APPENDIX A

Turning on VMR with a C# Application

The following wrapper edits must be made in the class file that is created with AxImp if your C# application is to gain the benefit of the AxClient VMR mode.

AxImp is the Microsoft provided tool that turns a C++ DLL into an activeX DLL.

To gain this performance benefit, run AxImp with the /source parameter against imsclient.dll (naming the output file aximsclient). Then edit the aximsclient.cs file that is created with the edits listed below. The edited class file must then be used to compile the activeX dll version of imsclient.dll that will be used in your C# application.

The newly created activeX dll will playback video in VMR mode, which offers enormous performance gains by utilizing your video card memory and GPU. It is strongly recommended that all C# applications apply these edits to the wrapper class file in order to get the performance benefit associated with VMR.

Edits are shown below in normal font, while the original class syntax is in italics. Make sure to edit your own copy of the aximsclient.cs file by copying the new sections into it (do not copy the italic sections into your file).

Note The following example is only a code snippet and is not intended to represent a complete compiled client code.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.3603
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------
[assembly: System.Reflection.AssemblyVersion("1.0.0.0")]
[assembly: System.Windows.Forms.AxHost.TypeLibraryTimeStamp("7/20/2010 10:43:28 AM")]
namespace Aximsclient {

//ADDED FOR PROPERTIES
using System;
using System.Reflection;
using System.Security.Permissions;
using System.Windows.Forms;
//END ADD


[System.Windows.Forms.AxHost.ClsidAttribute("{41293422-93fd-443c-b848-e07edbf866c3}")]
[System.ComponentModel.DesignTimeVisibleAttribute(true)]
```

```
[System.ComponentModel.DefaultEvent("OnStopTimeChanged")]
public class axc : System.Windows.Forms.AxHost {

   private imsclient.IMediaPlayerCtrl ocx;
   private axcEventMulticaster eventMulticaster;
   private System.Windows.Forms.AxHost.ConnectionPointCookie cookie;
   public axc() :
      base("41293422-93fd-443c-b848-e07edbf866c3") {
   }

        //ADDED FOR PROPERTIES
        Type propertyBagStreamType;
        ConstructorInfo propertyBagStreamCtor;
        Object propertyBag;
        MethodInfo propertyBagStreamWriteMethod;
        ConstructorInfo stateCtor;
        Object state;
        private void enableVMRMode()
        {
            setOCXProperty("EnableVMRmode", "true");
        }
        private void setOCXProperty(String strName, String strValue)
        {
            try
            {
                ReflectionPermission perms = new
ReflectionPermission(ReflectionPermissionFlag.MemberAccess);
                perms.Assert();

                this.propertyBagStreamType =
this.GetType().BaseType.GetNestedType("PropertyBagStream", BindingFlags.NonPublic);
                if (this.propertyBagStreamType != null)
                {
                    this.propertyBagStreamCtor =
this.propertyBagStreamType.GetConstructor(BindingFlags.Instance | BindingFlags.Public,
null, Type.EmptyTypes, null);
                    if (this.propertyBagStreamCtor != null)
                    {
                        this.propertyBag = this.propertyBagStreamCtor.Invoke(null);
                        if (this.propertyBag != null)
                        {
                            this.propertyBagStreamWriteMethod =
this.propertyBagStreamType.GetMethod("System.Windows.Forms.UnsafeNativeMethods.IPropertyBa
g.Write", BindingFlags.FlattenHierarchy | BindingFlags.Instance | BindingFlags.NonPublic);
                            if (this.propertyBagStreamWriteMethod != null)
                            {
                               this.propertyBagStreamWriteMethod.Invoke(this.propertyBag,
new object[2] { strName, strValue });
                                stateCtor =
typeof(AxHost.State).GetConstructor(BindingFlags.Instance | BindingFlags.NonPublic, null,
new Type[1] { this.propertyBagStreamType }, null);
                                state = stateCtor.Invoke(new object[1] { this.propertyBag
});
                                System.Security.CodeAccessPermission.RevertAssert();
                                this.OcxState = this.state as State;
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                System.Diagnostics.Trace.WriteLine("Error in axc: " + ex.Message);
                // Your stuff here
```

```
            // For example set a flag somewhere that indicates error condition
        }
    }
    //END ADD
```