P7S Signer User Manual

Introduction

The main function of P7S Signer is to sign any kind of documents using X.509 digital certificates. Using this product you can quickly sign multiple files (bulk sign) by selecting input and output directory. This is ideal for bulk signing of a large number of corporate documents rather than signing each one individually.

Links

P7S Signer main page: http://www.signfiles.com/p7s-signer/
Download P7S Signer (Free 30-Day Trial): http://www.signfiles.com/apps/P7SSigner.msi

Warning and Disclaimer

Every effort has been made to make this manual as complete and accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this manual.

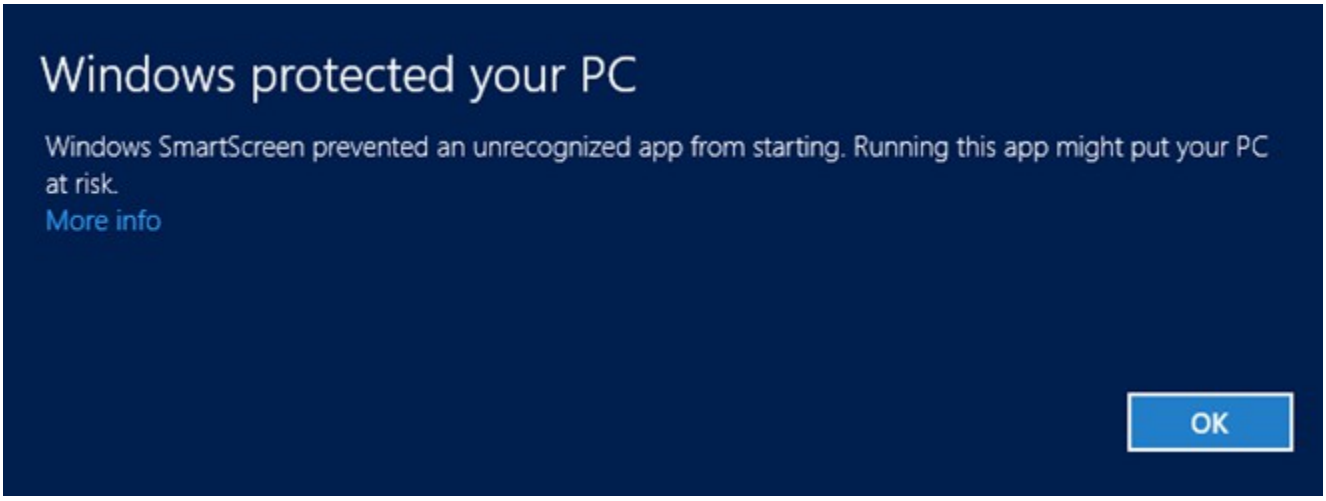
Trademarks

.NET, Visual Studio .NET are trademarks of Microsoft Inc.
All other trademarks are the property of their respective owners.

Product Installation

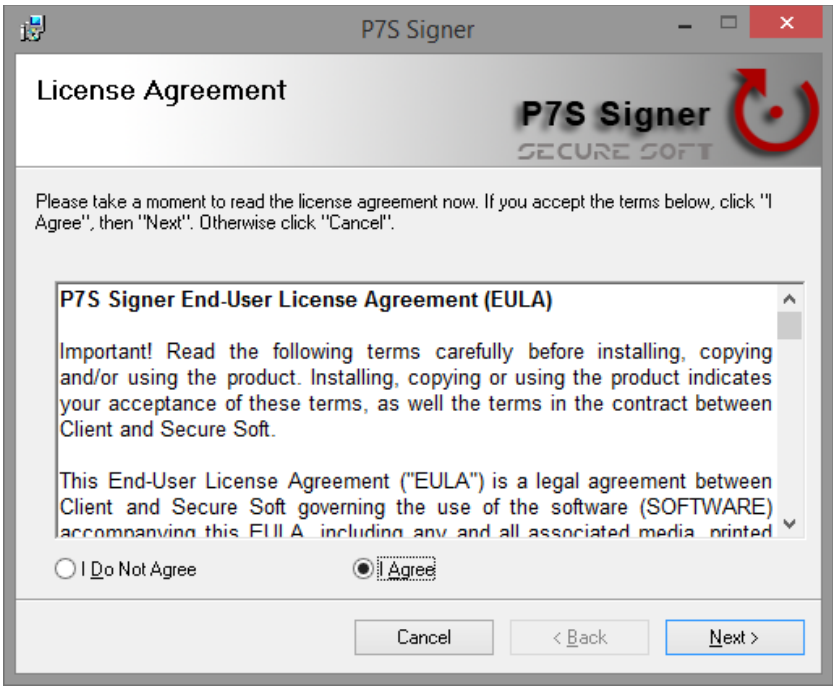
We recommend to install the product using an Administrator account.

After the setup file is verified, the operating system might request your permission to install this program.



Click More info and next click *Run anyway*.

Read the Eula and if you want to continue, select *I Agree* and click *Next* button until the setup is finished.



Digital Certificates

Digital Certificate Location

To use P7S Signer software, a digital certificate is needed. The digital certificates are stored in two places:

- in Microsoft Store
- in PFX on P12 files

The certificates stored on **Microsoft Store** are available by opening *Internet Explorer* – *Tools* menu – *Internet Options* – *Content* tab – *Certificates* button (see below).

To create digital signatures, the certificates stored on *Personal* tab are used. These certificates have a public and a private key.

The digital signature is created by using the private key of the certificate. The private key can be stored on the file system (imported PFX files), on a cryptographic smart card (like Aladdin eToken or SafeNet iKey) or on a HSM (Hardware Security Module).

Signing certificates available on Microsoft Store

Another way to store a digital certificate is a **PFX (or P12) file**. This file contain the public and the private key of the certificate. This file is protected by a password in order to keep safe the key pair.

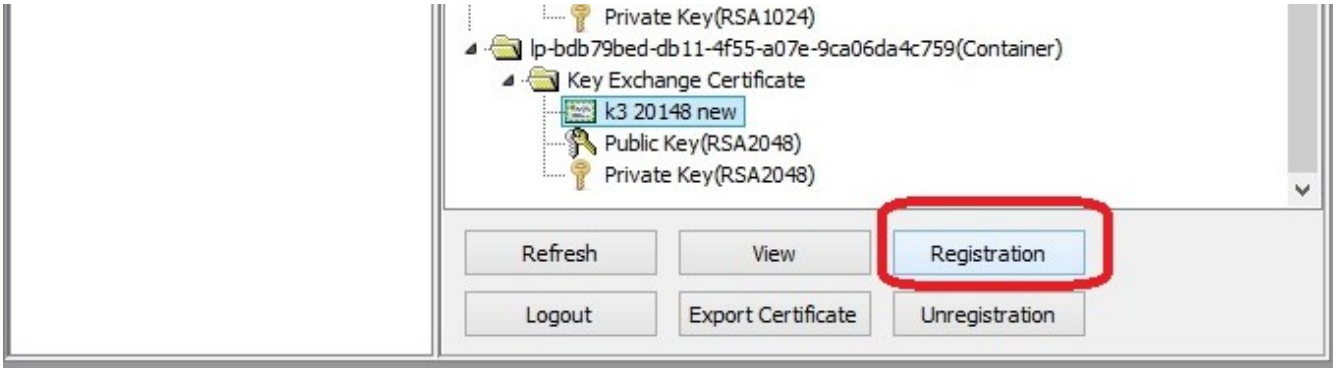
Note that a PFX/P12 file can be imported on Microsoft Store (just open the PFX/P12 file and follow the wizard).

Certificates Stored on Smart Cards or USB Tokens

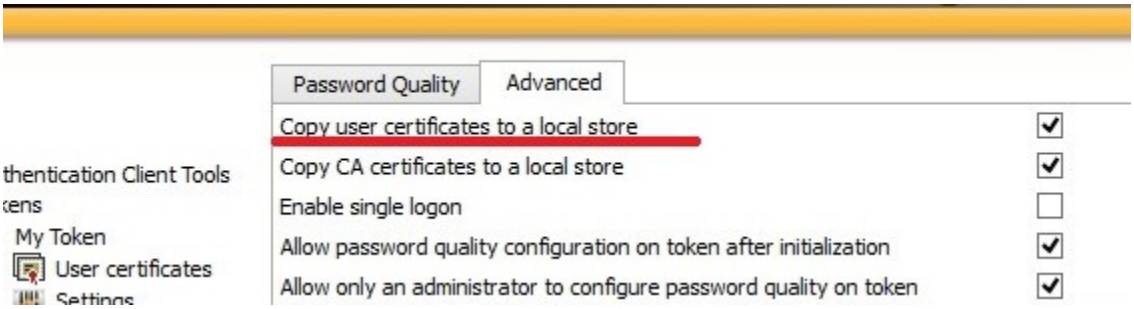
If your certificate is stored on a smart card or USB token (like Aladdin eToken), the certificate must appear on Microsoft Certificate Store in order to be used by the product.

If the certificate not appears on Microsoft Store, you must ask your vendor about how to import the certificate on the MS Store. Usually, the smart card driver or the middleware automatically install the certificate on Microsoft Certificate Store.

You should also look at the middleware options, like below:



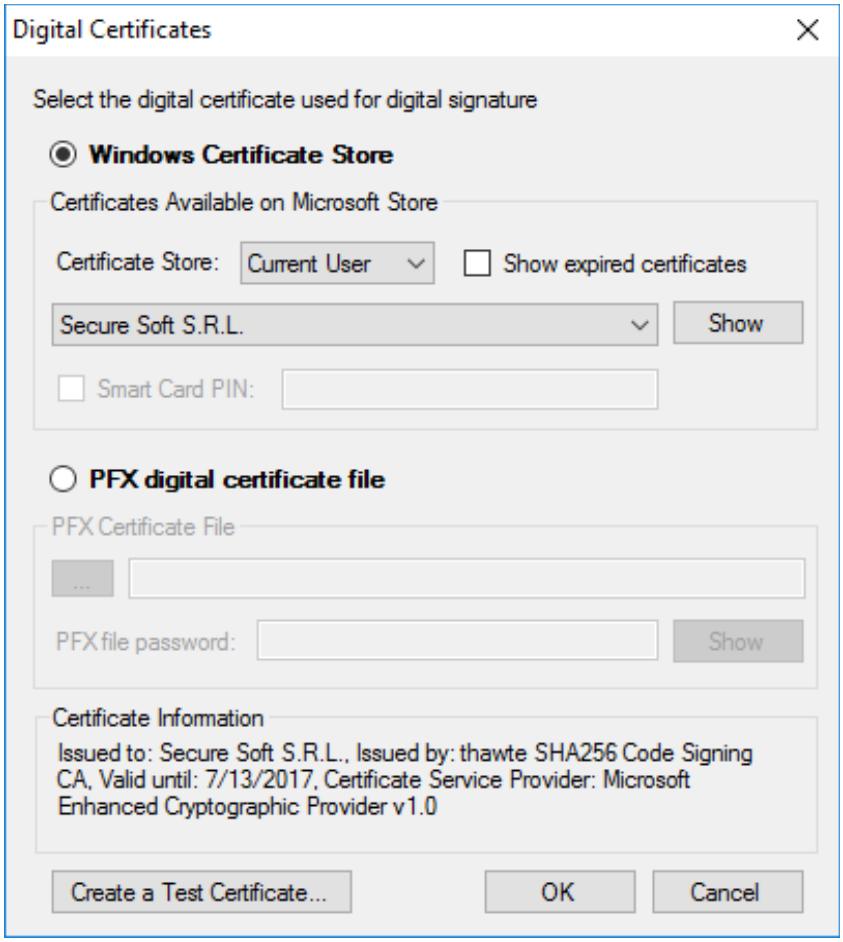
Adding the certificate on Microsoft Certificate Store



Adding the certificate on Microsoft Certificate Store

Select the Digital Certificate for Creating Signatures

To digitally sign a document, a digital certificate must be selected from Digital Certificates section. The digital certificate used to create the digital signature can be stored on Microsoft Store or a PFX file.

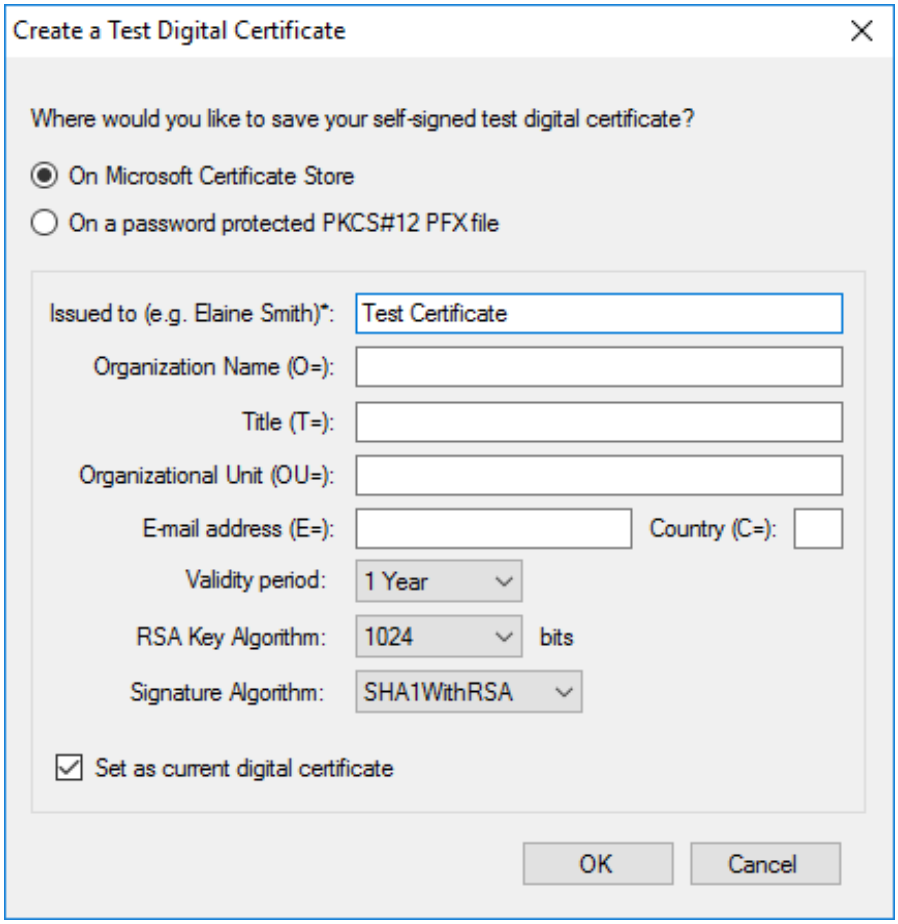


Select the digital certificate

Create a Test Digital Certificate

If no certificates are available on the computer, a test certificate can be created from *Create a Digital Certificate* section.

This certificate can be set as the default digital certificate used for creating signatures.



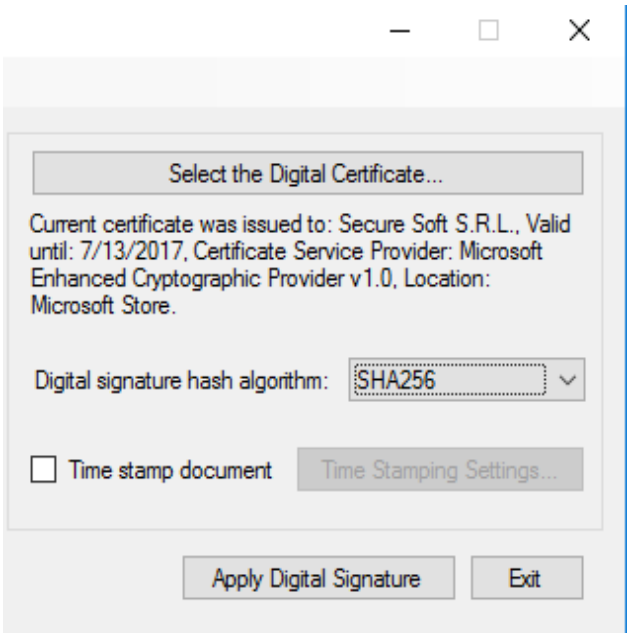
Create a digital certificate

Digital Signature Options

Using SHA256, SHA512 Hash Algorithms

The default hash algorithm used by the product is **SHA1** but in some cases, SHA256 or 512 must be used for the digital signature.

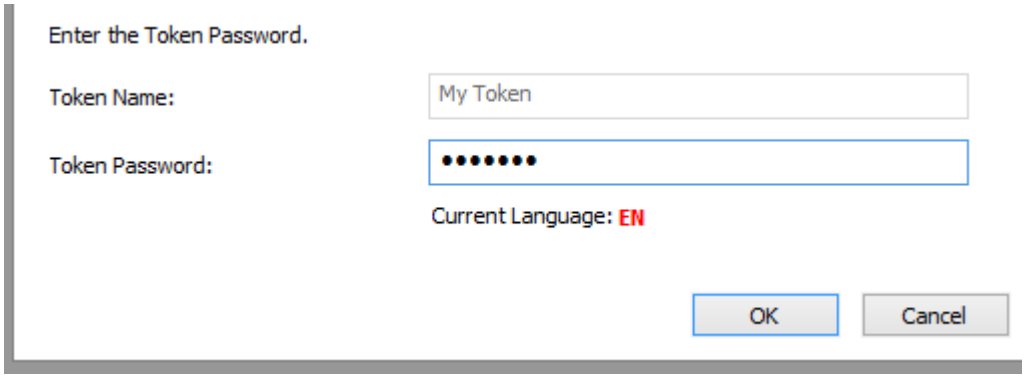
Attention: SHA-256 and SHA-512 hash algorithms are not supported by Windows XP. Note that some smart cards and USB tokens not support SHA-256 and SHA-512 hash algorithms.



Set the Hash Algorithm

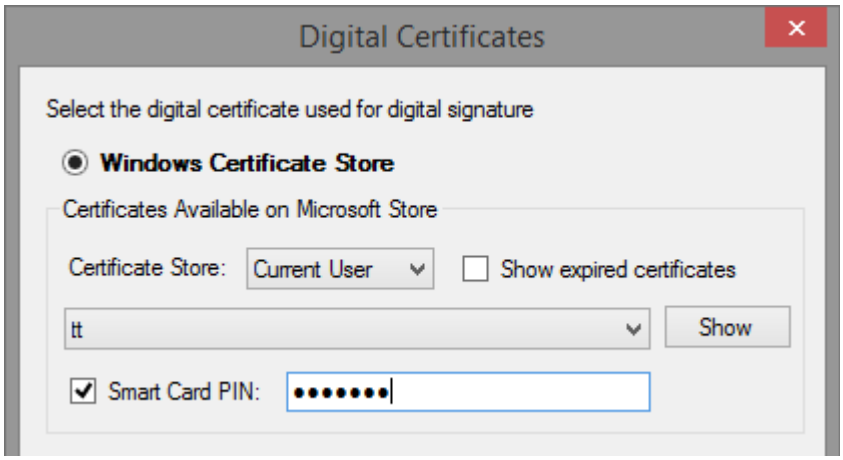
Bypassing the Smart Card PIN

In case the digital signature must be made without user intervention and the certificate is stored on a smart card or USB token, the PIN dialog might be automatically bypassed for some models.



PIN dialog can be bypassed

In order to bypass the PIN dialog window, the Smart Card PIN checkbox must be checked and the right PIN to be entered.DigitalCertificate.SmartCardPin propery must be set. This option bypass the PIN dialog and the file is automatically signed without any user intervention.



Bypassing the Smart Card PIN

Attention: This feature will NOT work for all available smart card/USB tokens because of the drivers or other security measures. Use this property carefully.

Time Stamping

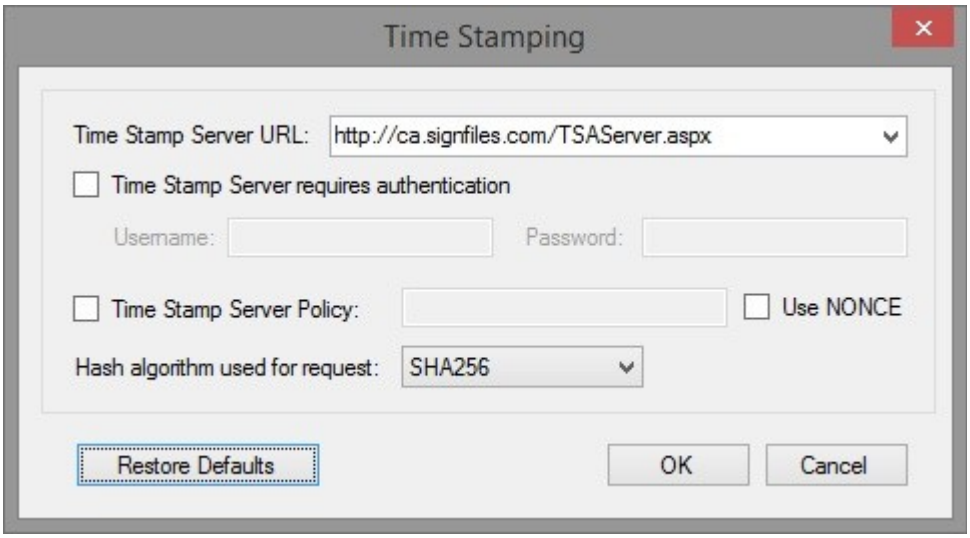
Time Stamp a Digital Signature

Timestamping is an important mechanism for the long-term preservation of digital signatures, time sealing of data objects to prove when they were received, protecting copyright and intellectual property and for the provision of notarization services.

To add time stamping information to the digital signature you will need access to a RFC 3161 time stamping server.

A fully functional version of our TSA Authority is available for testing purposes at this link: http://ca.signfiles.com/TSAServer.aspx (no credentials are needed).

The Time Stamping options can be configured on the *Time Stamping* section.



Nonce and Policy

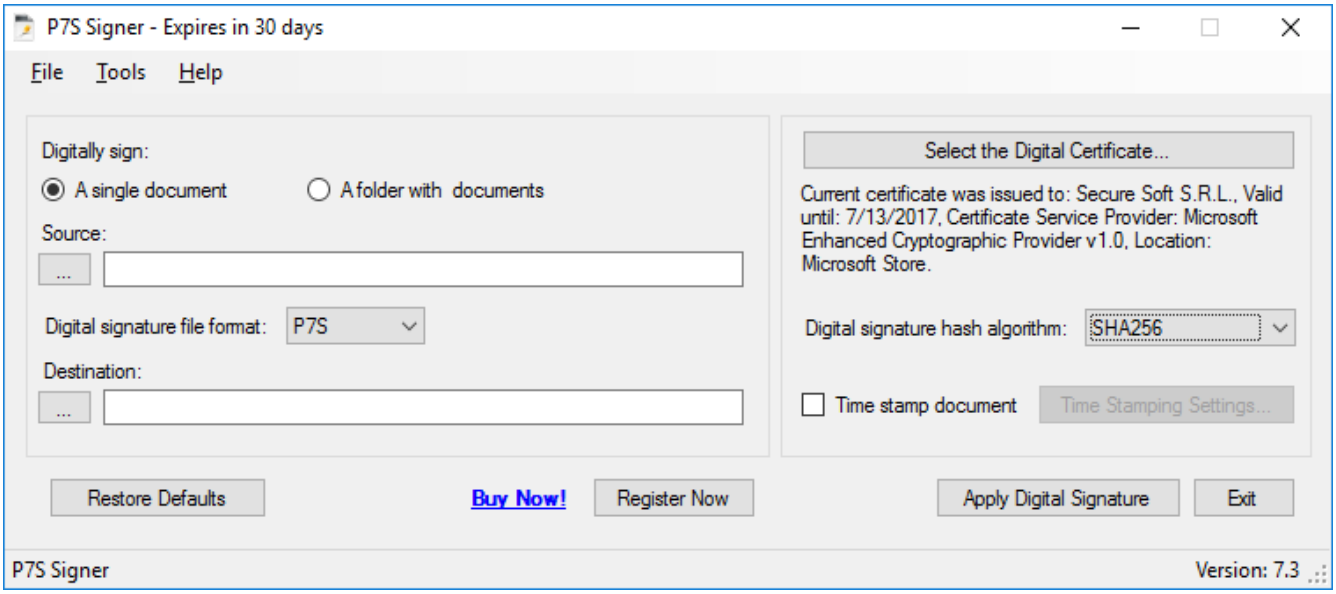
The **Nonce**, if included, allows the client to verify the timeliness of the response when no local clock is available. The nonce is a large random number with a high probability that the client generates it only once (e.g., a 64 bit integer).

Some TSA servers require to set a **Time Stamp Server Policy** on the Time Stamp Requests. By default, no Time Stamp Server Policy is included on the TSA request.

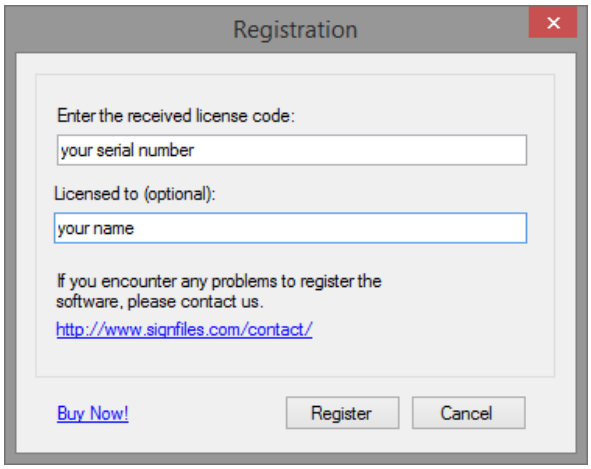
Product Registration

To register the product you will need a serial number. It can be purchased online directly from the product main page.

After you will obtain your serial number, open P7S Signer and click Register Now button.

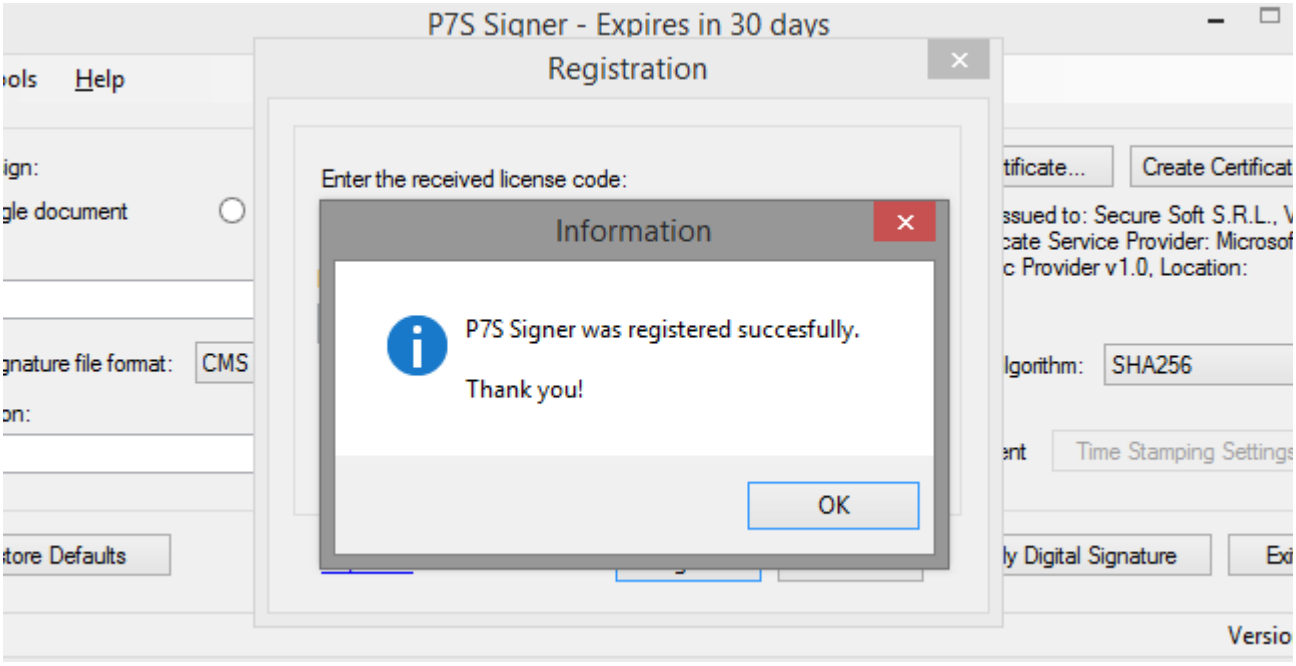


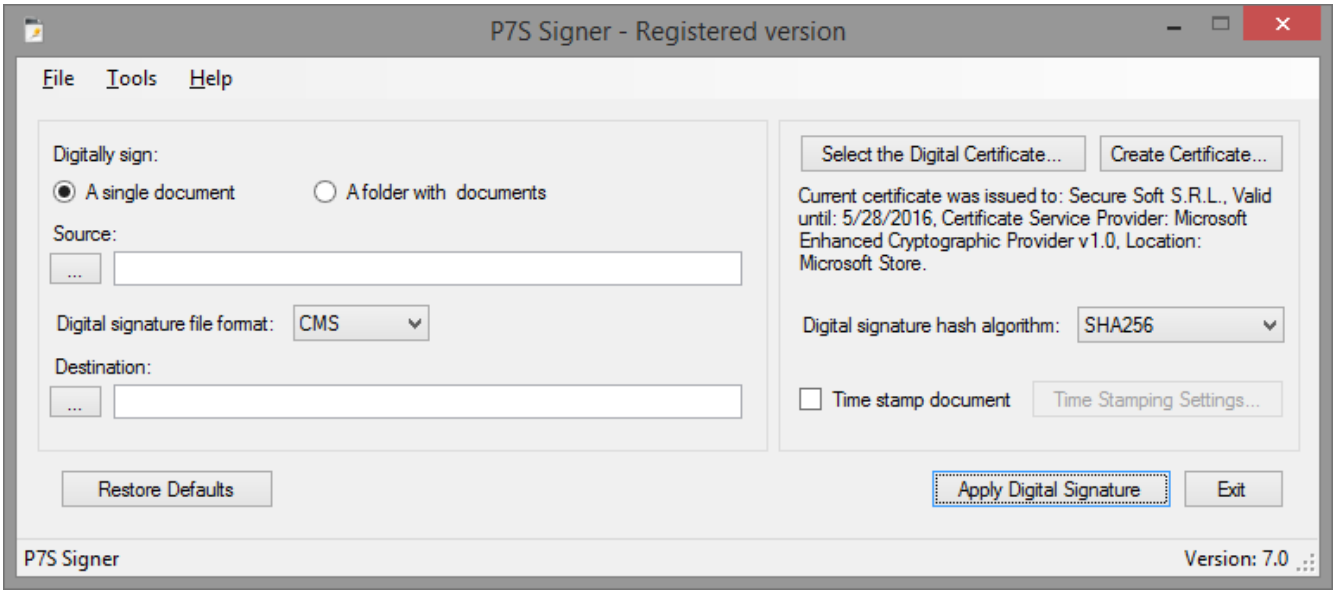
Enter the received serial on the Registration window, as below:



Click Register button.

If the serial number is correct, the product will be successfully registered.





Batch Signatures (Automatically Made Without User Intervention)

By default, P7S Signer is installed on this location:
C:\Program Files\Secure Soft\P7S Signer\P7S Signer.exe.

The command line parameters are:
P7S Signer.exe <source file | folder> <destination file | folder> [<XML configuration file>]

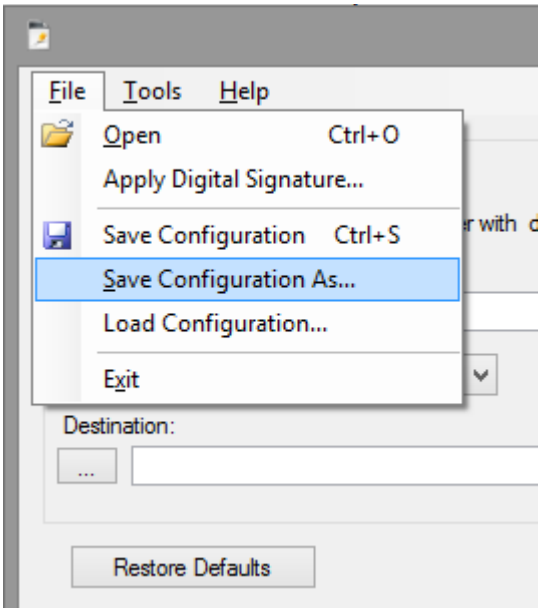
To automatically sign a **file**, use the following command:
c:\Program Files\Secure Soft\P7S Signer>"P7S Signer.exe" c:\TestFile.txt c:\TestFile.txt.p7s

To automatically sign a **folder** that contains files, use the following command:
c:\Program Files\Secure Soft\P7S Signer>"P7S Signer.exe" c:\InputFolder c:\OutputFolder

Custom Configuration

In some cases, you will need a different signature configuration (e.g. different signature appearance and digital certificates) for different files/folders.

To save a specific configuration, go to *File – Save Configuration As* and save the configuration on a file. Later, you can use that file in batch mode to apply different signature configuration on your signed file.



To automatically sign a **folder** that contains files, using a custom configuration, use the following command:

"P7S Signer.exe" c:\InputFolder c:\OutputFolder ***c:\config-client2.xml***

Digitally Sign Files Using Windows PowerShell

P7S Signer main functions are available on SignLib library available at this link: http://www.signfiles.com/sdk/SignatureLibrary.zip

To digitally sign a file using Windows PowerShell, simply download the library above and inspect *Signature Library\PowerShell Scripts* folder.

The Windows PowerShell script will look below:

```
#digitally sign a file file using a PFX certificate creted on the fly
#the format of the signed file will be CAdES
#the script can be configured to use an existing PFX file or a certificate loaded from
Microsoft Store (smart card certificate)

if ($args.Length -eq 0)
{
    echo "Usage: signInCAdESFormat.ps1 <unsigned file> <signed file>"
}
else
{

$DllPath = 'd:\SignLib.dll'
[System.Reflection.Assembly]::LoadFrom($DllPath)


#create a PFX digital certificate
     $generator = new-object -typeName SignLib.Certificates.X509CertificateGenerator("serial
number")
    $pFXFilePassword = "tempP@ssword"

    $generator.Subject = "CN=Your Certificate, E=useremail@email.com, O=Organzation"
    $generator.Extensions.AddKeyUsage([SignLib.Certificates.CertificateKeyUsage]::DigitalSig
nature)
    $generator.Extensions.AddEnhancedKeyUsage([SignLib.Certificates.CertificateEnhancedKeyUs
age]::DocumentSigning)

    echo "Create the certificate..."
    $certificate = $generator.GenerateCertificate($pFXFilePassword)

#digitally sign the file in CAdES format
    $sign = new-object -typeName SignLib.Cades.CadesSignature("serial number")
                                                $sign.DigitalSignatureCertificate           =
[SignLib.Certificates.DigitalCertificate]::LoadCertificate($certificate, $pFXFilePassword)

    echo "Perform the digital signature..."

    [System.IO.File]::WriteAllBytes($args[1], $sign.ApplyDigitalSignature($args[0]));

}
```

How to run the Windows PowerShell script from command line:

powershell -executionPolicy bypass -file d:\signInCAdESFormat.ps1 d:\test.txt d:\test.txt.p7s

Digitally Sign Files Using C# or VB.NET

P7S Signer main functions are available on SignLib library available at this link: http://www.signfiles.com/sdk/SignatureLibrary.zip

To digitally sign a file using C# or VB.NET, download the library above and inspect *Signature Library\VS2008 Projects* folder.

The C# will look like below:

```
CadesSignature cs = new CadesSignature(serialNumber);

//Digital signature certificate can be loaded from various sources

//Load the signature certificate from a PFX or P12 file
cs.DigitalSignatureCertificate =
DigitalCertificate.LoadCertificate(Environment.CurrentDirectory + "\\cert.pfx",
"123456");

//Load the certificate from Microsoft Store.
//The smart card or USB token certificates are usually available on Microsoft
Certificate Store (start - run - certmgr.msc).
//If the smart card certificate not appears on Microsoft Certificate Store it
cannot be used by the library
//cs.DigitalSignatureCertificate = DigitalCertificate.LoadCertificate(false,
string.Empty, "Select Certificate", "Select the certificate for digital
signature");

//The smart card PIN dialog can be bypassed for some smart cards/USB Tokens.
//ATTENTION: This feature will NOT work for all available smart card/USB Tokens
becauase of the drivers or other security measures.
//Use this property carefully.
//DigitalCertificate.SmartCardPin = "123456";

//optionally, the signature can be timestamped.
//cs.TimeStamping.ServerUrl = new Uri("http://ca.signfiles.com/TSAServer.aspx");

//write the signed file
//usually, the signed CAdES file should be saved with .p7s or .p7m extension
File.WriteAllBytes(signedDocument, cs.ApplyDigitalSignature(unsignedDocument));

Console.WriteLine("The CAdES signature was created." + Environment.NewLine);
```