



Add an Existing Window Control to a Silverlight for Windows Embedded Application

Windows Embedded Compact 7 Technical Article

Published: March 2011

Applies To: Windows Embedded Compact 7

Abstract

This document is for Windows Embedded Compact 7 developers who want to add an existing window control from earlier versions of Windows Embedded Compact to a Silverlight for Windows Embedded application.

Step-by-step instructions guide you through the process of adding an existing Win32 window control to an application by using the **IXRWin32Control** class in the Silverlight for Windows Embedded C++ API.

Example code illustrates how to support the Win32 control in Silverlight C++ application code.

Silverlight for Windows Embedded is a native (C++ based) user interface (UI) development framework for Windows Embedded Compact powered devices and is based on Microsoft Silverlight 3.

Introduction

You can reuse a window control from an earlier version of Windows Embedded Compact in an application that you based on Silverlight for Windows Embedded.

By reusing a Win32 control instead of creating a new Silverlight custom user control, you can reduce your development time.

To reuse your Win32 control, you first create either a placeholder **IXRWin32Control** in C++, or a placeholder **<xr:Win32Control>** XAML element. Then, you can write code to swap the placeholder window control with the actual window control by using its window handle.

For more information about window controls in earlier versions of Windows Embedded Compact, see [Working with Window Controls](#) at MSDN.

Add an Existing Window Control to a Silverlight for Windows Embedded Application

To add a window control that you created for an earlier version of Windows Embedded Compact to your Silverlight for Windows Embedded application, first create an **IXRWin32Control**, and then swap the handle of the **IXRWin32Control** with the handle to the window control. You can then configure the sizing and positioning of the **IXRWin32Control**.

Prerequisites

- You have a window control from an earlier version of Windows Embedded Compact that you want to reuse in Silverlight. The window control was created by using **CreateWindow** or **CreateWindowEx**, and the window control has its own **WndProc**.
- The window control is tested and works on Windows Embedded Compact 7.
- You have an OS design that you built into a run-time image and that includes the Silverlight for Windows Embedded catalog item (SYSGEN_XAML_RUNTIME).
- You have a Silverlight for Windows Embedded application. For more information, see "Create a Silverlight for Windows Embedded Application" in [Silverlight for Windows Embedded Developer's Guide](#).

Step 1 Open your Subproject

To add a window control to a Silverlight application, you must first open the Silverlight application in Microsoft Platform Builder.

To open the subproject

1. Open the subproject that you created for the Silverlight application that will reuse your window control.
 - a. On the **File** menu, point to **Open**, and then click **Project/Solution**.
 - b. Browse to the location of your OS design project file.
 - c. Select the project file, and click **Open**.
 - d. In Solution Explorer, expand **Subprojects**.

- e. Expand your subproject folder, expand **Source files**, and double-click the .cpp file that provides your C++ source code.

For more information, see "Create a Silverlight for Windows Embedded Application" in [Silverlight for Windows Embedded Developer's Guide](#).

Step 2 Create an IXRWin32Control Object

You have two options to create the IXRWin32 control object to represent the window control:

- You can define the empty Win32 control in the source XAML.
- If you cannot modify the source XAML, you can create the empty Win32 control in C++.

To define a placeholder Win32 control in XAML

1. In preexisting code, locate the C++ code that creates the window control and find its class name.
2. In Microsoft Visual Studio, open the source XAML file.
3. In the source XAML file, include the XAML namespace declaration for Win32 controls in the root element.

```
xmlns:xr="clr-namespace:EmbeddedXamlRuntime"
```

4. In the source XAML file, define a XAML element that has the **xr** namespace prefix and that specifies the **xr:Name** attribute and on-screen position.

```
<xr:Win32Control xr:Name="Win32CtlSample"
  xr:ClassName="ClassNameSample"
  Height="50" Width="50" Margin="8,8,93,0"
  HorizontalAlignment="Left" VerticalAlignment="Top" />
```

5. (Optional) Specify the class name by adding a value for the **ClassName** attribute.
 - If you specify the **ClassName** attribute in the XAML file and the corresponding **WNDCLASS** structure is already registered in C++, a Win32 window handle will be created after you parse the XAML, and it will contain a pointer to the **WndProc** that you provided with the class name.
 - If you do not specify the **ClassName** attribute in the XAML file, the visual tree will contain only an empty **IXRWin32Control** object that is a placeholder control and that represents an empty UI region in the visual layout. You must then call **IXRWin32Control::SetHandle** to set it to the real handle from another part of your code.
6. (Optional) Define an animation storyboard in the resources that belong to the panel element.

This storyboard animates a property that you define for the Win32 control. The property must be a layout-related property inherited from a parent class, such as **X** or **Height**. In XAML, you specify the property in the **Storyboard.TargetProperty** attached property. For more information about how to define this element in the source XAML for your application, see [Storyboard Class](#) at MSDN.

When the C++ application parses the XAML markup and loads it into an element tree, it also parses the Win32 control XAML element into an **IXRWin32Control** object.

To create an empty, placeholder Win32 control in C++

1. In the existing code, locate the C++ code that creates the window control, and then find the class name for the window control.
2. In Microsoft Platform Builder, open your Silverlight application.
3. In your application code, create an **IXRWin32Control** object, locate the panel object in the visual tree, and then add the **IXRWin32Control** object to the panel.

```
void CreateControl(IXRApplication* pApplication, IXRVisualHost*
pHost)
{
    IXRWin32ControlPtr pIWin32Ctl;
    pApplication->CreateObject(&pIWin32Ctl);

    IXRPanelPtr pPanel;
    IXRUIElementCollectionPtr pChildElements;

    pHost->FindName(L"MyPanel", &pPanel);
    pPanel->GetChildren(&pChildElements);
    pChildElements->Add(pIWin32Ctl, NULL);
}
```

4. In your application code, set the properties on the object you created by using the **Set** methods inherited from **IXRControl**.

You must specify the class name defined for the standard window control in the method **IXRWin32Control::SetClassName**.

```
void SetProperties(IXRWin32Control* pIWin32Ctl)
{
    pIWin32Ctl->SetClassName(L"StandardControlClass");
    pIWin32Ctl->SetName(L"Win32CtlSample");
    pIWin32Ctl->SetHeight(80);
    pIWin32Ctl->SetWidth(200);
}
```

5. Add the new Win32 control to the **IXRUIElementCollection** object of a panel object in the visual tree.

You can do this by obtaining a pointer to the collection from **IXRPanel::GetChildren**.

```
void AddToCollection(IXRStackPanel* pPanel, IXRWin32Control*
pIWin32Control)
{
```

```

        IXRUIElementCollectionPtr pChildCollection;
        pPanel->GetChildren(&pChildCollection);
        pChildCollection->Add(pIWin32Control, NULL);
    }

```

After you create the control, set additional supported properties on the **IXRWin32Control** object.

Step 3 Integrate the Window Control with Your Application

In your Silverlight application, swap the standard window control handle with a Silverlight window control handle, so that your **IXRWin32Control** object is associated with the standard window control.

Note Do not implement event-handling code for the TAB event in the **WndProc** for the Win32 control. When the window control receives focus, the Graphics, Windowing and Events Subsystem (GWES) routes messages and events to the **WndProc** that you created for the standard window control instead of to the default **WndProc** for the visual host window. To give focus back to the visual host window and to navigate away from the window control, the user must press the TAB key or another navigation key.

To integrate a standard window control into your application

1. In your code, find the variable that represents the window handle that the **CreateWindow** or **CreateWindowEx** function returns. For example, `g_hwndDlg`.
2. Initialize Silverlight and parse the source XAML file into an object tree.
3. Obtain a pointer to the new placeholder **IXRWin32Control** object in the object tree, as follows:
 - To locate a placeholder Win32 control that you added in C++, obtain a pointer to the **IXRWin32Control** object that you created in Step 2 Create an **IXRWin32Control** Object.
 - To locate a placeholder Win32 control that you parsed from XAML, call **IXRFrameworkElement::FindName** and supply the value that you specified for **x:Name**. For more information, see Step 2 Create an **IXRWin32Control** Object.

For example:

```

        IXRWin32ControlPtr pIWin32Ctl;
        pRoot->FindName(L"Win32CtlSample", &pIWin32Ctl);

```

4. Swap the default handle to the **IXRWin32Control** instance with the handle to your window control.

You can use the **HWND** variable that the **CreateWindow** or **CreateWindowEx** method returned. To do this, call **IXRWin32Control::SetHandle**.

```

        pIWin32Ctl->SetHandle(g_hwndDlg, false);
        ShowWindow(HostWindow, SW_SHOW);

```

```
UpdateWindow(HostWindow);
```

Your window control is now represented in the visual tree and is integrated with the on-screen layout of peer controls in the window.

Step 4 Build and Run the Application

To run the application that has the window control, you must first build the application in Microsoft Platform Builder.

To build your application

1. In Solution Explorer, expand **Subprojects**.
2. Right-click your subproject, and click **Build**.

In the **Output** window, verify that the build succeeded.

After you build the application, you can run the application on your device hardware by downloading the entire run-time image to the device hardware.

For more information about using a virtual CEPC for your device hardware, see [Getting Started with Virtual CEPC](#).

To run the application on a virtual CEPC

1. Open the Virtual PC console and start a virtual CEPC.
2. In Visual Studio 2008, in the **Device** list, click **VCEPC**.
3. On the **Target** menu, click **Attach Device**.
4. If the **Device Status** window displays the message "Waiting for download request from device", reset the device.
 - On the VCEPC window, on the **Action** menu, click **Reset**.
 - Click **Reset** again.
5. In Visual Studio 2008, on the **Target** menu, click **Target Control**.
6. At the command prompt, type **s <application>**, where <application> is the name of your application.

Example

Description

The following example code defines a Win32 control in a Canvas element in Silverlight 3 XAML.

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.

Code

```
<Canvas
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:xr="clr-namespace:EmbeddedXamlRuntime"
```

```

Width="1000" Height="300">

<Canvas.Resources>
    <Storyboard x:Name="AnimationTarget">

        <DoubleAnimationUsingKeyFrames
            Storyboard.TargetName="Win32ControlTranslateTransform"
            Storyboard.TargetProperty="X"
            Duration="0:0:10">

            <LinearDoubleKeyFrame Value="500" KeyTime="0:0:10" />

        </DoubleAnimationUsingKeyFrames>
    </Storyboard>
</Canvas.Resources>

<xr:Win32Control x:Name="Win32CtlSample"
xr:ClassName="ClassNameSample" Height="50" Width="50" />

</Canvas>

```

Comments

After the C++ application parses this example XAML markup and loads it into an element tree, you can locate the empty Win32 control by calling **IXRFrameworkElement::FindName** and then by using **FindName** to search for "Win32CtlSample".

Example

Description

The following example code shows how to find a placeholder Win32 control in the visual tree and swap its handle for a new one, so that it represents a new window control.

In this example code, the source XAML markup in Win32ctl.xaml predefines the placeholder Win32 control and assigns it the **x:Name** "MyWindowControl".

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.

Code

```
#include <windows.h>
```

```
#include <XamlRuntime.h>
#include <XRPtr.h>

AddWindowControl(IXRApplication* pApplication, HINSTANCE hInstance,
WNDPROC lpSampleWndProc)
{
    XRXamlSource Source;
    Source.SetFile(L"win32ctl.xaml");

    XRWindowCreateParams Params = {WS_POPUP | WS_BORDER, NULL, 0, 200, 200,
300, 300, NULL, NULL, 1, NULL};

    IXRVisualHostPtr pActiveHost;
    IXRFrameworkElementPtr pRootElement;
    IXRWin32ControlPtr pIWin32Ctl;

    pApplication->CreateHostFromXaml(&Source, &Params, &pActiveHost);

    HWND HostWindow = NULL;
    HWND NewHandle = NULL;

    // obtain the container window handle and a pointer
    //to the visual root
    pActiveHost->GetContainerHWND(&HostWindow);
    pActiveHost->GetRootElement(&pRootElement);

    // traverse the visual tree and
    // locate a predefined Win32Control object
    pRootElement->FindName(L"Win32CtlSample", &pIWin32Ctl);

    WNDCLASS WndClass1 = {0};
    LPCTSTR c_ClassName1 = L"Class Name";

    WndClass1.hInstance      = hInstance;
    WndClass1.lpszClassName = c_ClassName1;
    WndClass1.lpfnWndProc    = lpSampleWndProc;
    WndClass1.cbWndExtra     = sizeof(DWORD);
    WndClass1.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);

    RegisterClass(&WndClass1);

    NewHandle = CreateWindow(
```



```
        c_ClassName1,
        L"New Window",
        WS_THICKFRAME | WS_HSCROLL | WS_VSCROLL | WS_OVERLAPPED,
        0, 0, // position will be determined at runtime
            // by the layout system
        0, 0, // size value is not used
        NULL,
        NULL,
        hInstance,
        NULL
    );

    // swap the old handle for the new handle
    pIWin32Ctl->SetHandle(NewHandle, false);

    if (!IsChild(HostWindow, NewHandle))
    {
        goto Error;
    }

    // activate the visual host window and repaint the changed portion
    ShowWindow(HostWindow, SW_SHOW);
    UpdateWindow(HostWindow);

    UINT* exitCode = 1;
    pApplication->StartProcessing(&exitCode);

Error:
    if (pActiveHost)
    {
        pActiveHost->DestroyWindow();
    }

    if (pApplication)
    {
        pApplication->Release();
        pApplication = NULL;
    }

    if (IsXRInitialized)
```

```
{  
    XamlRuntimeUninitialize();  
}  
return 0;  
  
}
```

Comments

To run this example code in an application, you must provide the **HINSTANCE** handle to the application instance from the **WinMain** routine in your application, a pointer to the window procedure (**WNDPROC**) you have created for the Win32 control, and the **IXRApplication** instance.

Conclusion

You can reuse a Win32 control from an earlier version of Windows Embedded Compact in a Silverlight for Windows Embedded application.

To reuse your Win32 control, you first create either a placeholder **IXRWin32Control** in C++, or a placeholder **<xr:Win32Control>** XAML element. Then, you can write code to swap the placeholder window control with the actual window control by using its

Additional Resources

- [Windows Embedded website](http://go.microsoft.com/fwlink/?LinkId=183524) (http://go.microsoft.com/fwlink/?LinkId=183524)
- [Windows Embedded Compact 7 Documentation](http://go.microsoft.com/fwlink/?LinkId=190787) (http://go.microsoft.com/fwlink/?LinkId=190787)

Copyright

This document is provided “as-is.” Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

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

© 2011 Microsoft. All rights reserved.