**Exporting a code signing certificate to a PVK, SPC pair**

If your code signing certificate is in a Personal Information Exchange (.pfx) format file, you can use it to sign code using the **signtool.exe** included with Microsoft Visual Studio and the Platform SDK. If you are a shareware developer then you can also use it to sign software PAD files using the Association of Shareware Professionals' **signpad.exe** tool. Tech-Pro CodeSign helps automate the use of these tools, allowing you to digitally sign your software with drag and drop ease.

If you don't own a copy of Visual Studio then to obtain signtool.exe you have to download the [Windows Server 2003 Platform SDK](http://www.microsoft.com/downloads/details.aspx?FamilyId=0BAF2B35-C656-4969-ACE8-E4C0C0716ADB&displaylang=en) or the [Microsoft Windows Software Development Kit for Windows Vista](http://www.microsoft.com/downloads/details.aspx?FamilyID=4377f86d-c913-4b5c-b87e-ef72e5b4e065&displaylang=en). These are very large downloads, just to obtain a 75KB file.

As an alternative, you could convert the .pfx file to a private key (.pvk) and software publishing certificate (.spc) pair, which is usable by Microsoft's original code signing tools. This is quite a complicated process, involving the use of third party tools, but the downloads are much smaller. This article will describe how to carry out the conversion.

**Note:** If your code signing certificate is in the Internet Explorer certificate store, then you must first [export it to a Personal Information Exchange (.pfx) format file](http://www.tech-pro.net/export-to-pfx.html).

**Obtaining the conversion tools**

To export your private key and software publishing certificate from the .pfx file you need the [OpenSSL tools](http://www.openssl.org/). You can download a [ready compiled Windows binary package from Shining Light Productions](http://www.slproweb.com/products/Win32OpenSSL.html). The 'light' package is all you need.

The OpenSSL utility will export the private key to an OpenSSL .pem format file. The .pvk private key format required by the code signing tools is a Microsoft proprietary format which OpenSSL does not support. [Dr. Stephen N Henson, an OpenSSL consultant in the UK](http://www.drh-consultancy.demon.co.uk/), has reverse-engineered the .pvk file format and developed a conversion utility which you can download [here](http://www.drh-consultancy.demon.co.uk/pvk.html). In case you should have trouble accessing the server it is also mirrored [here](http://www.tech-pro.net/files/pvktool.zip). The utility is contained in a Zip archive and you simply need to extract it to the same folder as the OpenSSL tools.

**Exporting the files**

Having downloaded and installed the conversion tools, you are ready to export your code signing certificate and private key file from the .pfx file. The commands given below assume that the location of the conversion tools has been added to the PATH environment variable. Otherwise you should type the full path to each program.

In the examples we will also assume that the .pfx file name is **mycert.pfx** and that the desired output filenames are **mykey.pvk** and **mycert.spc**. You may, of course, substitute other names if you so wish, and specify a full path if the files are located in a different folder. Remember to quote the paths to the files if they contain spaces.

**Exporting the private key**

First export the private key to an OpenSSL .pem format file.

openssl.exe pkcs12 -in *mycert.pfx* -nocerts -nodes -out *mykey.pem*

You will be asked for the password of the private key file, if you specified one.

Now you use Dr. Henson's conversion utility to convert the .pem file into the proprietary Microsoft .pvk format.

pvk.exe -in *mykey.pem* -topvk -strong -out *mykey.pvk*

That takes care of the private key file.

**Exporting the software publishing certificate**

Converting your code signing certificate into a software publishing certificate .spc file is also a two stage process. First, the certificate is exported to an OpenSSL .pem format file, and then this is converted to the final .spc format.

openssl.exe pkcs12 -in *mycert.pfx* -nokeys -out *mycert.pem*

openssl.exe crl2pkcs7 -nocrl -certfile *mycert.pem* -outform DER -out *mycert.spc*

Again, you will be asked for the password of the private key file if you specified one.

You now have a .pvk and .spc pair which you can use to digitally sign executables using **signcode.exe**. The intermediate .pem files created during the conversion are no longer needed, and may be deleted.

**What is Authenticode?**

Authenticode is a technology that uses digital certificates to identify the author or publisher of program code and hashing algorithms to ensure its integrity. It is based on industry standards including Public Key Cryptography Standards (PKCS) #7 (encrypted key specification), PKCS #10 (certificate request formats), X.509 (certificate specification) and the Secure Hash Algorithm (SHA) and Message Digest 5 (MD5) hash algorithms. It is a specific implementation of a type of technology that has been in use for best part of a couple of decades. Another example is [Pretty Good Privacy](http://en.wikipedia.org/wiki/Pretty_Good_Privacy) (PGP) which was developed by Philip Zimmerman in 1991.

Digital signatures are created using a public key signature algorithm. This uses two different keys: a public key and a private key, called a "key pair." The private key is known only to its owner and must be kept secure, while the public key is made available to anyone. Without going into details, the way the algorithm works is that if one key is used for encryption, the other must be used for decryption, as it is impractical to calculate the decryption key from the encryption key. In a digital signature, the private key is used to generate the signature and the public key is used to validate it.

The way the code signing process works is essentially this:

* The code signing software creates a one way hash or message digest of the software.
* It encrypts this hash using the publisher's private key.
* The encrypted hash and the publisher's digital certificate (including the public key) are inserted into the software, signing it.

The signature verification process works like this:

* The user's computer creates a one way hash or message digest of the software.
* It decrypts the encrypted hash using the publisher's public key from the digital certificate.
* It compares the results - if they match then the signature is valid and the software has not been tampered with.

In addition to the publisher's public key, a digital certificate - of which a code signing certificate is just one specific type - contains information about the publisher such as the name, address, contact email address, certificate expiry date and the name and digital signature of the certification authority that issued it. The last three items mentioned are important, because the digital certificate is not only used to verify that a document or program is genuine, but that the publisher really is who the certificate claims it is.

It is possible to create your own Authenticode digital certificate that can be used to verify that your programs have not been modified since you signed them. Microsoft's code signing tools include utilities that enable you to create software publishing certificates. But since you could create a certificate that claimed you were anyone - even Microsoft Corporation - such a certificate would not be trusted by the Windows operating system or web browsers. Therefore you need a code signing certificate issued by a trusted certification authority, whose job it is to check that you are who your certificate will say you are before issuing it. There are four certification authorities that can issue Authenticode code signing certificates - Comodo, Globalsign, Thawte and Verisign. Comodo certificates are the cheapest, and you can [buy Comodo code signing certificates through Tech-Pro.net](http://www.tech-pro.net/code-signing-certificate.html).

**Code signing tools**

As well as a code signing certificate, you need software tools to enable you to sign your programs with it. Microsoft has two versions of these tools. The first was released in 1998 and used to be available from the Microsoft website as the Authenticode for Internet Explorer Toolkit. This version is no longer available from Microsoft's website but the tools are also included as part of the Microsoft .NET Framework Software Development Kit, a somewhat larger download. They may also be downloaded using a link in the free [Tech-Pro CodeSign](http://www.tech-pro.net/codesign.html) tool. They include the following:

* **signcode.exe** - the code signing tool
* **makecert.exe** - creates a digital certificate
* **cert2spc.exe** - converts a digital certificate into the Software Publisher Certificate (code signing) format.

The second version of these tools was released with Microsoft Visual Studio 2005. They are also available as part of the [Windows Server 2003 Platform SDK](http://www.microsoft.com/downloads/details.aspx?FamilyId=0BAF2B35-C656-4969-ACE8-E4C0C0716ADB&displaylang=en) and [Microsoft Windows Software Development Kit for Windows Vista](http://www.microsoft.com/downloads/details.aspx?FamilyID=4377f86d-c913-4b5c-b87e-ef72e5b4e065&displaylang=en) that can be downloaded from the Microsoft website. Again, these are unfortunately rather large downloads. These code signing tools include the following:

* **signtool.exe** - the code signing tool
* **makecert.exe** - creates a digital certificate
* **cert2spc.exe** - converts a digital certificate into the Software Publisher Certificate (code signing) format
* **pvk2pfx.exe** - imports the private key and software publisher certificate into the .pfx file format required by signtool.exe.

The principal difference between these two sets of tools is that signcode.exe requires the private key and software publisher certificate to be in two separate files (.pvk, .spc). The later signtool.exe requires that they are contained in a single Personal Information Exchange (.pfx) file. The .pfx file format is also needed to use the [Association of Shareware Professionals' PAD signing toolkit](http://www.asp-shareware.org/pad/spec/howto_padsign.php).

If you don't have Microsoft Visual Studio but want to import .pvk and .spc files to the .pfx format then you can download the [PVK Digital Certificate Files Importer](http://www.microsoft.com/downloads/details.aspx?FamilyID=F9992C94-B129-46BC-B240-414BDFF679A7&displaylang=EN) from Microsoft's website.

**Obtaining a digital certificate**

Obtaining a digital certificate is fairly straightforward. If you [purchase a Comodo code signing certificate from Tech-Pro.net](http://www.tech-pro.net/code-signing-certificate.html), you will be asked to create an account on Comodo's server. This allows Comodo to keep track of your details, which is useful when you come to renew the certificate.

After paying for your certificate - you can purchase certificates for a period of one, two or three years - you must then wait to be contacted by Comodo using the address you provided. You will be required to provide documentary proof that you are who you will claim to be in your digital certificate. If you are applying on behalf of a company you will be asked to provide a copy of the company's certificate of incorporation or a similar document. If you are applying as an individual you may be asked to supply copies of utility bills or other documents. Full details of the supporting information that may be requested can be found [here](http://www.instantssl.com/ssl-certificate-support/validation_guidelines_table.html).

When your application has been approved you will receive an email inviting you to download your code signing certificate. The exact process will depend on which web browser and operating system you use to order the certificate. Firefox and Internet Explorer under Windows XP will create separate .pvk and .spc files on your hard drive. Internet Explorer 7 under Windows Vista will save your certificate directly in the browser's certificate store, from where you will need to export it to a file.

As explained in the previous section, the different versions of code signing tools require the certificate to be in different formats. Conversely, the format in which you receive the certificate may determine which code signing tools you are able use. It's easy enough to convert a pair of .pvk and .spc files to the .pfx format if you want to use signtool.exe or the ASP PAD signing tools, but converting a .pfx file into a .pvk, .spc pair is somewhat more difficult.

**Security of digital certificates**

One point that is often overlooked is that if the details supplied to the certification authority change during the life of the certificate, you have a responsibility (which you agreed to when purchasing the certificate) to inform the certification authority of those details.

It is equally important to inform the certification authority if your digital certificate is lost or stolen. The certificate may be revoked if you are found not to have complied with this requirement, and you may then have trouble obtaining a new certificate. Because computers (especially laptops) can easily be stolen, some developers keep their digital certificates on a USB thumb drive or similar removable media that is locked away when not being used.

**Creating your own digital certificate**

If you want to try out the process of signing code before you buy a code signing certificate then you can create your own certificate for test purposes.

Open a command prompt in the folder containing the code signing tools, and type:

makecert.exe -sv mykey.pvk -n "CN=Acme Software Inc." mycert.cer

You can substitute your own business name for "Acme Software Inc." If the file mykey.pvk does not already exist, you will be prompted to enter a password for the private key file. The password may be blank. If you will be signing your code using these command line tools it is more convenient if it is blank. However, if anyone should get hold of your unprotected private key file they will be able to sign software as if they were you.

On completion of this command you should have two files, **mykey.pvk** and **mycert.cer**. Now you need to convert the digital certificate into the Software Publisher Certificate (.spc) format. To do this, type:

cert2spc.exe mycert.cer mycert.spc

You will be prompted to enter the password for the private key file. (Hopefully, you haven't forgotten it already!) When the program finishes you should have a new file, **mycert.spc**. Only the two files **mykey.pvk** and **mycert.spc** will be used when signing your code.

**Signing your software**

Signing your software is a simple process, but the process is different depending on which version of the Microsoft code signing tools you are using. In the examples given below, we will assume that your private key is stored in a file **mykey.pvk** and that your code signing certificate is named **mycert.spc** and that both these files are in the same folder as the code signing tools themselves. If that isn't the case, just change the paths accordingly, and put quotes round the paths if they contain spaces.

**Using signcode.exe**

If you are using the signcode.exe tool from the Authenticode for Internet Explorer Toolkit or the Microsoft .NET Framework Software Development Kit, open a command prompt and type:

signcode.exe -t <timestamp URL> -spc mycert.spc -v mykey.pvk "<file to be signed>"

where:

* **<file to be signed>** is the path of the .exe, .dll, .ocx or other executable code that you wish to sign;
* **<timestamp URL>** is the URL of a time stamping server. This may be one of the following:  
  - http://timestamp.verisign.com/scripts/timstamp.dll  
  - http://timestamp.globalsign.com/scripts/timstamp.dll  
  - http://timestamp.comodoca.com/authenticode

It is important that executable code is time stamped to show when it was signed. If it is not time stamped, then when users try to run your software after the certificate you signed it with has expired, they will receive a warning message about it.

**Using signtool.exe**

If you are using the signtool.exe from Microsoft Visual Studio 2005 or later or the Platform SDK, then you must first import your private key and software publisher certificate into a single PFX file. This is a one-off process that need only be repeated whenever you renew your code signing certificate. Open a command prompt and type:

pvk2pfx.exe -pvk mykey.pvk -pi <password> -spc mycert.spc -pfx mycert.pfx -po <password>

replacing <password> with your private key password. If you used a blank password when you created your private key file then you can omit the -pi <password> but we found that it is necessary to specify a password for the output file. When we didn't, it appeared that pvk2pfx created a password of its own, and we received a password error when trying to sign code with the resulting PFX file.

If you downloaded the the [PVK Digital Certificate Files Importer](http://www.microsoft.com/downloads/details.aspx?FamilyID=F9992C94-B129-46BC-B240-414BDFF679A7&displaylang=EN) then you can create a PFX file using that instead. This tool does not have a problem creating PFX files with a blank bassword. To run it, type:

pvkimprt.exe -pfx mycert.spc mycert.pvk

This will open a Certificate Import Wizard. Be sure to check "Yes, export the private key" under "Export Private Key" when this page of the wizard appears.

At the end of this process you should have a PFX file. Now you are ready to start signing programs with it. To do this, open a command prompt and type:

signtool.exe sign /f mycert.pfx /p <password> /t <timestamp URL> /v "<file to be signed>"

replacing <password> with the password specified when you created the PFX file (omit /p if this was blank) and <timestamp URL> with one of the time stamping server URLs listed above.

http://www.tech-pro.net/export-to-pvk-spc.html  
http://www.tech-pro.net/code-signing-for-developers.html