Deploying Controls on the Web

With the Visual Basic 5.0 Control Creation Edition, developers now have the ability to create ActiveX controls as easily and quickly as creating form based applications with previous versions of Visual Basic. The benefits of creating these re-usable, industry standard components are numerous, including:

- Construction of applications based on re-usable components.
- Ability to use these components in applications and tools that support ActiveX, including Visual Basic, Internet Explorer, Microsoft Office, FoxPro, Borland Delphi, and so on.
- Ability to create rich Internet- and intranet-based applications by integrating Visual Basic-authored ActiveX controls with HTML pages.

1

Creating ActiveX controls for use in HTML pages opens up a new world of opportunities. The combination of matching the up the rich content capabilities of HTML and application-specific functionality of Visual Basic provides the ability to create solutions not possible before. Additionally, these solutions are more accessible to end users, who can simply navigate to them over the Internet or intranet using their Internet browser.

Contents

2

Note If you plan to distribute your control to other developers who may use the control at design time, be sure to use the traditional options in Setup Wizard, rather than creating an Internet component download.

3

Steps to Preparing Your Control for the Web

There are four steps to take in preparing your control component for use on Web pages:

- Package the component for Internet Component Download.
- Digitally sign the software to be distributed.
- Ensure that the component is safe for scripting and initialization, and mark it as such.
- Arrange to provide for licensing of components that require it.

Internet Component Download

On the Web, the browser is responsible for copying all of the needed files to users' hard disks — and for not copying those they already have. Depending on the browser's security settings, they may not get your control at all.

When you use Setup Wizard to create Internet Component Download, you create a cabinet (.cab) file for your ActiveX control. This .cab file contains all of the information necessary to download, install, and register the components required to run your control on an HTML page. The benefits of this architecture include:

- File compression for faster download.
- A single file for your .ocx and .inf which describes other files required.
- Dependency files, such as Msvbvm50.cab, will only be downloaded as necessary — thus minimizing download time.
- Easier updating when new versions of your component are created.
- Automatic installation performed when the control component is downloaded.

4

For More Information The Setup Wizard step by step later in this document illustrates these concepts.

1

Digital Signing

Code received via the Internet lacks shrink-wrapped packaging to vouch for its reliability, and users are understandably skeptical when they're asked to download it. A *digital signature* provides an opportunity for you to reassure them by creating a path from them to you, should your software harm their system. (Note that digital signing puts your name to your code, but does not ensure that it's hazard-free.)

When you develop software for distribution over the Internet, you work with a third party known as a *certificate authority* (CA) to obtain a *digital certificate,* which will

give users information about you. The CA provides and renews your certificate, authenticates your identity, and handles legal and liability issues for broken security. In addition, the CA typically provides the tools you need to digitally sign your components. Your digital certificate is included with all code you digitally sign and distribute over the Internet.

Important The default setting of Internet Explorer doesn't allow software not digitally signed to be downloaded to the end user's machine. It is very important that you obtain a digital signature for software components you intend to distribute them over the Internet.

ActiveX SDK

The ActiveX SDK, available from Microsoft, includes tools to begin digitally signing your components, as well as test certificates you can use when testing and debugging downloading scenarios. For more information and to download the ActiveX SDK, use your Internet access to visit www.microsoft.com/intdev/sdk/.

For More Information Details on digital signing are available later in this document. The latest on digital signing can be found at www.microsoft.com/intdev/security/misf8-f.htm within the Site Builder Workshop on the Microsoft Web site.

Safety

Unless you design your component so that it's guaranteed to interact safely with script and data passed to it during initialization, a malicious script or data could have harmful results on users' computers — and users would come looking for you when that happens. By default, Internet Explorer will display a warning, and will not download a component that has not been marked safe for scripting and initializing.

Safe for Scripting

On an HTML page, your component's functionality is accessed through scripting, such as when events are handled through VBScript. For ActiveX controls, scripting is the only way to fully utilize the control's features in a browser. So while your control may be a trusted component from a reputable source (you), a malicious script may be able to use its methods to delete files on the user's machine, install macro viruses, and worse. A component is safe for scripting when it can't be scripted to harm the user's computer.

Safe for Initialization

Another potential security hazard is initializing your control's state using untrusted data. On an HTML page, your control's initial state is set from the PARAM attributes that accompany the OBJECT tag in the HTML embedding the control. A component is safe for initializing when its properties can't be passed data that in some way harms the user's computer.

For More Information Details on control safety are available later in this document. Also, guidelines for making your controls safe for scripting and initialization are available in "Designing Controls for Use With HTML" in the document, "Building ActiveX Controls," available with the Control Creation Edition.

4

Licensing

If you wish to make a business of creating ActiveX content for the Internet, you can create controls that require a license for use in an HTML page. A Visual Basic application supplies the run-time license to a control automatically but this process does not happen with a HTML page. The license manager supplies the license to the control from the .lpk file.

Note It's important not to confuse this sort of *run-time* license — in which your control is being used as part of an HTML solution — with the *design-time* license developers receive when they download and run a setup program containing your control, so that they can use the control in their own applications.

5

For More Information Details on the licensing model and the .lpk file is available later in this document, and from the ActiveX SDK.

6

Preparing Software for Internet/intranet Download with Setup Wizard

This section outlines the basic steps for creating a downloadable Windows setup file as a cabinet (.cab) file for a Visual Basic 5.0-created ActiveX control using the Visual Basic 5.0 Application Setup Wizard.

The following Setup Wizard step by step illustrates the process of packaging components for Internet Component Download. In this example, the ActiveX control project is the same project you can create with the step by step procedure in "Creating an ActiveX Control," available with the Control Creation Edition. If you've created your own ActiveX control project, you can easily substitute it and other details for those given here.

Note Before using the Setup Wizard to package your own control component for Internet Component Download, you should become familiar with several important concepts — including component safety and digital signing, covered in detail later in this document.

7

Using Setup Wizard to Prepare a Control for Internet Component Download

1. Click the **Start Menu**, point to **Programs**, point to **Visual Basic 5 Control Creation Edition**, then click **Application Setup Wizard**.
2. If the **Introduction** dialog box appears, click **Next**.
3. On the **Select Project and Options** dialog box, click in the box near the top and enter the location of your Visual Basic project.

   1You can also click **Browse** to find the project.
4. Click **Create Internet Download Setup.**

1 5

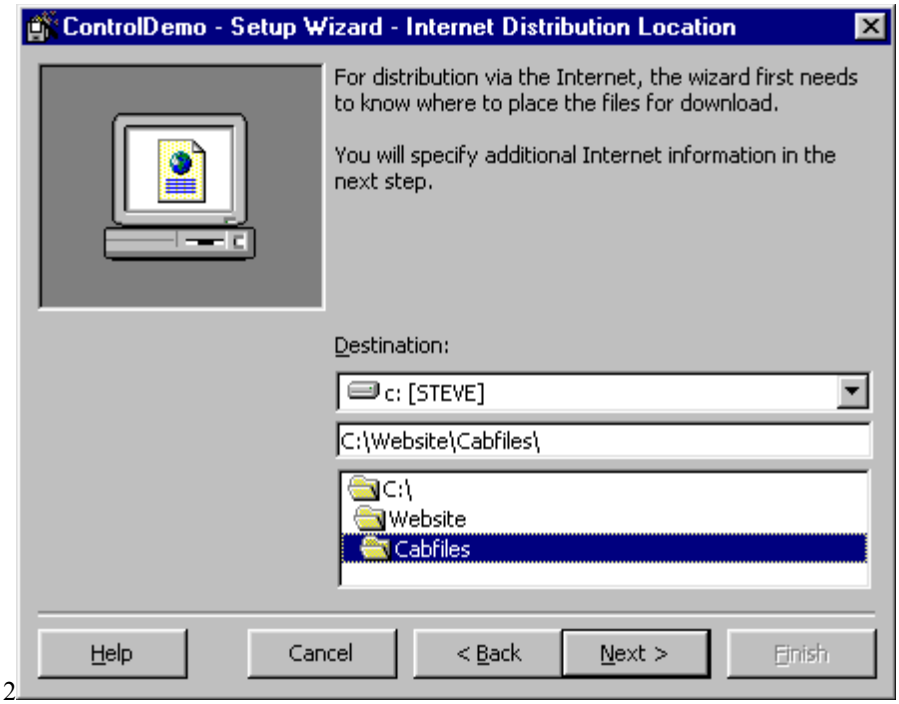
2Before proceeding with the wizard it's a good idea to click **What's New** to check the Microsoft Web site for the latest information. The Internet is a rapidly changing environment that requires developers to keep up to date with current technology and developments.

1**Important** If you are creating a setup program to distribute your control to other developers who may use the control at design time, be sure to click **Create a Setup Program** and use the traditional options in Setup Wizard, rather than creating an Internet component download. A setup program for Internet component download (providing for *run time* use) is

configured very differently from a traditional setup program (which provides for *design time* use).

6

5. Click **Next** to open the **Internet Distribution Location** dialog box.

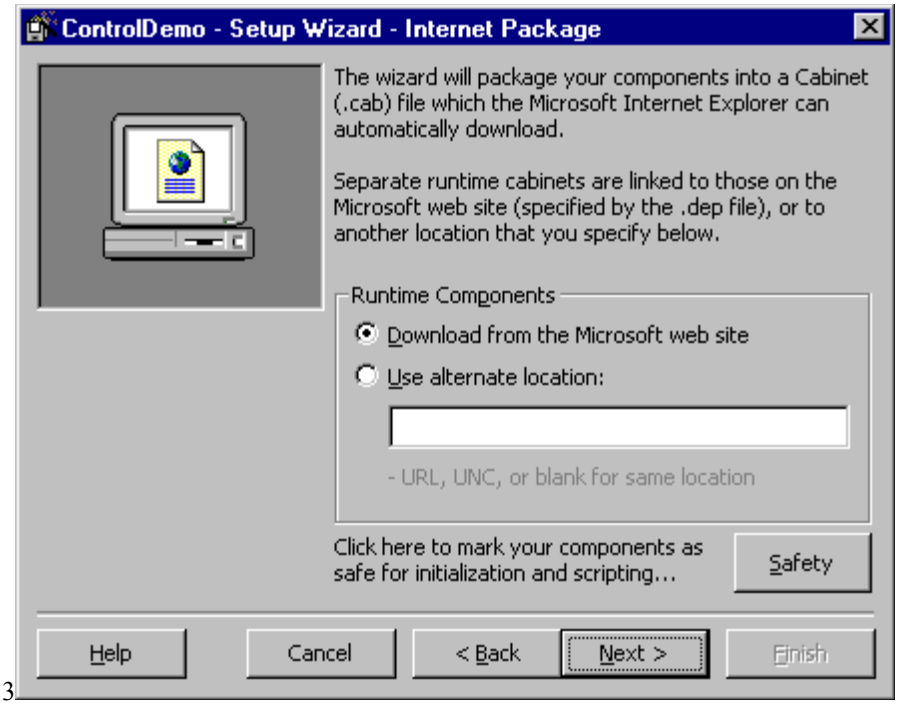


2

7

3Using this dialog box, you can specify where to put the .cab, .htm, and support files the Setup Wizard will create. The .cab file is the software installation file. If you look at your Windows 95 Installation CD-ROM, for example, you will see that Windows 95 software images are stored in .cab files. Placing all of your .cab files in one directory can simplify administration and tracking of the components.

6. Under **Destination**, select the drive to which Setup Wizard will copy the files, then select a path.
7. Click **Next** to open the **Internet Package** dialog box.



3

8

4Using this dialog box, you can choose whether to use Microsoft's Web site as a source for common files, or to supply them from your own server. Unless you are in an intranet environment with no access to the Microsoft Web site, you should accept the default setting. This ensures that your users will get the latest versions of Microsoft-supplied DLLs, and that customized versions of files (say, for a particular operating system) will be sent. The correct version will be sent based on information about its operating system that Internet Explorer reports to the HTTP site.

5Common support files available on the Microsoft Web site and the Visual Basic 5.0 Control Creation Edition include:

- Msvbvm5b.cab — Visual Basic 5.0 VM, required for all Visual Basic 5.0 built applications.
- CmCtlb32.cab — Visual Basic 5.0 Common Controls, including TreeView, ListView, TabStrip, and so on.
- Cmdlgb32.cab — Visual Basic 5.0 Common Dialog Control.

9

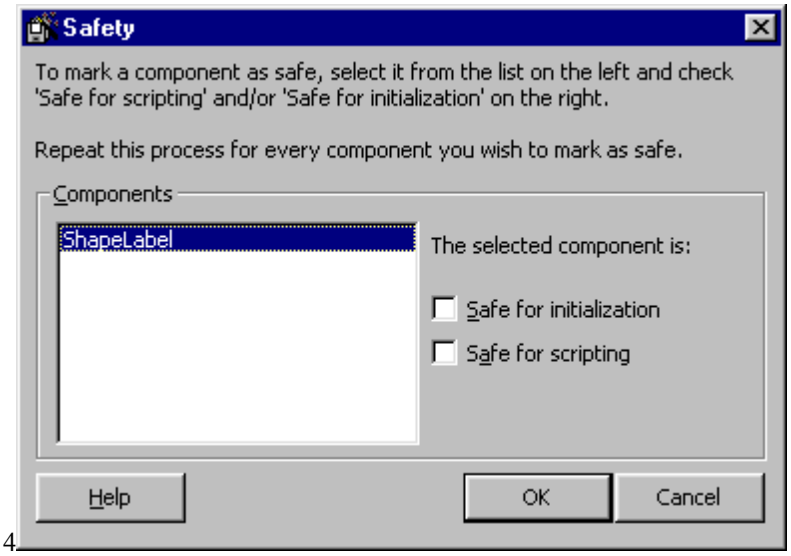
6Each of these files have been digitally signed by Microsoft and may be freely downloaded.

7In an intranet environment you should use a single intranet URL to reduce file replication and allow better administration. Ideally, all developers on your intranet will use the same URL.

2**Note** These .cab files will not be downloaded if the correct version already exists on the end user's machine.

8

8. Click **Safety** to open the **Safety** dialog box.



4

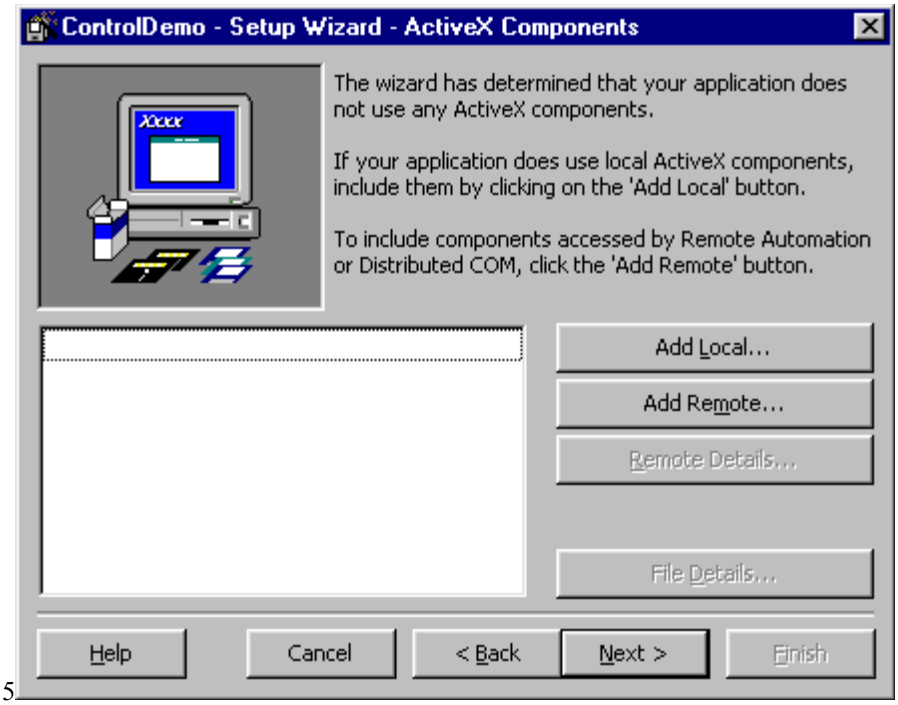
10

8Using this dialog box, you can select **Safe for scripting** and **Safe for initialization** for each control listed under **Components**.

3**Important** Safety is an important issue that will be discussed later in this document. Be sure you have a thorough understanding of safety issues before using these settings with controls you deploy on the Web.

9

9. Click **OK** to close the **Safety** dialog box, then click **Next** to open the **ActiveX Components** dialog box.



5 11

9Using this dialog box, you can specify ActiveX components to include with your control.

10. Click **Next**.

10If your control uses ActiveX components, the **Confirm Dependencies** dialog box will open so that you can verify or change your selections. Since ControlDemo.ocx has no dependencies, the **File Summary** dialog box opens. This dialog box shows what additional files the application uses.

6

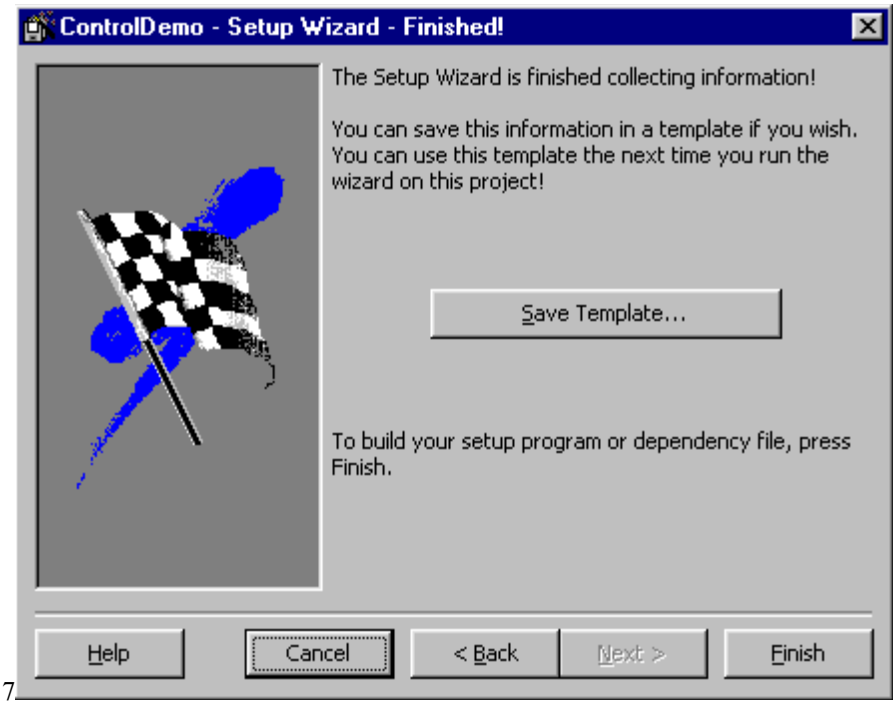
12

11 At this point, it is good to do a bit of double-checking before proceeding to the next step. It can be very difficult to take back or alter things once they are posted to the Internet, so caution is needed. Redistributed files must not be altered in any way, for both legal and practical reasons.

12 **Important** The list of files displayed here doesn't necessarily mean all of these files will be put into your .cab file — it's simply a list of all files required to run your control. For example, Msvbvm50.dll will not be included in the .cab file. If you are unsure about the files displayed, use those Setup Wizard lists by default.

13

11. Click **Next** to open the **Finished!** dialog box.



7

14

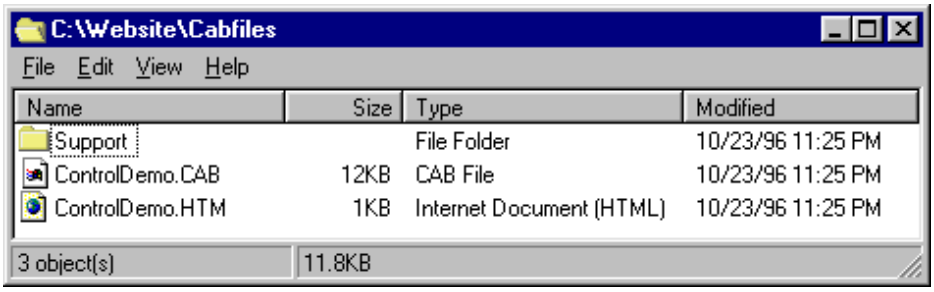
13Here you can save the template for later use.

12.Click **Finish**.

15

Files Created by the Wizard

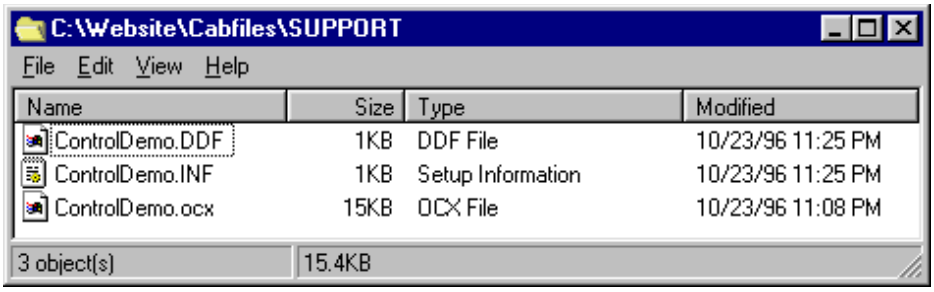
Viewing the target .cab directory shows the files created by the wizard. There are two sets of files: distribution files and support files (for any customization such as adding license information). The distribution files are located in the Cabfiles directory in this example, as shown in the following illustration. The support or intermediary files are not deleted. You may need to rebuild the .cab files with additional files or you may wish to apply a digital signature to the components in the .cab file.



16

The 15KB ActiveX control is now a smaller 12KB .cab file. The Support directory was created by the wizard. The Support directory contains the input files for the .cab and a Diamond Directives (.ddf) file. (Additional information about .cab files is available at www.microsoft.com/workshop/java/CAB-f.htm.)

The contents of the Support directory are shown in the following illustration:



17

All together, the wizard creates five file types, described in the following table:

Extension	Description
.cab	Windows setup file or "cabinet" file that contains the .ocx file, the .inf file and other dependent files. You can digitally sign this file to prevent tampering.
.htm	HyperText Markup File. This file illustrates how to insert the ActiveX control into an HTML page with both Codebase attribute for automatic downloading, and with license files.
.ddf	Diamond Directives File. This is the project file for creating the .cab files.
.inf	Setup disk information file. This file includes information on how the control should be installed. It permits customization of installation.
.ocx	ActiveX control component. You can digitally sign this file to prevent tampering.

10

HTML Object Tag

The following fragment from an .htm file created in the preceding example illustrates what Setup Wizard produces. ActiveX controls are placed in an HTML page by using the Object tag as defined by the World Wide Web Consortium (W3C). The actual download is controlled by the Codebase attribute. (You can obtain more information about the Object tag and its use at http://www.w3c.org/.)

```
<HTML>
<!-- If any of the controls on this page require licensing, you must create a license package file.
     Run LPK_TOOL.EXE in the tools directory to create the required LPK file.

<OBJECT CLASSID="clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
     <PARAM NAME="LPKPath" VALUE="LPKfilename.LPK">
</OBJECT>
-->

<OBJECT
```

```
    classid="clsid:F651BF93-239B-11D0-8908-00A0C90395F4"
    id=ShapeLabel
    codebase="ControlDemo.CAB#version=1,0,0,1">
</OBJECT>
</HTML>
```

11

Examine the second Object tag in the fragment. This tag contains a reference to the WmfView control. It has a Class Identifier (CLSID) of D68CFE2A-139A-11D0-BD7B-00A0D1028E9A. Each different ActiveX control you create will have a different CLSID. The CLSID is used to create an instance of the control on the HTML page (similar to the process of placing a control on a Visual Basic form).

Using this information, Internet Explorer checks the registry to see if the control exists. When it does not exist, or when the version number of the control is less than the version specified in the Codebase attribute of the Object tag, Internet Explorer downloads and installs the file specified in the Codebase attribute.

The important parts of the Object tag are summarized below:

Tag Attributes	Description
Classid	Class Identifier contains the CLSID for the ActiveX control.
Id	Name of the control. Used in scripting, this is equivalent to the Name property of a control on a Visual Basic form.
Codebase	Minimum version number of the control required and the location of an installation point.
Width	The width of the control in pixels.
Height	The height of the control in pixels.

12

For More Information For more information on Internet component download and modifying the download files created by the Setup Wizard see "Advanced Steps of Preparing Software for Internet/Intranet Download" later in this document.

13

Deployment Steps in Depth

Before packaging your control components with the Visual Basic Setup Wizard, it's important to understand the issues involved in deploying controls on the Web. The following topics offers details on the remaining three steps in preparing your component:

- Digital Signing Explained
- Marking Controls as Safe for Scripting and Safe for Initialization
- Licensing of Software on the Internet

18

Digital Signing Explained

Internet Explorer's default security setting requires that any ActiveX component (portable executable, .inf, or .cab) must have a digital signature before that component can be downloaded to the client's machine.

A digital signature identifies the legal entity that created the software (who may be held legally responsible). In other words, it is a claim to the user made under a digital signature for authenticity.

There are four types of files available in Visual Basic 5.0 that may have a digital signature:

- .exe files
- .cab files
- .dll files
- .ocx files

19

Digitally signing controls and components is required for smooth, responsible ActiveX control distribution. Adding a digital signature establishes accountability between your name and certain types of files.

Digital signatures themselves must be purchased from a certificate authority — a company that validates your identity and issues a certificate for you. If a legal action is brought against you as a digital signature holder, the certificate authority becomes an identity witness.

If you cannot obtain a signature for your own use, you may need to arrange to have your control signed by a firm that has a signature. Typically, a third party firm providing a signature would receive your complete source code, review the code, and then sign the component file after recompiling it from the source code.

Note There are different types of signatures, called classes. Some classes of signatures may not be available in some parts of the world for various reasons.

14

Authenticode

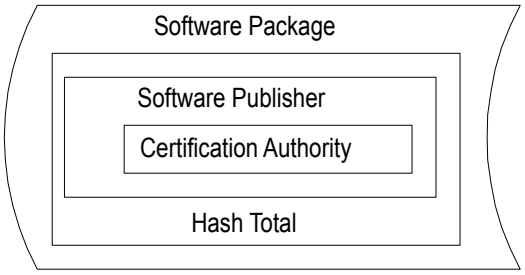
The Authenticode™ technology used in the ActiveX SDK is derived from public-key signature algorithms. The technology uses an algorithm that requires one key to encrypt the data and a second different but complimentary key to decrypt the data. These two keys are called a key pair. For technical details on how this is done, search the Internet for RSA or public-key signatures.

Full encryption or decryption of a large file is very time consuming, so an alternative approach is done with ActiveX controls. In this approach, a number called a *hash* is calculated from all of the bytes in the file (similar to a CRC, or Cyclical Redundancy Check). This hash is encrypted using the private key of the key pair and inserted into the file. When the file is downloaded or installed, a hash is calculated from the file

and compared to the inserted hash. If the numbers match each other, then the file is deemed to be verified.

This process works for verifying the contents but does not identify the source of the software. The certificate authority verifies the identity of the source and issues a certificate that contains the source's name encrypted with the private key of the certificate authority. Internet Explorer has access to the public key so it can determine your identity and the name of the authority supplying the public key.

The following figure illustrates the nesting of one set of data and public keys inside another set. To find out the name of the publisher you must progressively apply the public key obtained from one level to unlock the next layer. For someone to create a bogus certificate, they must obtain the private key on the next higher level. It is required that this information be protected by physical security schemes.



The goal of Authenticode is to establish clear accountability as a deterrent to the distribution of harmful or irresponsible code.

The utility needed to apply a digital signature to your software is not included in Visual Basic 5.0. The Authenticode software utility and documentation are in the ActiveX SDK, available for download from Microsoft's Internet site. The digital signature must be obtained from an issuing authority such as GTE, VeriSign, Inc. (See the ActiveX SDK for more information on obtaining and using signatures.)

For More Information For more information and to download the ActiveX SDK, use your Internet access to visit www.microsoft.com/intdev/sdk/.

The Holder of the Digital Signature

Keeping your digital signature safe is very important. Some firms (including Microsoft) do not keep their signature file on site. The signature is kept with the certificate authority and files are sent there for signing. If you decide to keep your signature on site, then you must ensure that it is strictly controlled. Any file signed with the signature is warranted by your firm — regardless of whether the signature was authorized or not.

Marking Controls as Safe for Scripting and Safe for Initialization

The default security setting of Internet Explorer requires that before any ActiveX component can be instantiated on an HTML page (with specific values or used with a script), it must be marked safe for scripting and safe for initialization. This basically states that that the developer certifies that the control cannot be made to do “harmful” things (such as reformatting your hard drive) via an HTML script or during its initialization.

Safe for Scripting Responsibilities

Marking a control as safe for scripting means that it is safe when automated by any HTML writer or developer. The developer who marks their software as safe for scripting states a warranty: “No matter what the code in VBScript or Java Script is, this control cannot be made to do anything wrong.” The software does not allow any harm to the user. In particular, there should be no possibility of corrupting the user’s PC or obtaining unauthorized information from the PC (for example, no security leaks have been introduced to the system). A control that is safe for scripting is usually safe for initialization, unless there are properties that cannot be set except at initialization.

A control that permits any of the following actions to occur as a result of scripting may not be safe for scripting:

- Create a file with a name specified by scripting.
- Insert information into the registry or .ini files.
- Retrieve information from the registry or .ini files.
- Read a file from the hard drive with a name specified by scripting.

Obtaining any information automatically about the user via an ActiveX control and exposing it to the script writer means the control is not safe for scripting. Such innocent activities may be criminal acts in some countries.

There is a fine line between a safe or unsafe action. For example, a control that always writes information to its own registry entry may be safe. A control that allows you to name the registry entry is unsafe. A control that creates a temporary file with a name it creates without using any initialization or scripting value may be safe. A control that allows the name of the temporary file to be assigned at initialization or by scripting is unsafe.

Determining whether a control is safe is not a trivial exercise. To understand what safety means in terms of what not to implement for your control, you might begin by noting the API calls and commands not implemented in VBScript (see http://www.microsoft.com/vbscript/).

Prior to marketing a control as safe for scripting, it is a good practice to create documentation recording the justification. Remember that you are making an

assertion under a digital signature — you should take the same type of care due any legal contract. One part of your documentation might be two tables listing the following elements:

- All the exposed methods, events, and properties of the ActiveX control.
- All the files opened, API calls used, and the information obtained or set.

If there are any dependencies or data transfer between the elements of these two tables, then the control is probably not safe for scripting. A second part of the documentation may be a review by a seasoned external expert developer who understands both the source code and VBScript.

Safe for Initialization Responsibilities

Marking a control as safe for initialization means that it is safe when initialized by either a well meaning but incompetent HTML writer or a seasoned developer. Safe for Initialization is a weaker safety standard than safe for scripting. The control makes no claim about the safety of its methods or of run-time properties. There are no claims about what information the control makes available to the VBScript writer.

A control marked safe for initialization guarantees to do nothing bad no matter what data it is initialized with. The definition of "bad" is the same as in safe for scripting. The critical issue is what may happen as a result on initialization with particular values. A control that automatically sends information at initialization to an HTTP server may be safe, as long as the server address cannot be changed. You may create files and any other activities needed for the control as long as the file properties (name, file size) or activity's nature does not change because of assigned values.

A safe for initialization control does not write or modify any registry entries, .ini files, or data files as a *result of initialization* parameters.

What the Safety Flags Do Not Mean

In the two prior sections there seem to be long lists of activities that a control could not do. Actually, these are lists of things that the control cannot do as a result of *initialization* or as a result of *scripting*. A control can create, update or delete a file, modify an .ini file, or update a registry entry and still be marked as safe for scripting and safe for initialization. These activities must occur as a result of using the control *regardless of how scripting or initialization is done*.

Safe for scripting or safe for initialization does not necessarily mean that the control is safe for use. A control could reformat your hard-drive after 10 uses, since this action does not occur as a result of scripting or initialization, it may be marked as safe. Of course, the person who writes such a control is liable for the usual penalties reserved for virus and Trojan horse writers.

The following examples illustrate these two ideas:

- A control that is licensed until December 31,1999 or 10 usages on a specific PC may modify a registry entry on each use, and on December 31, 1999, spawn a

batch file to erase itself. This behavior is not the result of either scripting or initialization values.

- A file that creates a temporary file of pointers to memory locations and other working space elements. When the size of a temporary file is significantly dependent on the initialization parameters (for example 10 times the size of a .GIF file pointed to), this behavior may render it as an unsafe control.

23

The key phrase to understand is "as a result of". In the second preceding example, creating a temporary file that corresponds to the file size of a resource passed to the control, may seem a gray area to many developers. If the developer adds code to prevent the temporary file from ever exceeding 10 percent of the available free space on a drive then the control *could* be marked safe.

If, on the other hand, there are no such checks, an HTML writer could consume every free byte available on the drive; the control would be unsafe. In the latter case, the user's PC may become unusable without a reboot.

Responsibility for this does not belong to the end user, who has only one or two megabytes of free space; nor does it belong to the HTML writer, who assigned a three megabyte image file to the control. The responsibility belongs to the developer, who failed to include adequate safeguards.

The bottom line to ensure the safety of your controls is to always have your code independently reviewed by a seasoned developer who understands the issues well.

For More Information Guidelines for making your controls safe for scripting and initialization are available in "Designing Controls for Use With HTML" in the document, "Building ActiveX Controls," available with the Control Creation Edition.

15

Licensing of Software on the Internet

Some controls can require a license for use in an HTML page. A Visual Basic application supplies the run-time license to a control automatically, but this process does not happen with an HTML page. The license manager supplies the license to the control from the .lpk file.

For More Information Detailed information on the licensing model and the .lpk file is available in the ActiveX SDK.

16

Licensing ActiveX controls for use on the Internet is a complex subject. This section explains Microsoft's current licensing scheme, how to implement it with Visual Basic 5.0, and how HTML writers will use it on their pages. A general discussion of some licensing issues is also included. An expression in a license such as "used on one PC" has a totally different meaning on the Internet when the "one PC" may be an HTTP server that is hosting 200 different IP addresses, with 30,000 pages, created by 300 HTML writers. The software may be in-use (running in memory) on thousands of anonymous PCs and never in use on the PC that it was licensed for. These issues

should be reviewed before you start licensing your ActiveX controls for use on the Internet.

Microsoft's Licensing scheme

Visual Basic 5.0 implements a licensing scheme using the **IClassFactory2** mechanism that is implemented by most existing licensed controls. The ActiveX Software Development Kit includes a section entitled "Justification of the Proposed Scheme" that reviews some of the strengths and weakness of this scheme.

Referring back to the HTML file written by the wizard we see the following lines for the License Manager.

```
<!-- If any of the controls on this page require licensing, you must create a license package file.
     Run LPK_TOOL.EXE in the tools directory to create the required LPK file.

<OBJECT CLASSID="clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
    <PARAM NAME="LPKPath" VALUE="LPKfilename.LPK">
</OBJECT>
-->
```

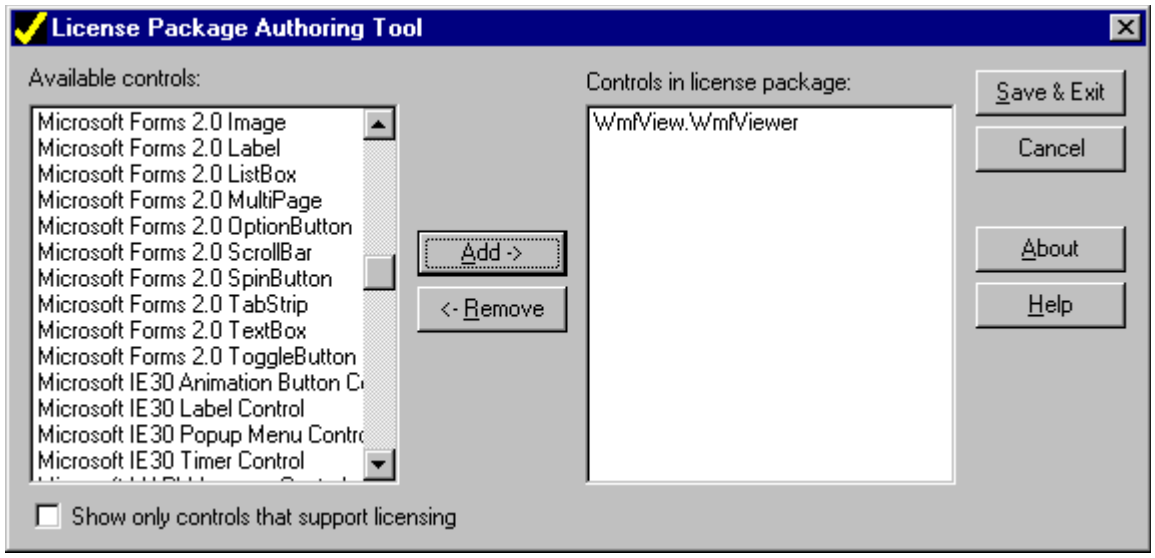
17

The basics of License Manager can be summarized as follows:

- Each HTML page using a licensed control requires a license file (.lpk).
- Only the first encountered .lpk file is used on any HTML page.
- The .lpk file must contain runtime licenses for all of the licensed controls on the HTML page.
- The .lpk file must be on the same server as the HTML page.
- The .lpk file contains a plain text copyright notice to dissuade casual copying of .lpk files.

24

The HTML writer uses the License Package Authoring Tool to create appropriate .lpk files for their pages. This tool presents the HTML writer with a list of installed controls that they can embed in an .lpk file.



25

The .lpk file appears similar to what is shown below:

LPK License Package

//
// WARNING: The information in this file is
// protected by copyright law and international
// treaty provisions. Unauthorized reproduction or
// distribution of this file, or any portion of it,
// may result in severe criminal and civil penalties,
// and will be prosecuted to the maximum extent
// possible under the law. Further, you may not reverse
// engineer, decompile, or disassemble the file.
//
{3d25aba1-caec-11cf-b34a-00aa00a28331}
AQWWF/QT0BG9ewCg0QKOmo=
BQAAAA=yhtrFpw/zxGAdURFU1QAACkAAAB
DAG8AcAB5AHIAaQBnAGgAdAAgACgAYwApACAAMQA5ADkANQAsACAAMQA5ADkANgAg
AE0AYQBjAHIAbwBtAGUAZABpAGEALAAgAEkAbgBjAC4A=

18

Alternative License Models

At least one firm is offering alternative licensing services. This alternative licensing service uses a simple effective license mechanism for the Internet. The control requires a key that varies with time. In the online environment, a call is made to an HTTP server that then examines the URL of the HTML page making the call (and the user's IP location if needed for runtime licenses) and returns a license key if appropriate. This solution does not work well in an intranet environment isolated from the Internet.

There are other licensing mechanisms possible. For example, if run-time licenses are required, then Basic authentication, NT authentication, or other authentication may be required before the license file may be obtained.

Testing Your Internet Component Download

Testing your download file is more complex than testing a conventional setup program because the software installs only if it's missing from the PC or if an older version exists.

Two aspects of component download should be tested:

- Downloading, or copying the files
- Verifying safety levels

26

Also, when there are files listed in the Confirm Dependencies dialog box when you run Setup Wizard, you should test the download on Window 95, Windows NT 3.51, and Windows NT 4.0 to detect any operating system specific problems.

Checking the Installation of Download Components

The simplest level of checking your download is to remove the software registration from the registry. This may be done by running Regsvr32, as in the following example:

```
regsvr32 /u controldemo.ocx
```

27

Thereafter, loading the HTML page generated by the wizard should cause the software to be downloaded and re-installed.

Note Remember to select Binary Compatibility in your Visual Basic Project settings to prevent multiple different CLSIDs for your software.

19

If you have left the DestDir= line in the .inf file blank (the default), you should see the file appear in your Occache directory. You may also find other copies of your component in subdirectories (called conflict directories), such as Occache\Conflict.2 and Occache\Conflict.5. The date/time of each file will be different, indicating different builds of the control (and different CLSIDs).

If, on the other hand, you specify the System directory in the DestDir= line in the .inf file, only the latest version will saved. Care must be taken to ensure that the CLSID remains the same between the builds.

If you want to do heavy duty testing to ensure that all of the support files also install and register, you could un-register everything in your system directory and your

occache directories. This may be done by the following command lines in each directory:

```
for %f in (*.ocx *.dll) do regsvr32 /u /s %f
```

20

When you are finished testing, then the following restores the entries:

```
for %f in (*.ocx *.dll) do regsvr32 /s %f
```

21

Important When making such dramatic changes to the contents of your System directory, it's a good idea keep your setup disks handy.

22

Checking the Safety Levels

You can check safety levels by creating additional HTML pages for testing, as suggested in the following table:

Type of HTML Page	Characterized By
No initialization or scripting	No PARAM values or other variables set by the control.
Initialization with no scripting	PARAM values assigned.
Scripting with no initialization	PARAM values are set by VBScript only.
Scripting and initialization	Initial PARAM values are set and then modified by VBScript.

23

Advanced Customization Through the .inf File

You can customize the installation process by modifying the .inf file. The modified .inf file can be included in a manually built .cab file (using the .ddf project file), or it can be directly referenced by the Codebase attribute of the Object tag.

Note An .inf file is not normally used because it cannot have a digital signature. If an .inf file is used, the .ocx file should have a digital signature.

24

Typical modifications to an .inf file include:

- Adding a license agreement.
- Adding a Read-Me.
- Adding additional documentation.

28

Note As an ActiveX Control developer, you can modify your Internet Component Download, but you should be aware that doing so may place a potential liability on yourself or your firm if the modifications are done incorrectly. Some changes indicate that you guarantee, assure, or warranty that the changes are correct and truthful. Attempts to avoid these liabilities by

citing “as is” or “suitability” clauses in a license agreement may be ruled invalid by many courts.

25

Here is an example of an .inf file:

```
;INF file for ControlDemo.ocx
;DestDir can be 10 for Windows directory, 11 for Windows\System(32) directory, or left blank for the Occache directory.

[version]
signature=$CHICAGO$

[Add.Code]
CONTROLDEMO.OCX=CONTROLDEMO.OCX
MSVBVM50.DLL=MSVBVM50.DLL

[CONTROLDEMO.OCX]
file-win32-x86=thiscab
RegisterServer=yes
clsid={F651BF93-239B-11D0-8908-00A0C90395F4}
DestDir=
FileVersion=1,0,0,1

[MSVBVM50.DLL]
hook=MSVBVM50.cab_Installer
FileVersion=5,0,34,21

[MSVBVM50.cab_Installer]
file-win32-x86=http://activex.microsoft.com/controls/vb5/MSVBVM50.cab
InfFile=MSVBVMb5.inf
```

26

Note Only the original developer may legally mark a control as safe. There are *no* circumstances where you should ever mark someone else’s control as safe. Changing the safety of a control may not only be a copyright infringement but may result in criminal charges.

27

Hand Building .cab Files

If you have modified the .inf file generated by Setup Wizard, you will need to rebuild the .cab file. To rebuild the .cab file you will need Microsoft’s Diamond Cabinet Builder, Diantz.exe. This file is usually located in \VB\SetupKit\kitfil32.

The simplest way is to use the existing .ddf file with the following command:

```
DIANTZ /F CONTROLDEMO.DDF
```

28

If you modify the .inf file in the Support directory and digitally sign the .ocx in the Support directory, be sure to change the locations in the .ddf. You will also need to verify that any files added in the .inf file have also been added to the .ddf file so they will be included in the .cab (and thus available for installation).

Here is an example of a .ddf file. This was created by the Setup Wizard in the step by step procedure in "Preparing Software for Internet/intranet Download with Setup Wizard":

```
.OPTION EXPLICIT
.Set Cabinet=on
.Set Compress=on
.Set MaxDiskSize=CDROM
.Set ReservePerCabinetSize=6144
.Set DiskDirectoryTemplate=
.Set CompressionType=MSZIP
.Set CompressionLevel=7
.Set CompressionMemory=21
.Set CabinetNameTemplate="ControlDemo.CAB"
"C:\Website\Cabfiles\ControlDemo.INF"
"C:\ControlDemo\ControlDemo.ocx"
```

29

The following entries are very significant and should not be changed:

Entry	Description
MaxDiskSize=CDROM	This allows the .cab file to be as large as needed.
ReservePerCabinetSize=6144	This reserves space for a digital signature.

30

The Diantz.exe generates two additional informational files, Setup.inf and Setup.rpt.

If the .cab file is not created, try the following command to get a verbose log of the build process.

```
DIANTZ /V3 /F CONTROLDEMO.DDF
```

31

Please refer to the Microsoft Developer's Network for information on modifying .inf files.

Recommended Readings

For detailed information about .inf files, see the documentation included with Microsoft's Visual C product and on MSDN. The following overview article is available on the Microsoft Developers Network Library.

- Teri Schiele, Windows 95 Application Setup Guidelines for Independent Software Vendors, MSDN Library

29

To obtain further information about this resource, go to http://www.microsoft.com/msdn/

30

Appendix: Check List of Resources

The following resources may be needed to create and test Internet Component Download. Most of the resources are part of the Visual Basic 5.0 package.

Resource	Purpose	Source
SetupWiz.exe	Application Setup Wizard	Visual Basic 5.0
Lpk_Tool.exe	License Pack Tool	Visual Basic 5.0 ActiveX SDK

Diantz.exe	Diamond Cabinet Builder	Visual Basic 5.0
SignCode.exe	Authenticode Signing Software	ActiveX SDK
Authenticode Certificate	Identifies the developer	Verisign Inc., GTE, and so on
Internet Explorer 3.0 or later	Testing of HTML download and installation	http://www.microsoft.com
RegSvr32.exe	Register and un-register software for testing	Visual Basic 5.0

Appendix: Check List for Creating a Professional Internet Component Download

The following check list shows the steps that you might follow to create a fully signed *safe* Internet Component Download Component.

Step	Comments
Design the software specifying the intended software safety level.	Have the design of the control reviewed by a VBScript developer.
Create the software and test.	Features may be added by the developer that inadvertently make the control unsafe.
Create a document showing the object is safe. Independent review.	Most developers are unaware of what their code actually does (they know what it is supposed to do well). Powerful features often create safety problems accidentally.
Set the appropriate Safety Flags and recreate the .cab.	The .inf file is changed with the addition of safe for scripting and safe for initialization entries
All digital signature to files created by you (.ocx, .dll, exe).	Never sign a file that originates from someone else. A digital signature is a statement of *ownership,* not a statement of distribution source or origin.
Create your final .cab file by using Diantz.exe	Do not recreate any file that has been digitally signed or the signature will be lost.
Digitally sign your .cab file.	Both the .cab and the appropriate contents of the .cab are signed.
Test this final .cab with each platform it may run on.	Safety levels must always be checked carefully.