,152.808 145.118,154.831
145.118,154.831 145.118,63.013 145.118,63.013z
M90,61.988C105.47,61.988 118.012,74.53 118.012,90 118.012,105.47
105.47,118.012 90,118.012 74.53,118.012 61.988,105.47 61.988,90
61.988,74.53 74.53,61.988 90,61.988z M90,60C73.431,60 60,73.431
60,90 60,106.569 73.431,120 90,120 106.569,120 120,106.569 120,90
120,73.431 106.569,60 90,60z
M90,8.028C135.272,8.028,171.972,44.728,171.972,90L171.671,95.963
167.342,91.008C163.169,86.436 159.143,82.186 157,78.686
154.091,73.934 151.429,59 151.429,59 151.429,59 140.964,59
140.964,59 140.964,59 140.964,148.714 140.964,148.714
140.964,148.714 133.286,146.571 125.714,150 118.143,153.429
110.398,158.075 108.429,167.857L108.183,169.879
106.52,170.307C101.184,171.399 95.659,171.972 90,171.972
50.387,171.972 17.337,143.874 9.693,106.52L9.461,105
10.226,105C16.781,105 43,105 43,105 51.284,105 58,98.284 58,90
58,81.716 51.284,75 43,75 43,75 16.781,75 10.226,75L9.461,75
9.693,73.48C17.337,36.126,50.387,8.028,90,8.028z M90,2C138.601,2
178,41.399 178,90 178,93.038 177.846,96.039
177.546,98.997L176.97,102.772 176.25,101.655 173.587,98.171
174,90C174,43.608 136.392,6 90,6 43.608,6 6,43.608 6,90 6,136.392
```

```
43.608,174 90,174 95.799,174 101.461,173.412
106.929,172.293L108.084,171.996 108.171,174.076 108.593,175.991
107.735,176.212C102.006,177.384 96.075,178 90,178 41.399,178
2,138.601 2,90 2,41.399 41.399,2 90,2z" UseLayoutRounding="False"
Stretch="Fill" Margin="4,0" >
    <Path.Fill>
        <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                   <GradientStop Color="White" Offset="0"/>
                   <GradientStop Color="#4CFFFFFF" Offset="1"/>
        </LinearGradientBrush>
    </Path.Fill>
  </Path>
</Grid>
</UserControl>
```

5. Customize or redraw the Path element into a shape that complements your application. For more information, see Draw a shape or path in your project (http://go.microsoft.com/fwlink/?LinkId=212266).

To add a new menu item that launches a customized Silverlight-based application

1. Open a text editor.
2. Paste the following XML code into the text editor:

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

  <Name _locId="IDName">Shortcut_Name</Name>
  <Description
_locId="IDDescription">Shortcut_Description</Description>
  <Target>\Windows\application.exe</Target>
  <Args></Args>
  <Icon>
    <![CDATA[
            <!--Add XAML Path Element here-->
 ]]>
</Icon>

</XRShellShortcut>
```

3. In the CDATA section in the Icon element, paste the Path element, including all of its attributes and child XAML elements, from the Expression Blend project that you created earlier.
4. In the Name element, replace “Shortcut_Name” with the name of your application.

5. In the Description element, replace “Shortcut_Description” with a description for your application.
6. In the Target element, replace “\Windows\application.exe” with the path to the .exe for your application.
7. Save the file as an .xrsl file in %_WINCEROOT%\Public\Shell\Oak\XRShell\Src. For main menu items, prefix the filename with “Main_”. For settings menu items, prefix the filename with “Setting_”. For example, save it as Main_Application.xrsl.
8. In %_WINCEROOT%\Public\Shell\Oak\Xrshell\Src, open either Main_default.xml for the main menu, or Setting_default.xml for the Settings menu.
9. Add an Item element that references your .xrsl file. For guidelines on which location to add the element in the element tree, see Change the Order of Menu Items.

   Note If the menu item is visible only when a specific variable or SYSGEN is set in the build, put the entry inside a @CESYSGEN conditional statement. Replace BUILD_VARIABLE_TO_SET with the name of the SYSGEN or variable.

   For example:

```
<!--
; @CESYSGEN IF BUILD_VARIABLE_TO_SET
-->
<Item Widget="Main_Item">
        <File>Main_Application.xrsl</File>
    </Item>
<!--
; @CESYSGEN ENDIF
-->
```

10. Open %_WINCEROOT%\Public\Shell\Oak\Files\Shell.bib.
11. In the FILES list of the shell.bib file, add an entry for the .xrsl file after the entries for default .xrsl files such as Main_Photo.xrsl.

    Note If the menu item is visible only when a specific variable or SYSGEN is set in the build, put the entry inside a @CESYSGEN conditional statement. Replace BUILD_VARIABLE_TO_SET with the name of the SYSGEN or variable.

    For example:

```
; @CESYSGEN IF BUILD_VARIABLE_TO_SET



Main_Application.xrsl   $(_FLATRELEASEDIR)\Main_Application.xrsl
NK SH
; @CESYSGEN ENDIF
```

12. Open %_WINCEROOT%\Public\Shell\Oak\Files\Shell.dat.
13. In shell.dat, locate the section titled “Shortcuts for binaries built before shell in _DEPTREES.”

14. Add an entry for the new menu item under ”root:-Directory("\XRShell__Storage\Main").

    Note If the menu item can only appear if a specific variable or SYSGEN is set in the build, put the entry inside a @CESYSGEN conditional statement. Replace BUILD_VARIABLE_TO_SET with the name of the SYSGEN or variable.

    For example:

    ```
    ; @CESYSGEN IF BUILD_VARIABLE_TO_SET
    Directory("\XRShell\__Storage\Main"):-File("Main_Application.xrsl",
    "\windows\Main_Application.xrsl")
    ; @CESYSGEN ENDIF
    ```

15. If Platform Builder is currently attached to a virtual CEPC, on the **Target** menu, click **Detach Device**.
16. Rebuild the run-time image.
    - On the **Build** menu, click **Rebuild Solution**.

Step 6: Localize, Build, and Run the Silverlight-Based Application Launcher

After you have customized the Silverlight-based application launcher, you can configure, build, and run it on your device hardware. This tutorial describes how to run the Silverlight-based application launcher on Virtual CEPC.

You can also create a localized version of the Silverlight-based application launcher. In Windows Embedded Compact 7, the sample home screen provides prelocalized UI string resources for the following languages:

- Chinese (People’s Republic of China)
- Chinese (Taiwan)
- Dutch (Netherlands)
- French (France)
- German (Germany)
- Italian (Italy)
- Japanese (Japan)
- Korean (Korea)
- Portuguese (Brazil)
- Russian (Russia)
- Spanish (Spain)
- Swedish (Sweden)

To configure the OS design for localized builds of the Silverlight-based application launcher (optional)

1. (Optional) Edit UI text resources by customizing the localized string files under the directory for the LCID.

- To modify a menu item in a foreign language for a localized build, follow the steps in Delete Text in a Menu Item and save the .xrsl files in the directory for the LCID at %_WINCEROOT%\public\shell\oak\files\intlfile.
- To add a new menu item in a foreign language for a localized build, follow the steps in Add a New Menu Item and save the .xrsl file in the directory for the LCID at %_WINCEROOT%\public\shell\oak\files\intlfile.

2. On the **Project** menu, click **<OS_Design> Properties**.
3. Expand **Configuration Properties**, and click **Locale**.
4. In **Language Packs to Build**, click the "…" button.
5. Check the box next to the language pack you want to build with, and click **OK**.
6. In **Installed UI Language Packs**, click the "…" button.
7. Check the box next to the language pack you want to include in the run-time image, and click **OK**.
8. In **Default UI Language**, select the default language for the run-time image.
9. In **Default Locale**, select the default locale for the run-time image.
10. Click **OK**.
11. In **Catalog Items View**, browse to <OS_Design>\Core OS\International\Language.
12. Expand the folder for the language for your run-time image.
13. Add additional catalog items for the language you selected by checking the box next to each catalog item. For more information about the catalog items you can include to support a particular locale, see National/Regional Language Support (NLS) Catalog Items and Sysgen Variables in the Windows Embedded Compact 7 Documentation.

To build and run the customized Silverlight-based application launcher

1. In Microsoft Platform Builder, open your OS design project.
2. On the **Build** menu, click **Build Solution**.
3. In the **Output** window, verify that the build succeeded.
4. In the **Device** drop-down menu, select the name of the Virtual CEPC device that you already configured.
5. Start the virtual CEPC that you have preconfigured and connected to Platform Builder, as described in Getting Started with Virtual CEPC (http://go.microsoft.com/fwlink/?LinkId=199788). For example, to start Virtual CEPC on a Windows 7 computer:
   - On the **Start** menu, point to **All Programs**, point to **Windows Virtual PC**, and then click **Windows Virtual PC**.
   - Double-click the virtual CEPC you have preconfigured.
6. On the **Target** menu, click **Attach Device**.

Step 7: (Optional) Create Functionality for Users to Switch the Theme

Theme switching functionality lets a user personalize their device.

Theme switching is an advanced programming task. Instead of selecting one preliminary theme for your Silverlight-based application launcher in Step 1, you can select all four preliminary themes for a display screen size, and then edit all four XAML projects.

Then, create a settings application for users that accesses the registry to switch the theme at run time.

> **Note:** To save registry changes after a device is rebooted, the OS design must support a hive-based registry (SYSGEN_FSREGHIVE), which provides persistent storage.

Determine the Correct Theme Registry Values for Your Code

First, determine which registry values to use in your C++ code.

The registry key for the sample home screen's theme is **HKEY_CURRENT_USER\Software\Microsoft\XRShell\Settings\Theme**. Its value is set during the build process. However, you can change its value in C++ code.

To determine the registry value to use for each XAML project, use the following syntax:

```
<Display_Screen_Size>_Theme<Theme_Number>
```

> **Note** The Default Theme theme number is 0.

Implement Theme Switching

You can implement theme switching to let a user change the theme for the Silverlight-based application launcher on a device.

To create different themes for users

- Follow the steps given in Step 2 of this tutorial to modify the XAML for the four different themes provided for the display screen size that is appropriate for your device.

To create a theme settings application

1. In Platform Builder, open the OS design that you created in Step 1.
2. Create a theme settings application by using Windows Embedded Silverlight Tools. For more information, see A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink/?LinkID=189508).
3. In your settings application, use Expression Blend to design a UI that lets users select a theme name. For an example, see the Personalization control panel in Windows 7.

To write an event handler to change the theme registry value

1. In your settings application, identify a UI element, such as a button, that raises an event that will switch the theme.
2. In **Solution Explorer**, expand the Subproject name, expand **Resource files**, and double-click MainPage.xaml.

3. On the **View** menu, point to **Other Windows**, and then click **Windows Embedded Events**.
4. In **Windows Embedded Events**, click the UI element's name.
5. In the **Name** column, identify the event that you want to handle.
6. In the **Handler** column, double-click the field corresponding to the event.
7. In MainPage.cpp, write code that obtains user input and updates the registry by calling the function RegSetValueEx and RegFlushKey. In your registry code, use the registry values that you identified in Determine the Correct Theme Registry Values for Your Code.
8. Use the MessageBox function to inform the user that the device must be reset in order to apply the new theme settings.
9. Build the application.
   - In Solution Explorer, right-click the subproject, and then click **Build**.
10. To help ensure that updates to registry files are preserved after you reset the target device, adjust the preconfigured connection as follows:
    a. On the **Target Menu**, click **Connectivity Options**.
    b. Select the virtual CEPC device you configured, and then click **Core Service Settings**.
    c. In **Download Image**, select **Only on initial download**.
    d. In **KITL Settings**, clear the **Clear memory on soft reset** box.
    e. Click **Apply**, and then click **Close**.
11. Start the virtual CEPC that you have preconfigured and connected to Platform Builder, as described in Getting Started with Virtual CEPC (http://go.microsoft.com/fwlink/?LinkId=199788). For example, to start the virtual CEPC on a Windows 7 computer:
    - On the **Start** menu, point to **All Programs**, point to **Windows Virtual PC**, and then click **Windows Virtual PC**.
    - Double-click the virtual CEPC you have preconfigured.
12. In Platform Builder, on the **Target** menu, click **Run Programs**. Choose the .exe file for your application, and then click **Run**.
13. After you test your Theme settings application and switch the theme, reset the device.
14. On the **Target** menu, click **Reset**, and then in the dialog box click **Yes**.

To add the theme settings application to the settings menu

1. Add a menu item to the settings menu that launches your settings application. For more information, see Add a New Menu Item.
2. Create a GUID for the settings application. For example, use the Guidgen Tool.
3. Add a registry subkey that has the GUID for the application, and add registry settings for the application. For more information, see Executable Control Panel Registry Settings in Control Panel Registry Settings in the Windows Embedded Compact 7 Documentation.
4. In the .xrsl file for the menu item, add the GUID, enclosed in brackets, to the text content in the Args start and end tags.
5. Build the OS design into a run-time image.

- On the **Build** menu, click **Build Solution**.

Example Code

The following example code shows an event handler for the **Click** event for a button that changes the Theme registry value to Medium_Theme1.

> **Important** For readability, the following code example does not contain security checking or error handling. Do not use the following code in a production environment.

```
#include “XamlRuntime.h”
#include "stdafx.h"
#include "MainPage.h"
#include "App.h"
#include "resource.h"

#define HKCU_Layout L"\\Software\\Microsoft\\XRShell\\Settings"
#define Theme_Key L"Theme"
HRESULT MainPage::_Button1_Click (IXRDependencyObject* pSender,
XRMouseButtonEventArgs* pArgs)
{
    HRESULT hr = E_NOTIMPL;

    if ((NULL == pSender) || (NULL == pArgs))
    {
        hr = E_INVALIDARG;
    }
      HKEY    hThemeKey = NULL;
    DWORD   Type = REG_SZ;
    WCHAR   ThemeValue[MAX_PATH+1] = L"Medium_Theme1";
    DWORD   Length = MAX_PATH;
    long res = 0;


        StringCchLength(ThemeValue, _countof(ThemeValue),
(size_t*)&Length);

        if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_USER, HKCU_Layout,
0, KEY_WRITE, &hThemeKey))
        {
            res = RegSetValueEx(hThemeKey,
                Theme_Key,
                NULL,
                REG_SZ,
                (LPBYTE)ThemeValue,
                (Length+1)*sizeof(WCHAR)
                );
            RegCloseKey(hThemeKey);
            hThemeKey = NULL;
        }

        res = RegFlushKey(HKEY_CURRENT_USER);
        if(ERROR_SUCCESS == RegOpenKeyEx(HKEY_CURRENT_USER, HKCU_Layout,
0, KEY_READ, &hThemeKey))
        {
            res = RegQueryValueEx( hThemeKey,
                Theme_Key,
                NULL,
                &Type,
                (LPBYTE) &ThemeValue,
                &Length
                );
            RegCloseKey(hThemeKey);
            hThemeKey = NULL;
        }


    hr = S_OK;
    return hr;
}
```

For more information about creating event handlers in Silverlight for Windows Embedded, see A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink/?LinkId=189508) and Silverlight for Windows Embedded Developer's Guide (http://go.microsoft.com/fwlink/?LinkId=191039).

The following example code shows one way to display a message in the UI, by using the MessageBox function.

To use this code, you must first create the **IXRVisualHost** for the application by using **IXRApplication::CreateHostFromElementTree** or **IXRApplication::CreateHostFromXaml**, and you must include the Silverlight- based Message Box catalog item (SYSGEN_XAMLMSGBOX) in the OS design. Then, you must copy msgbox.lib from %_WINCEROOT%\Public\Common\Oak\Lib\x86\<*debug_configuration*> to %_WINCEROOT%\OSDesigns\<*OSDesignName*>\<*OSDesignName*>\Wince700\<*Build Configuration*>\cesysgen\oak\lib\x86\<*debug_configuration*>, and from your Theme settings application link to msgbox.lib.

> **Important** For readability, the following code example does not contain security checking or error handling. Do not use the following code in a production environment.

```
#include “XamlRuntime.h”
#include “App.h”

HRESULT MainPage::NotifyUser()
{
    HRESULT hr = E_FAIL;
    HWND hwnd = NULL;
    IXRVisualHost* pHost;

    hr = App::GetVisualHost(&pHost);
```

```
        if(hr == S_OK)
        {
              pHost->GetContainerHWND(&hwnd);
        }

        MessageBox(hwnd, L"To apply the new settings,
        please restart your device.", L"Theme Update", MB_OKCANCEL);

        return S_OK;
    }
```

Conclusion

To create a Silverlight-based application launcher, you can start with the sample home screen that is available in Windows Embedded Compact 7. The sample home screen supports three different display screen sizes and provides four types of themes for each screen resolution. You can create a Silverlight-based application launcher for a Windows Embedded Compact device by modifying the UI design for one of the themes for a sample home screen.

In addition, you can create a custom battery status settings application, and you can implement a UI component so that users can switch the theme.

By starting your development project with a sample home screen, you can reduce the development time required to create a Silverlight-based application launcher for a Windows Embedded Compact powered device.

By using the sample home screen, Silverlight for Windows Embedded, and Microsoft Expression Blend, you can create a professional, refined UI for your Windows Embedded Compact device that launches both Windows Embedded Compact applications and your customized applications, and provides a UI design tailored for the needs of your users.

Additional Resources

- Silverlight for Windows Embedded Developer's Guide (http://go.microsoft.com/fwlink/?LinkId=191039)
- Performance Tuning Guide for Silverlight for Windows Embedded (http://go.microsoft.com/fwlink/?LinkId=205643)
- Editing XAML (http://go.microsoft.com/fwlink/?LinkId=211444)
- Windows Embedded website (http://go.microsoft.com/fwlink/?LinkID=203338)

Copyright