Entrust Certificate Services

Authenticode Signing

User Guide

For software release: 11.2

Date of Issue: April 2013

Document issue: 1.0

Entrust®
Securing Digital Identities
& Information

Copyright © 2009-2013 Entrust. All rights reserved.

Entrust is a trademark or a registered trademark of Entrust, Inc. in certain countries. All Entrust product names and logos are trademarks or registered trademarks of Entrust, Inc. in certain countries. All other company and product names and logos are trademarks or registered trademarks of their respective owners in certain countries.

This information is subject to change as Entrust reserves the right to, without notice, make changes to its products as progress in engineering or manufacturing methods or circumstances may warrant.

Export and/or import of cryptographic products may be restricted by various regulations in various countries. Export and/or import permits may be required.

Obtaining technical support

For support assistance by telephone call one of the numbers below:

- 1 (866) 267-9297 in North America
- 1 (613) 270-2680 outside North America

You can also email Customer Support at:

- SSL@entrust.com

Code signing

Signing Microsoft Authenticode

This guide contains information about signing Microsoft® Authenticode files. Sections in this guide include:

- "Obtaining and using an Entrust Microsoft Authenticode signing certificate" on page 4
- "The code signing process for Microsoft Authenticode" on page 8
- "Verifying the authenticity of the software" on page 10

Browsers use the signature and its accompanying information to provide some confidence to the end user that the code is from a legitimate source and is free of tampering. Entrust offers PKCS#7 (Public Key Cryptography Standard # 7) certificates for use with Authenticode.

User guides containing information about:

- Entrust code signing certificates for Windows macros and Visual Basic files
- Entrust code signing certificates for Java code

are available from http://www.entrust.net/ssl-resources/whitepapers.htm.

Obtaining and using an Entrust Microsoft Authenticode signing certificate

When you install the certificate, a private key is created on your machine. This process provides added security, as the private key does not exist until it is created on the signer's computer. Microsoft Authenticode files can be signed using SignTool—an application that is included when you download and install the Windows Software Development Kit (SDK). The Windows SDK is available from Microsoft's Web site.

Topis in this section include:

- "Obtaining a certificate from Entrust" on page 4
- "Signing Microsoft Authenticode" on page 4
- "Signing kernel mode software using SignTool" on page 6

Obtaining a certificate from Entrust

To obtain a code signing certificate from Entrust, log into the Entrust Web site URL https://buy.entrust.net/buy. Code signing certificates can be only purchased by customers who have registered for the Entrust Certificate Management Service (CMS). For information about enrolling in the CMS see the *Entrust Certificate Management Service Enrollment Guide*. For information about buying and managing code signing certificates see the *Entrust Certificate Management Service User Guide*.

Signing Microsoft Authenticode

The procedure in this section assumes:

- that you have purchased and installed an Entrust certificate for signing Authenticode
- Microsoft SDK is installed on the machine you are using

Note:
SignTool supports PFX format. If you selected PVK format when you purchased your certificate you can convert it to PFX using the PVK2PFX utility, located in the same folder as SignTool. Alternatively, when you import the certificate into your certificate store you can specify PFX format.

Converting a PVK formatted file to PFX

This procedure outlines how convert a PVK formatted file to PFX using the PVK2PFX utility.

To convert a PVK file to a PFX file

1 From the command prompt, enter the command:

```
<Path to the PVK2PFX executable>\PVK2PFX.exe -pvk <name of PVK file>
-pi <password for PVK> -spc <name of SPC file> -pfx <name of PFX file>
```

Where:

- `<Path to the PVK2PFX executable>` is usually the `bin` folder of the Windows SDK installation
- `<name of PVK file>` is the name of the PVK file that you purchased from Entrust
- `<password for PVK>` is the password that you created to safeguard and use the certificate
- `<name of SPC file>` the name of the SPC file
- `<name of PFX file>` is an option to allow you to use a different name for the PFX file. If you choose not to use this option an export wizard opens and subsequent options (-po and -f) in the command are ignored.
- Optionally you can add `-po <password>` to set a password for the PFX file that is different from the one set for the PVK file. If you choose not to set a new password the original password applies.
- Optionally you can add `-f` to force an existing PFX file to be overwritten.

Signing a file using SignTool

There are several options available when signing files using SignTool. Some of these allow you to automate the process but may not be as secure as the method used in this example. A full explanation of the available options can be found on the Microsoft Web site.

This example, uses the PFX file in the command and requires a password.

Note:
Older versions of SignTool (version 6.0 and lower) require you to specify the timestamp in a separate command, rather than as an option in the signing command.

To sign code using SignTool

1 From the command prompt, enter the command:

```
<Path to the SignTool executable>\signtool.exe sign /f
<path>\<cert.pfx> /p <password> /t
```

`http://timestamp.entrust.net/TSS/AuthenticodeTS /v <path>\<file to be signed>`

Where:

`<Path to the SignTool executable>` is usually the `bin` folder in the Windows SDK installation.

`<path>\<cert.pfx>` is the location and name of your certificate

`<password>` is the password for your certificate

`http://timestamp.entrust.net/TSS/AuthenticodeTS` is the URL of Entrust's timestamp authority for Authenticode

Note:
This URL applies only to Authenticode signing certificates. Other types of Entrust code signing certificates use different URLs.

`/v` specifies verbose execution for information about success, failure, or errors

`<path>\<file to be signed>` is the location and name of the file containing the code that you want to sign

Signing kernel mode software using SignTool

There are several options available when signing files using SignTool.

Before you can sign kernel mode software, you must obtain an Entrust certificate that has been cross-signed with Microsoft®. These certificates are available from the Microsoft Web site at:

http://msdn.microsoft.com/en-us/windows/hardware/gg487315

The command used to sign kernel mode software is similar to the command outlined in the previous section. However, this command allows you to use the Entrust cross-certificate obtained from Microsoft with your Entrust Authenticode signing certificate to sign Kernel mode software.

This example, uses the PFX file in the command and requires a password.

Note:
Signtool 6.0 or higher is required to use this command.

To sign kernel mode software using SignTool

1. From the command prompt, enter the command:

```
<Path_to_the SignTool_executable>\signtool.exe sign /ac
<path>\<name_of_cross_certificate> /f <path>\<cert.pfx> /p
<password> /t <Entrust_timestamp_server> /v
<path>\<file_to_be_signed>
```

Where:

- `<Path_to_the_SignTool_executable>` is usually the `bin` folder in the Windows SDK installation.
- `<path>\<cross certificate>` is the location and name of the cross-certificate
- `<path>\<cert.pfx>` is the location and name of the code signing certificate
- `<password>` If you opted to create a password, use this option to enter it
- `<Entrust_timestamp_server>` is the URL of Entrust's timestamp authority for kernel mode signing (RFC3161 compliant)
  - to use a SHA1 hash in the timestamp enter the value `http://timestamp.entrust.net/TSS/RFC3161sha1TS`
  - to use a SHA2 hash in the timestamp enter the value `http://timestamp.entrust.net/TSS/RFC3161sha2TS`
- Many other options are available. For example:
- /sha1 <SHA_VALUE> where <SHA_VALUE> is the SHA1 hash value—can be used to identify the certificate by its hash value
- /d allows you to add a description of the signed content; for example the name of the software "XYZ Desktop Client"
- /du allows you to add a URL with additional information about the signed content

See the Microsoft Web site for a complete list of options.

The code signing process for Microsoft Authenticode

When the code is signed, several pieces of information are added to the file. This information is used when the code is downloaded though your browser to authenticate the author of the code and to check for tampering.

The bundle that is used to verify the authenticity of the code is created during two sequences of events.

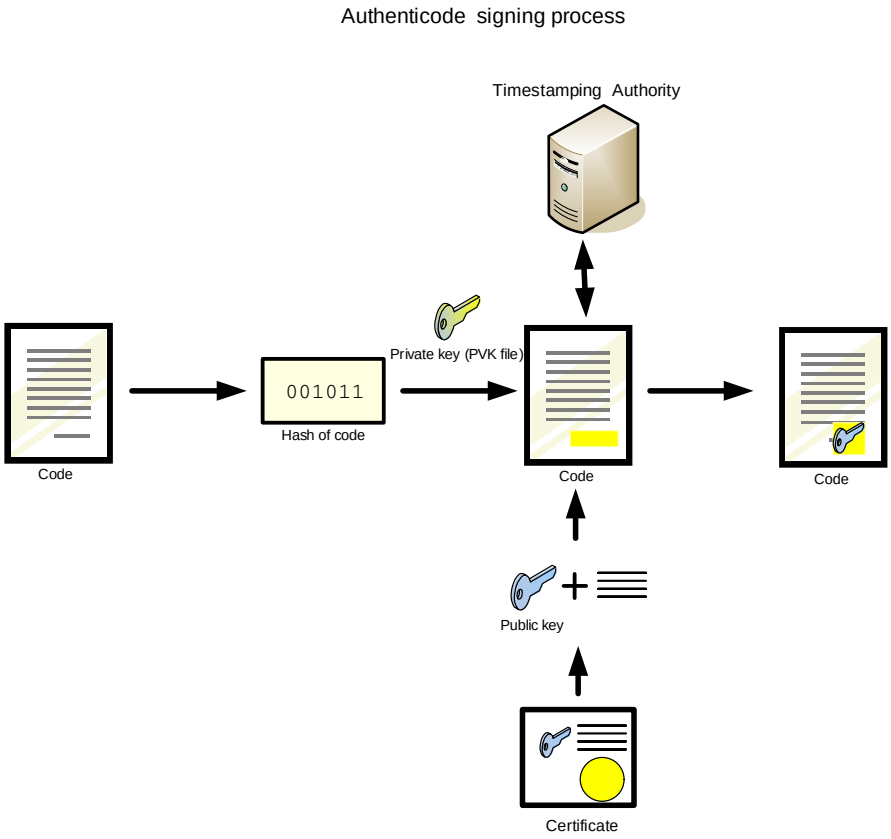
- A mathematical representation of the code, called a hash, is created and signed. The hash and signature are timestamped, hashed (with the timestamp) and signed again.
- The timestamp and second signature are applied by a timestamping authority (TSA). Timestamping Authorities are usually maintained by a third party (such as Entrust) that can insure the validity of the timestamp.

The entire sequence takes place as follows:

- The code is passed through a hashing algorithm creating a hash of the file.

  The hash is an exact numerical representation of the file. The hash is only reproducible using the unaltered file and the hashing algorithm that was used to create the hash. The hash is bundled with the file.
- The hash is signed using the signer's private key.
  - Information identifying the creator of the signature is drawn from the signer's certificate and incorporated into the signature.
  - Information about the CA or CAs that signed the signer's certificate is drawn from the signer's certificate and incorporated into the signature.
- The signer's public key is added to the bundle (it is required to authenticate the code when it is verified).
- The signature is sent to the timestamping authority (TSA).
  - The TSA adds a timestamp to the to the bundled information and computes a new hash.
  - The TSA signs the new hash with its private key creating a new bundle of information.
  - The timestamped bundle, original bundle that was sent to the TSA and the time stamp are re-bundled with the original code.

Figure 1: The code-signing process for Microsoft Authenticode



Verifying the authenticity of the software

When the end user's browser loads the code, it checks the authenticity of the software using the signer's public key, signature and the hash of the file. The timestamp is checked using a similar process.

If both the timestamp and the signature are verified successfully, the browser accepts the code as valid. If either the timestamp or signature are not successfully verified, the browser will react by warning the user or rejecting the code, depending on the level of security set in the browser.

Verifying the timestamp

The following sequence of events is used to verify the timestamp.

- The timestamp is added to the bundled signature information and the combined signature and timestamp are hashed.
- The Timestamping Authority's public key is applied to the timestamped signature block revealing the hash calculated by the TSA.
- The validity of the TSA's public key is verified by checking its expiry date and consulting the revocation lists to be sure that it has not been revoked.
- The two hashes are compared. If the hashes are equal, the timestamp is considered to be valid.

Verifying the signature

The signature is verified as follows:

- The original code is passed through a hashing algorithm creating a hash.
- The public key of the designer or publisher is extracted from the bundle and applied to the signature information. Applying the public key reveals the hash that was calculated when the file was signed.
- The expiry date of the public key is checked.
- The public key is checked against the revocation lists to be sure that it is valid.
- The two hashes are compared. If equal, the signature is considered to be valid.
- If the file is considered to be valid it is accepted by the browser. If the file is not considered to be valid the browser takes the security measure appropriate to its current level of security.

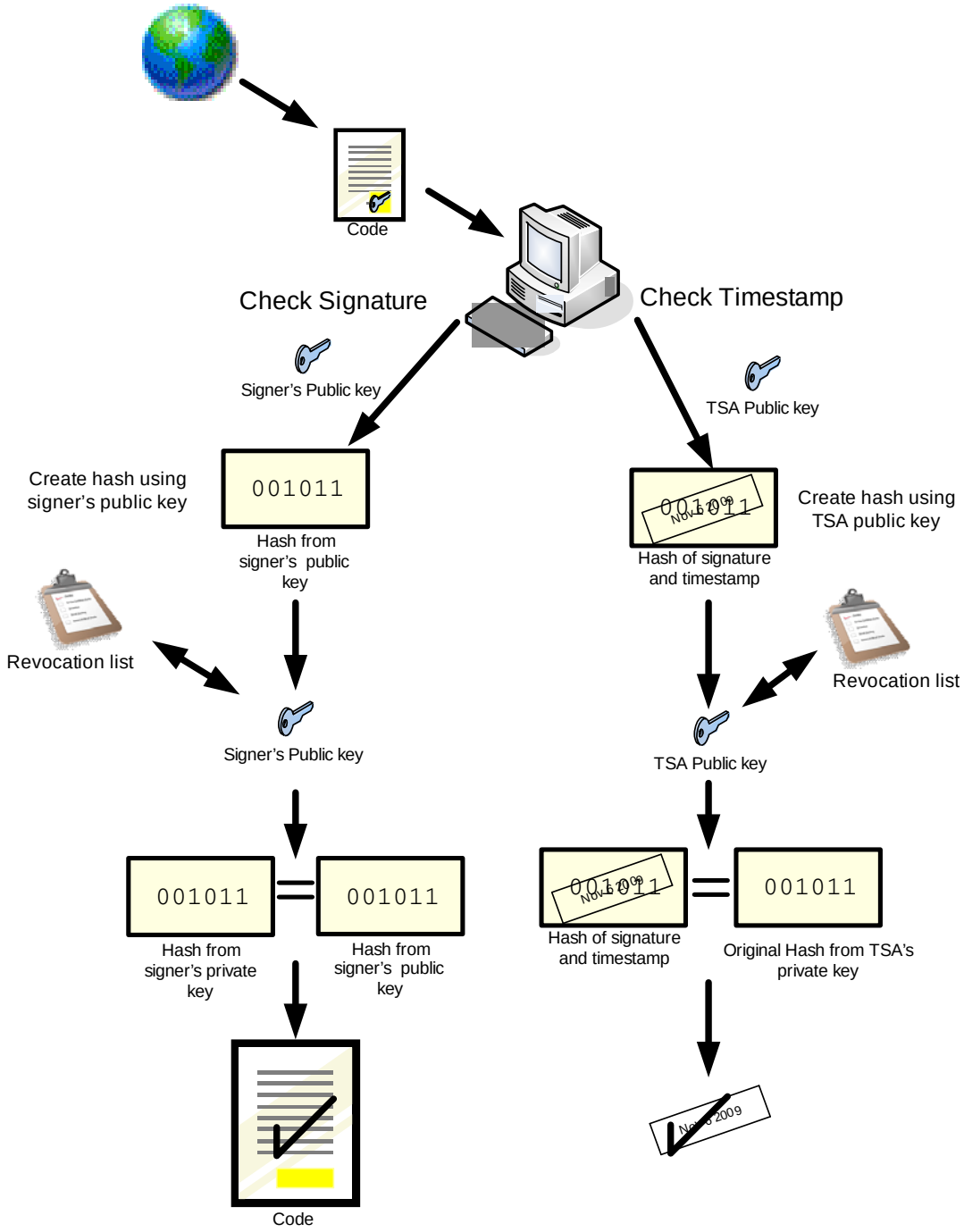



Figure 2: Verifying the authenticity of the code