

A GUIDE TO ACTIVEX AND MICROSOFT CRYPTO API PROGRAMMING

Version 1.2

January 11th, 2001

Daniel J. Sanders (Xetex, Inc.)

© 1999-2001 Xetex, Inc. All Rights Reserved.
Unpublished rights reserved under the copyright laws of the United States.

Xetex, Inc. ("Xetex") makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, Xetex reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such revision or changes.

Table of Contents

INTRODUCTION	3
KEY CONCEPTS	3
DEVELOPMENT PLATFORM	3
Integrated Development Environment: VC++ 6.0.....	3
Visual Studio Service Pack.....	3
Microsoft Platform SDK.....	4
Operating System.....	4
ACTIVEX CREATION	4
Dialog Boxes.....	5
Option 1: Cdialog.....	5
Option2: CdialogAxImpl	5
CRYPTO API	6
Usage	6
DEPLOYMENT	6
CAB and Authenticode.....	6
Scripting Safety.....	7
REGISTRY	7
Testing	7
ActiveX Properties.....	8
SUMMARY.....	8

INTRODUCTION

This document is intended as a brief guide to some of the more important issues involved in creating and deploying ActiveX controls and in using the Microsoft Crypto API ('MS CAPI' or 'Crypto API'). The focus of this guide is on issues that (a) are not easily found in existing documentation or not found in any other documentation, and (b) are crucial to successful ActiveX development.

The information contained in this document was gathered during the development of an ActiveX control. The control contains C and C++ code that uses the Crypto API and the Microsoft Foundation Classes (MFC) to access a user's certificates and private keys to sign text strings. The ActiveX control is downloaded via the Internet and called by other Windows applications, such as Microsoft Internet Explorer, to perform digital signing.

KEY CONCEPTS

Creating ActiveX controls requires familiarity with a number of essential Microsoft and Windows related concepts. They are listed below with brief descriptions:

ATL: ATL is the Active Template Library, a set of template-based C++ classes with which you can easily create small, fast, Component Object Model (COM) objects such as ActiveX controls. ATL is integrated into Microsoft Visual C++ (VC++) in the ATL COM AppWizard to provide automation of ATL object creation. Microsoft's ATL tutorial is at:

http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/visualc/vc/mfc/atl_atl_tutorial.htm

MFC: The Microsoft Foundation Classes (MFC) is a library of C++ classes used by various programming packages on Microsoft Windows. To build a graphical user interface program under MFC (Microsoft Foundation Classes), use the automated tools that are part of the VC++ integrated development environment (IDE).

Crypto API: Crypto API is the Microsoft cryptographic API. It is used to perform cryptographic operations in Windows such as private key signing and certificate management. The Crypto API documentation is at:

http://msdn.microsoft.com/library/psdk/crypto/portalapi_3351.htm

DEVELOPMENT PLATFORM

Integrated Development Environment: VC++ 6.0

Microsoft's Visual C++ (VC++) integrated development environment is designed to partially automate the tasks necessary to create an ActiveX control. VC++ contains a number of 'wizards' (automation tools) that speed up development time, by removing the need to hand-code certain project files. It is therefore highly recommended this tool be used for Windows programming. It is also recommended to use the latest version: VC++ 6.0.

Visual Studio Service Pack

Downloading and installing Service Pack 4.0 for Visual Studio is highly recommended. The Service Pack solves a number of deployment problems relating to DLL timestamps in Visual C++ 6.0. ActiveX controls created on a non-Service Pack

version of Visual Studio are likely to fail deployment on certain operating systems due to timestamp conflicts with system DLLs.

The service pack is available at

<http://msdn.microsoft.com/vstudio/sp/vs6sp4/default.asp>

Microsoft Platform SDK

If you are using Microsoft's Crypto API, you should download the Microsoft Platform SDK from:

<http://msdn.microsoft.com/downloads/c-frame.htm?/downloads/sdks/platform/platform.asp>

It contains the newest versions of the header (".h") and library (".lib") files for Crypto API. It is important to obtain the latest version of these files, as the versions included with VC++ 6.0 do not include a number of Crypto API functions and variables, most notably the CertGetNameString function and the variables it uses to accomplish retrieval of a certificate's identifiers.

Operating System

VC++ can be loaded onto most Windows operating system. It has been used on Win NT and 2000 and it appears to function identically on both platforms.

ACTIVEX CREATION

IDE Path Settings

After downloading the Visual C++ Service Pack you will have two versions of the core Crypto API files: wincrypt.h and crypt32.lib. One pair of these files will be in the SDK directory, the other pair will be in the 'Visual Studio' directory. You should set the IDE path so that the SDK files are in front of the 'Visual Studio' files in the IDE path.

Set the IDE path by selecting from the menu: Tools/Options. Then select the tabbed pane named 'Directories'. The pane will allow you to set the path for Include Files and Library Files as follows:

Set the path for the "Include Files" so that the version of wincrypt.h in the SDK directory is above the wincrypt.h file in the 'Visual Studio' directory. Set the path for Library Files so that the version of crypt32.lib in the SDK directory is above the version of crypt32.lib in the 'Visual Studio' directory.

This should prevent link errors associated with having multiple copies of the Crypto API files on one machine, and you will be using the newer, more comprehensive version of Crypto API.

Setting up a project

An ActiveX control project is set up by choosing New/Project in the VC++ menu. Choose the ATLComAppwizard and check the following options 'DLL format' and 'Support MFC'. This will create an empty ATL project. You can insert an ActiveX control into the project by choosing "Insert" from the main menu, and then "New ATL Object" from the drop-down menu.

There are two types of ActiveX control that you can choose from on this menu, a lite control and a full control. Choose "Lite Control", as it is the type of control designed for html embedding and downloading via the Internet. After pressing NEXT, you will be presented with a 4-pane menu that allows you to configure the control's properties. It is important to make note of the following options:

- **Normalize DC:** This is under the Miscellaneous pane. When selected, the Normalize DC option causes your control to override the ActiveX OnDraw method for its rendering. When not selected, the control overrides the OnDrawAdvanced method. By default, OnDrawAdvanced saves the state of the device context, switches to MM_TEXT mapping mode, calls OnDraw, and then restores the saved device context. Therefore, when you ask for a Normalized DC and do not override OnDrawAdvanced, you introduce a little more overhead.
- **View Status/Misc. Status:** These options, under Miscellaneous, determine the visibility of the control. If your control is intended to be invisible, you should check the "Invisible at runtime" option. The other buttons determine whether the control's behavior will be similar to a button or a label.

After choosing the appropriate options for your control and pressing OK, you will have a new set of files for the ActiveX control inserted into the project.

Dialog Boxes

There are two ways to create dialog boxes for an ActiveX control, both of which rely on the MFC:

Option 1: Cdialog

If you choose 'Insert' from the main menu, 'Resource' from the drop-down menu you can highlight the 'Dialog' option. When you click OK, this will insert a resource file representing a dialog box into your project (under 'Resources'). The resource file will not have any associated cpp or h files. In order to use the resource as a C++ object within the project you will have to create the cpp and h files representing the dialog object. Do this by right-clicking on the resource file and choosing Class Wizard. The Class Wizard will guide you through creating a class derived from the MFC base class Cdialog.

Option2: CdialogAxImpl

If you choose 'Insert' from the main menu, 'Miscellaneous' from the drop-down menu, highlight Dialog and press NEXT, the menu will guide you through the process of creating a class to associate with the Dialog box. After completion, you will have a class that extends the MFC base class CdialogAxImpl and a resource file associated with that class.

Comparison

Creating a dialog that extends Cdialog rather than CdialogAxImpl is probably preferable for most ActiveX controls. A Cdialog class is smaller and easy to configure for initialization and value passing. A CdialogAxImpl class is a fully functional COM component with a COM interface; it is also larger than a Cdialog class. As such, it is probably too complex an object for many uses, especially those that simply requiring a GUI that allows value setting, retrieval and user-interface event capturing.

CRYPTO API

Usage

The Microsoft Crypto API provides C style data structures and functions. As noted earlier, ActiveX controls are written in C++. As a result, using the Crypto API in an ActiveX control requires writing C-style code within C++ object-oriented data structures such as classes. Visual C++ compiles the C and C++ easily without the need for 'Extern C' declarations for the C code.

Function Usage

Many Crypto API functions use the convention of calling the method twice. On the first call to the function a variable representing the required output (e.g. a PKCS7) is set by the programmer to NULL and passed in to the function, along with any other parameters (parameter X in the example below). After this first call, the parameter representing the function output size has been set with a value (parameter Y in the example below). Now, the programmer must use malloc or alloc to allocate a space the size of that parameter. The programmer then calls the function again. This time X is a pointer to an allocated buffer, into which the function output will be placed.

```
CryptoAPIFunction(BYTE* X=NULL, DWORD Y, ...);  
//Y has now been set to the size required for X  
  
X=Malloc(sizeof(Y));  
//Space has been allocated for X  
  
CryptoAPIFunction(X, Y, ...);  
//X now holds the output of the function
```

Figure 1. Example of the Crypto API function calling convention.

DEPLOYMENT

CAB and Authenticode

An ActiveX control must be packaged in a CAB file, in order to facilitate Internet download. CAB is an archiving format similar to JAR or ZIP. After compiling your ActiveX control, you will have a ".DLL" file. Using the Cabarc tool, you can create a CAB file that contains the ".DLL". Cabarc is bundled with the Microsoft Platform SDK.

Authenticode is a mechanism for verifying the source of a piece of software. Authenticode signing is required in order for a user to decide whether to accept an ActiveX control. Internet Explorer will not download unsigned ActiveX controls under default settings. In order to obtain an Authenticode certificate you must contact Verisign or Thawte and pay \$200-\$400 per year for a certificate. Information is available at www.verisign.com and www.thawte.com

Once you have an Authenticode certificate, you can use it to sign your code by using the signcode utility in the Microsoft Platform SDK. Also, see the "Xetex Guide to CAB and Authenticode".

Scripting Safety

An ActiveX control must be marked 'safe for scripting' otherwise it will download correctly but will cause a scripting error on the page. You can make a control safe for scripting by doing one of the following things:

1. Implement the **IObjectSafety** interface on your control.
2. Cause the control to self-register in the registry on the user's machine.

The first approach is much simpler, and requires insertion of only a few lines of code into the control. The main class in the ActiveX project always has the same name as the project itself. For a project named SampleSigner, the main class would be comprised of SampleSigner.h and SampleSigner.cpp and would be named CSampleSigner (The C is the MS naming convention to indicate that this is a class and not an interface). For SampleSigner, the class declaration would start with:

```
class ATL_NO_VTABLE CSampleSigner:  
    public CComObjectRootEx<CComSingleThreadModel>,  
    etc...
```

The following lines list all the interfaces that CSampleSigner implements. At the end of the inheritance list place the following line (all on one line) with the comment line above it:

```
/*Safety Interface implementation*/  
  
public IObjectSafetyImpl<CSampleSigner, INTERFACESAFE_FOR_UNTRUSTED_CALLER |  
INTERFACESAFE_FOR_UNTRUSTED_DATA>
```

This statement is an inheritance of the implementation of the IObjectSafety interface with default parameters.

There is another line that you must add to the file in order to implement IObjectSafety. The SampleSigner.h file discussed above contains a COM map that begins with these lines:

```
BEGIN_COM_MAP(CSampleSigner)  
    COM_INTERFACE_ENTRY(ISampleSigner)...
```

At the end of the COM map, place the following lines:

```
/*IOBJECTSAFETY Implementation*/  
COM_INTERFACE_ENTRY(IObjectSafety)
```

The ActiveX control is now safe for scripting.

REGISTRY

ActiveX controls are registered in the Windows registry and are identified by their class ID, a string that looks something like "CLSID:A4F00E3A-E749-4260-8283-8CD94CD7376E".

Testing

You should test an ActiveX control on a machine other than the development machine. When you compile the ActiveX control, VC++ registers the control on the

local registry. If you test a download scenario on the development machine, you are using a pre-registered object, which is likely to influence test results, and will also bypass the Authenticode prompt during download, because the object is already registered on the machine.

ActiveX Properties

After downloading an ActiveX control to the test machine, you can access it by clicking on Internet Explorer's Tools menu and selecting Options from the drop-down list. When the Internet Options panel pops up, highlight the 'General' pane and click on Settings (in the 'Temporary Internet File' subpanel). This will pop up another window with a "View Objects" button. Pressing this button will show you all the downloaded objects on the local machine, including ActiveX controls. You can view the control's properties or delete the control here. Note that to delete the control that you have just downloaded, you must close and reopen Explorer before performing the operations described above.

SUMMARY

As noted in the Introduction, the issues in this document are some of the 'hidden' issues involved in ActiveX and Crypto API. Combining this document with the resources available from Microsoft (at <http://msdn.microsoft.com>) should lead to a smoother development process for cryptographic ActiveX controls.