Realtime publishers

The Essentials Series:
Code-Signing Certificates

What's the Process for Using a Certificate?

sponsored by

VeriSign®

by Don Jones

Copyright Statement

What's the Process for Using a Certificate?

If you're convinced that code signing is good for you and your software, then it's time to get a certificate and start signing. Understand that the process differs slightly depending upon the tools and type of software you're working with. For example, mobile devices often have specialized toolsets that must be used to both obtain and use the proper digital certificate. In the next few sections, I'll outline the basic steps as they apply to ordinary computer software.

Obtaining a Certificate

The first step is to obtain a certificate. Specifically, you're looking for a Class 3 Code-Signing Certificate, which is distinct from the many other types of certificates offered.

> **Note**
>
> The class of a certificate conveys a rough sense of how trustworthy the certificate might be. A Class 1 certificate is used for personal email, and while that's important, the identity verification process doesn't have to be as thorough because a compromised email isn't as potentially dangerous to as large a population as, say, a malicious piece of software. You can expect the identity verification process for a Class 3 certificate to be at least somewhat more rigorous, and with some CAs vastly more rigorous, than that of a Class 1 certificate.

Once issued, certificates are encoded with their allowed uses. It's certainly less expensive to obtain a Class 1 certificate, for example, but it won't be encoded with "code signing" as one of its uses—and therefore, your software code-signing tools won't accept the certificate.

You also need to make sure you obtain the correct kind of certificate because different software technologies need the certificate in slightly different formats. Microsoft uses a technology called Authenticode, while Sun's Java uses something slightly different; Microsoft Office and VBA use different certificates from Adobe AIR, which are in turn different from Macromedia Shockwave or Marimba Castanet.

Typically, you'll request your certificate through a Web-based interface. You'll enter key information, such as your contact information, organizational information, and payment information (if you're obtaining the certificate from a commercial Certification Authority—CA). The type of certificate you selected will typically determine the final delivery format.

Installing and Securing a Certificate

Depending on the type of certificate you got, and the operating system (OS) you're running on, you'll have slightly different installation steps. For example, Windows Vista users who obtain an Authenticode code-signing certificate can have the certificate delivered and installed directly from the CA's Web site. Older versions of Windows will usually be given two files: a .PVK file, which contains the certificate's private key, and a .SPC file, which contains the remainder of the certificate. Usually, the .PVK file is provided during the certificate request process, and you get the .SPC file after the CA has completed their identity verification and are ready to issue your certificate. Double-clicking the file will usually begin the installation process, if you want to install the certificate into your local certificate store; you should consult your CA's documentation for complete instructions.

Delivered certificates are usually protected, either by a password, PIN, or some other secret that you set up during the certificate-request process or that is delivered to you separately. During the installation (if you choose to do so), you will often be given an option to protect your certificate with a password. **You should definitely do this.** Failure to do so enables any software running on your computer to digitally sign code *without your knowing,* meaning malware could sign itself using your identity—making you seem responsible for anything malicious the code does.

Once installed, the certificate "lives" in your computer's *certificate store,* usually a protected area of storage where all kinds of certificates are kept. On Windows computers, this store is accessed (somewhat oddly) through the Internet Options Control Panel utility, as Figure 1 shows.

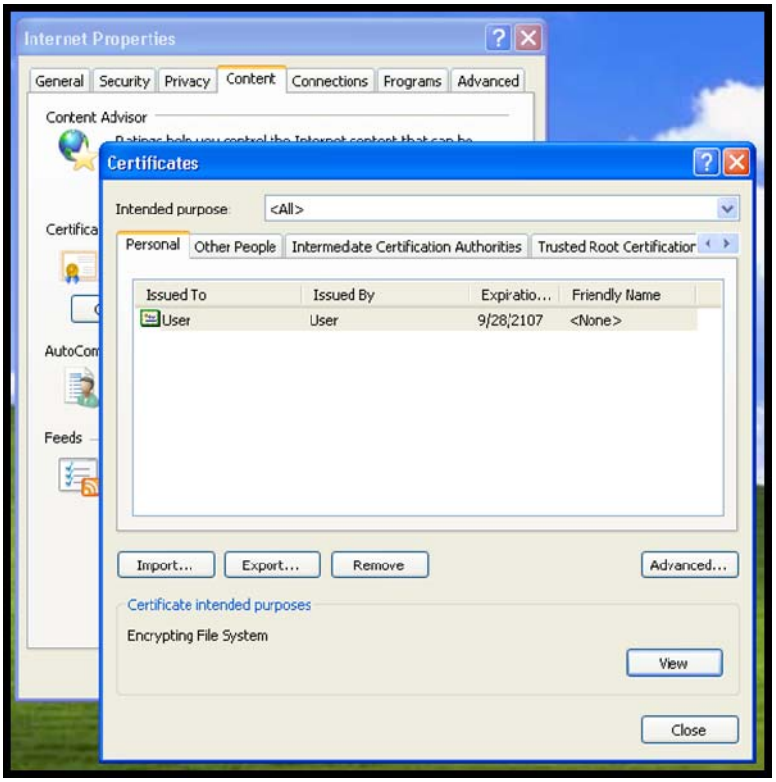

Figure 1: Viewing the certificates on a computer.

Note that Windows doesn't necessarily require a certificate to be installed in order to be used; it depends a bit on the version of Windows and what tool you're using to actually create the signature. In some cases, you might simply need to secure the .PVK and .SPC files—putting them onto an optical disc (CD) or USB key and storing them in a physical safe is a good precaution.

Different technologies, as I've stated, behave differently. For Java, for example, you need to first obtain the Java 2 Software Development Kit (SDK), which contains the Keytool and JarSigner tools that are used to manage certificates. You'll use Keytool to create a certificate store (called a *keystore*). From that, you use Keytool to generate a Certificate Signing Request (CSR), which is copied and pasted into your CA's Web site during the certificate request process. Your CA will deliver your certificate in a text file with a .P7B (or similar) filename extension; you then use Keytool to import that certificate into your keystore.

For Shockwave, Adobe provides an Xtra Packaging Kit. You request your certificate directly from your CA, which delivers the certificate in a file. The Xtra Packaging Kit then imports that certificate and immediately uses it to sign the selected package; there is no certificate store, so you're responsible for keeping track of the certificate file—again, keeping them on CD or USB key and locked in a physical safe until they're needed is a good idea.

As you can see, different software technologies can vary *greatly* in how they require you to enroll and store your certificates; a good CA will provide detailed instructions for the different code-signing technologies that they support.

Using a Certificate to Sign Code

Once you've received your certificate, it's time to use it. If you thought *requesting* a certificate varied across platforms, you can imagine how different the *use* of those certificates is across platforms! For example, versions of Windows prior to Vista relied upon a tool called SignTool.exe, which was included in the freely-available Windows Platform SDK. After installing your certificate, you run **signtool signwizard** to begin a graphical "wizard" process that prompts you for your .SPC file and .PVK file, and then digitally signs the executable you've specified. On Windows Vista, you still use SignTool, although Vista doesn't utilize the .SPC and .PVK files. Instead, you'll have to select a certificate that has been installed on your computer as described earlier.

> **Note**
>
> SignTool.exe also allows you to test the signature of a file—always a good idea after you're done signing it—to make sure that the signature looks correct.

Java developers can sign a JAR file by using the JarSigner utility and can be used to verify the signed result.

The End Results of Signed Code

What matters most to users is that your code *is* signed, and there are a number of visual cues that can alert users to that fact—or let them know that code *isn't* signed. Keep in mind that merely *signing* code isn't sufficient; the user must also trust the CA that issued the signing certificate. In Windows, this list of trusted root CAs can be managed centrally through Group Policy in a domain environment or on individual computers managed through the Internet Options Control Panel application. As Figure 2 illustrates, Windows XP came preconfigured with a large number of trusted root CAs; later versions of Windows trust significantly fewer by default, leaving it up to users and corporate administrators to determine who they trust.

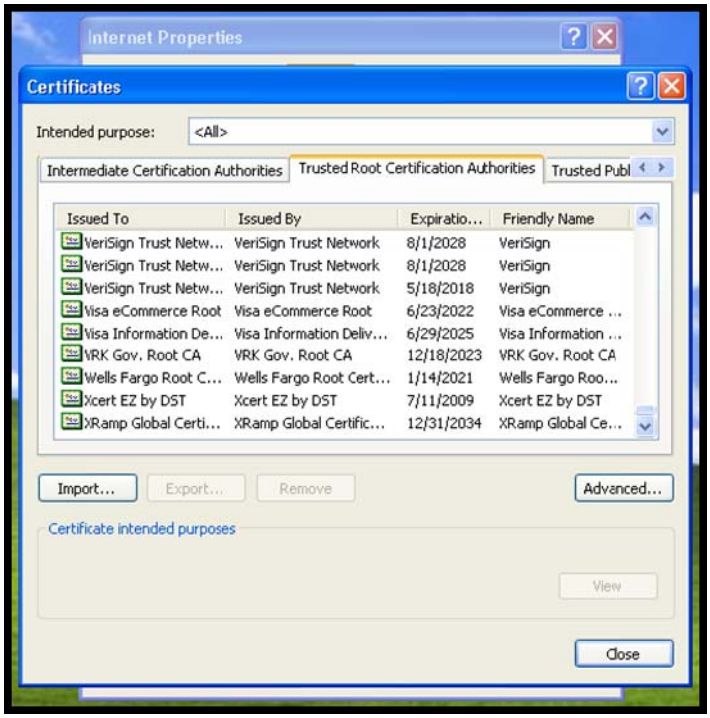

Figure 2: Reviewing the trusted root CAs on a Windows computer.

An application's signature contains the complete path, or *chain,* of CAs who issued the signing certificate—all the way back to the top-level root CA. That top-level root CA must be trusted by the client computer or the entire signature doesn't count.

In Desktop Applications

The first article in this series illustrated one way in which desktop applications show their signatures; users can certainly check for this information on their own if they desire. But users don't need to be able to see an application's signature in order for that signature to affect them; Figure 3 illustrates Windows' SRP, with a new certificate-based rule being created to allow applications that have been signed using a specified certificate. This type of whitelisting allows corporate administrators to more easily control the software that runs in their environment and to rely less on reactive anti-software.

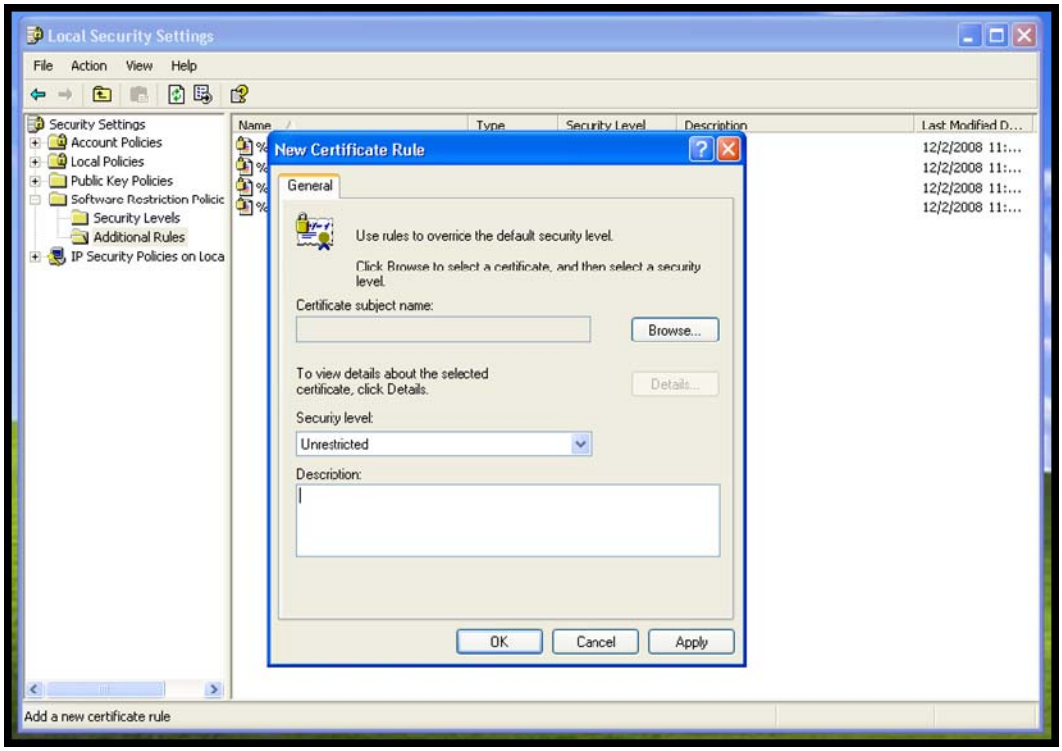

Figure 3: Configuring SRP to allow signed applications.

In Web Applications

Software delivered over the Web has a much more visible impact on users. For example, Figure 4 shows how a user can download software from the Web and review its signature before deciding to actually run it.

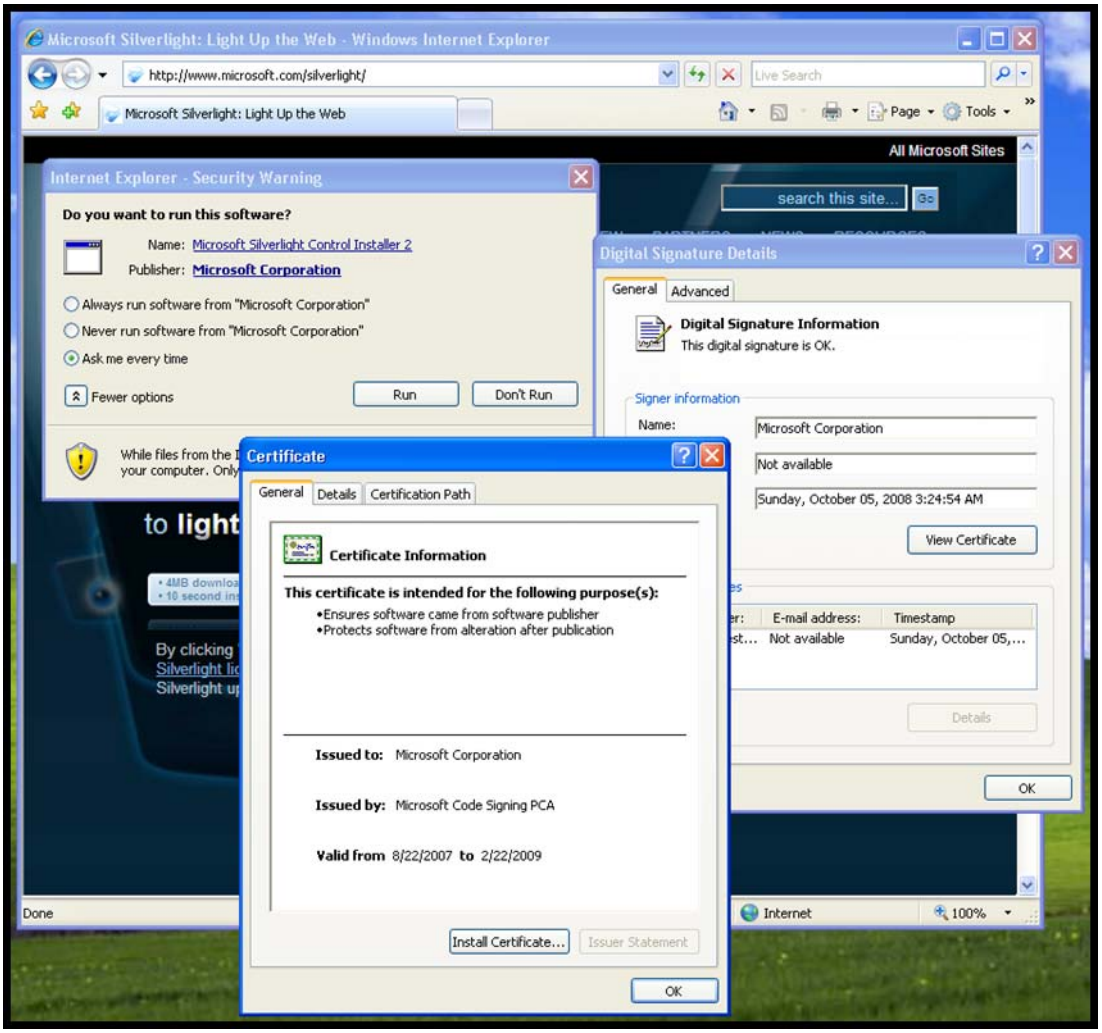

Figure 4: Reviewing a certificate from downloaded software.

In other instances, as illustrated in Figure 5, information from signed code can be used to populate browser-based notifications. In this example, the notification extracts the publisher's identity (Microsoft Corporation) and displays it as part of the notification, helping the user decide whether they trust that publisher.

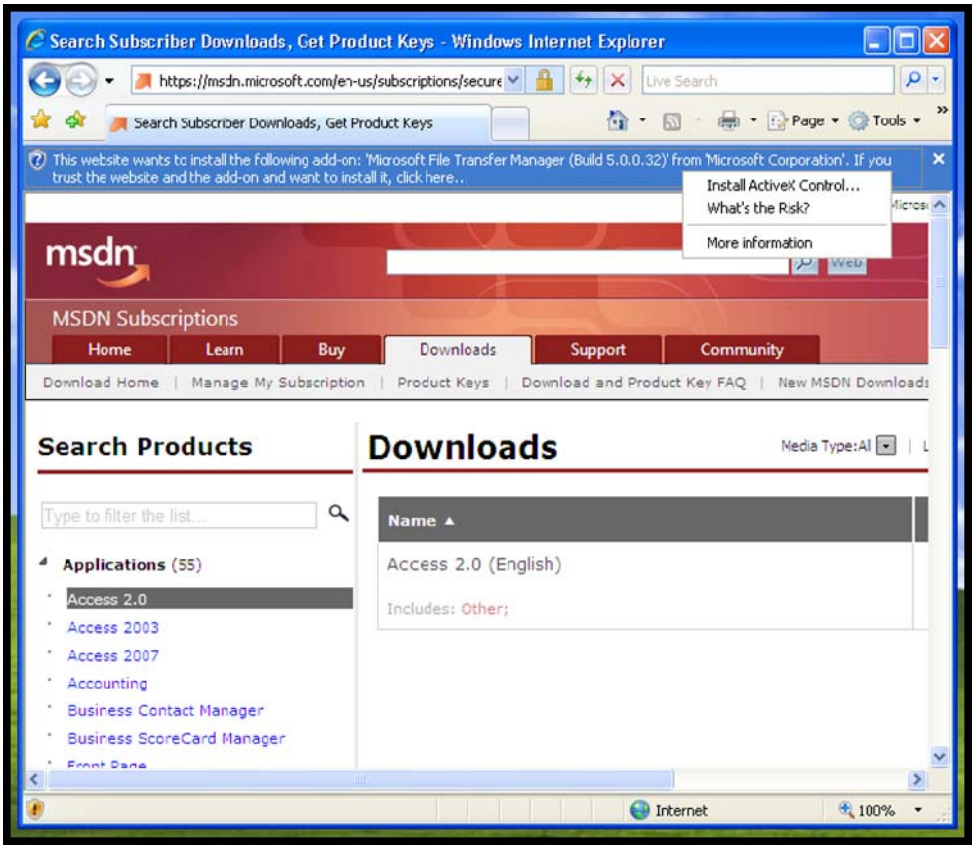

Figure 5: Reviewing browser notices for signed code.

Conclusion

Certificates have come a long way from the public's perception of them as merely for Web server encryption. Today's code-signing certificates are finding more and more uses as users, platform vendors, and software manufacturers seek to implement trust-based relationships with their customers, prevent the spread of malicious software, and protect their assets. Modern software development tools and rigorous commercial CAs have made code-signing certificates easier to use and more trustworthy, and code-signing certificates are contributing more and more to a safer software environment for all of us.