DIRECTSKIN

A GUIDE TO ADDING DIRECTSKIN TO YOUR APPLICATION

This document explains how to decide which version of DirectSkin you should use and then how to add this to your project.

Deciding on which version your application should use

When adding DirectSkin to your application you need to decide if you are going to use the version which requires your installer to install the ocx into the system32/syswow64 folders, or if you are going to use the version that allows you to have your own unique copy of DirectSkin in your application folder.

For new projects we normally recommend using the version which allows you to have your own unique copy of the ocx as this reduces the risk of other DirectSkin client apps on your machine upgrading the ocx without your knowledge. In the documentation below we will call this the wbocx32 version.

Note: that if you are using this version then on the development machines (and only the development machines), it may be easiest if you also copy the ocx to the system32/syswow64 folders to ensure that the development environment is able to easily locate the ocx. This is because the ocx is pathless and so would typically load from the current application folder.

If you are in complete control of the machines in question and have multiple apps using DirectSkin then having a single copy in system32/syswow64 may be simpler for your needs. In the documentation below we will call this the wbocx version.

Once you have made this decision you need to add the correct ocx to your project.

The ocx will show up as the following depending on the version you are using:

"wbocx ActiveX Control Module" - If using wbocx
"DirectSkin 5 ActiveX Control module" - If using wbocx32

You must select the right version when adding it to your project, although if you change your mind you should be able to easily reimport the right version later depending on the capabilities of your development environment.

Adding the control to Visual Basic 6

Adding the control is easy in VB6. Just open up the project menu and select the components menu item (Ctrl-T) or right click on the toolbox and pick components from that menu.

Note: that on Windows 7 you may need to run VB6 elevated to add ocx controls to projects.

Then simply pick the ocx based on the name of the ocx version you want to use.

Once you have done this there should be a new icon in the toolbox for DirectSkin. **This looks like this:**

To add the ocx to the application just put this control on the first dialog which gets created in your application. You should be aware that when this dialog is destroyed DirectSkin would be disabled.

Once added you can start DirectSkin with the following lines of code:

```
Wbocx1.SetRootPathStr "replace with location of skin folder"
Wbocx1.LoadUIS "skinname\skinname.uis"
Wbocx1.DoWindow 0
```

Congratulations, you have now added DirectSkin to your VB6 application!

Adding the control to Visual Studio 6 (VC6) via a control on a dialog using MFC

Adding the control to Visual Studio 6 using VC6 is simple.

From the Project menu select Add to Project and on the submenu that it opens you should pick Components and Controls.

Then pick Registered ActiveX Controls from the file dialog and locate the ocx using the correct name for the version you want to use. Once added there should be a new icon on the toolbox for DirectSkin.

It will look like this:
To add the ocx to the application just put this control on the first dialog which gets created in your application. You should be aware that when this dialog is destroyed DirectSkin would be disabled.

Once added use the Member Variables tab on the MFC Class Wizard to assign a variable to this control and then make the following calls in your InitDialog handler:

```
m_DirectSkin.SetRootPathStr (“replace with location of skin folder”);
m_DirectSkin.LoadUIS (“skinname\\skinname.uis”);
m_DirectSkin.DoWindow (0);
```

Adding the control to Visual Studio 6 / Visual Studio 2002/2003/2005/2008/2010 (no dialog)

If you want to add DirectSkin to an application which either does not use MFC, or does not have a single dialog that remains during the entire process life then you can add the ocx to the project manually.

Firstly add the header wbocx.h to your project manually. This file can be created by making a test project and following the instructions for adding the control to a dialog or you can find a suitable header included with the sample projects.

If you are not using MFC or are using another development environment you should follow the procedures for creating a header for an ActiveX control for use with your development environment.

Once you have added the header you need to create the control at runtime. We recommend you create this early on in your process startup so that it can be running before you open any windows.

Firstly create a class or global variable of type CWbocx.

Then create the control using the following lines:

```
BSTR bstrLicense = ::SysAllocStringLen(pwchLicenseKey,
sizeof(pwchLicenseKey)/sizeof(WCHAR));
if (!m_DirectSkin.Create (“”, 0, CRECT (0,0,0,0),  this, 0, NULL,
FALSE, bstrLicense))
{
        AfxMessageBox(“Failed to load WindowBlinds skinning
        OCX”);
}
::SysFreeString(bstrLicense);
```

pwchLicenseKey is the structure created by the Microsoft utility Licreqst (see Q151771)

Once done you can now turn on DirectSkin using the following calls:

```
m_DirectSkin.SetRootPathStr (“replace with location of
skin folder”);
m_DirectSkin.LoadUIS (“skinname\\skinname.uis”);
m_DirectSkin.DoWindow (0);
```

Adding the control to a dialog in Visual Studio 2005/2008/2010 – MFC

Unlike earlier versions of Visual Studio, you only need to add the control to the IDE once rather than once per project.

To do this you need to open up a dialog and right click on the toolbox and select Choose Items... from the menu that appears.

After a longish pause this will bring up a dialog with two tabs in VS2005 – later versions have more tabs, but they are still named the same.

You need to select the COM components tab and then find the correct Active X control, tick it and then press “ok” to close the dialog.

You should now have a Wbocx control under General in your toolbox with an icon which looks like this:
The choose toolbox items window in Visual Studio 2005.

To add the ocx to the application just put this control on the first dialog which gets created in your application. You should be aware that when this dialog is destroyed DirectSkin would be disabled.

Once added right click the control on your dialog and add a variable for this control and then make the following calls in your InitDialog handler:

```
m_DirectSkin.SetRootPathStr (“replace with location of skin folder”);
m_DirectSkin.LoadUIS (“skinname\\skinname.uis”);
m_DirectSkin.DoWindow (0);
```

Adding the control to a .NET project (Dialog based)

Unlike earlier versions of Visual Studio, you only need to add the control to the IDE once rather than once per project.

To do this you need to open up a dialog and right click on the toolbox and select Choose Items... from the menu that appears.

After a longish pause this will bring up a dialog with two tabs in VS2005 – later versions have more tabs, but they are still named the same.

You need to select the COM components tab and then find the correct Active X control, tick it and then press ok to close the dialog.

You should now have a Wbocx control under General in your toolbox with an icon which looks like this:

To add the ocx to the application just put this control on the first dialog which gets created in your application. You should be aware that when this dialog is destroyed DirectSkin would be disabled so if this is a problem please follow the instructions for non dialog use.

Once added the control should already have a name (probably axWbocx1).

Use this in your FormLoad handler or InitializeComponent function and add the following (convert the C++ syntax into VB.net, C# etc based on which ever language you are using):

```
axWbocx1->SetRootPathStr (“path to skin”);
axWbocx1->LoadUIS (“skinname\\skinname.uis”);
axWbocx1->SetVersionExpected (107);
axWbocx1->SetVersionExpected (185);
axWbocx1->SetVersionExpected (186);
axWbocx1->SkinAllThreads ();
axWbocx1->UxThemeEmulation (TRUE);
axWbocx1->DoWindow(0);
```

The SkinAllThreads call is recommended because otherwise only the primary thread would be skinned and UxThemeEmulation (TRUE) produces the best results when you have a manifest.

You should also note that your controls may need their FlatStyle property set to System.

Adding the control to a Delphi / C++ Builder project

To add the control to Delphi / C++ builder, use the component menu and the “Import ActiveX control option”.

Find the correct ocx for the version you are using and select it. Then use the install button to add it to your project.

Once added you can use it like any other control and you should make the following calls from DirectSkin:

```
SetRootPathStr (‘path to skin dir’);
LoadUIS (‘SkinName\SkinName.uis’);
SetVersionExpected (119);
DoWindow (0);
```

If the application has a themeaware manifest then calling UxThemeEmulation (TRUE) is recommended. SetVersionExpected (119) tells DirectSkin that the application is using VCL and so it will alter how it handles certain controls to avoid problems.

Adding the control to a .NET application dynamically

There are many cases where you do not want to add the control to a dialog at design time and instead want to construct the Wbocx control at runtime.

The following links describe how to do this in two languages:

C#
VB.NET

The license file is the first line of your .lic file and you must include any trailing spaces.

Example changes to make to C#:

Perform the steps described on that link to create the interop assemblies (using aximp) and then from the project menu pick Add Reference and then browse to the ActiveX Control interop assemblies that you created using aximp.

Example syntax for aximp:

```
aximp /out:C:\temp\AxMyProject.dll "C:\Windows\
System32\wbocx.ocx"
```

Add the following to your class:

```
private static AxWBOCXLib.AxWbocx _directSkin = null;
```

Then create a function to init the ocx and add the following to it:

```
_directSkin = new AxWBOCXLib.AxWbocx();
// set the license key
System.Reflection.FieldInfo f =
typeof(System.Windows.Forms.AxHost).
GetField("licenseKey",
System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance);
f.SetValue(_directSkin, "Put value from wbocx.lic in here");
_directSkin.Name = "DirectSkin";
_directSkin.CreateControl();
_directSkin.Size = new System.Drawing.Size(0, 0);
_directSkin.SetRootPathStr("replace this with the directory
to the skin");
if (_directSkin.LoadUIS("skinname\\skinname.uis") == 0)
{
        _directSkin.SetVersionExpected(107);
        _directSkin.SetVersionExpected(185);
        _directSkin.SetVersionExpected(186);
        _directSkin.UxThemeEmulation(true);
        _directSkin.SkinAllThreads (true);
        _directSkin.DoWindow(0);
        }
```

You should put the license information from the wbocx.lic file in the code along with changing the path for the skin and the skin name.

Special note for .net developers

The default target processor for .net projects created in VS2005/2008 is "Any CPU". If you use this setting then you must remember to include the 64 bit OCX as well as the 32 bit OCX as on a 64 bit system your application will run as a 64 bit process.

If you do not wish this to happen you should ensure you select x86 as the target cpu as this will run on 32 and 64 bit systems as a 32 bit application.

Additionally some .NET controls will only pickup the system colours of the theme and to ensure this is used you should add ForceSystemColours (TRUE) to your DirectSkin calls and this needs to be made before the DoWindow call.

Using other programming languages

If you use another programming language then you should follow the instructions for using ActiveX controls with that language.

Then you should call SetRootPathStr, LoadUIS and DoWindow. You may wish to call UxThemeEmulation (TRUE) and SkinAllThreads (TRUE).

Next Steps

Next you should read the "Performing common tasks using DirectSkin" guide to explain how to do the most common tasks with DirectSkin such as changing skin as well as a brief explanation of all of the key DirectSkin apis.