

Entrust Certificate Services

Java Code Signing

User Guide

Document issue: Draft

Date of Issue: February 2009



Copyright © 2009 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:

- 1-877-754-7878 in North America
- 1-613-270-3700 outside North America

You can also email Customer Support at:

- support@entrust.com

Code signing

This guide contains information about signing Java Archive (JAR) files. Sections in this guide include:

- ["Signing JAR files"](#)
- ["Verifying the authenticity of the software"](#)
- ["Obtaining and using a code signing certificate"](#)

Entrust offers customers X.509 certificates to sign JAR files.

- For more information about Public Key Infrastructure (PKI), code signing, certificates, and keys, see the *Entrust Certificate Services Code Signing General Information Guide* available from *****
- For information about using an Entrust certificate with Microsoft Office files see the *Entrust Certificate Services Microsoft Office and VBA Code Signing Guide* available from *****
- For information about using an Entrust certificate with Microsoft Authenticode see the *Entrust Certificate Services Microsoft Authenticode Code Signing Guide* available from *****

Signing JAR files

When the code is signed, several pieces of information are added to the file. This information is used when the code is downloaded through 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.

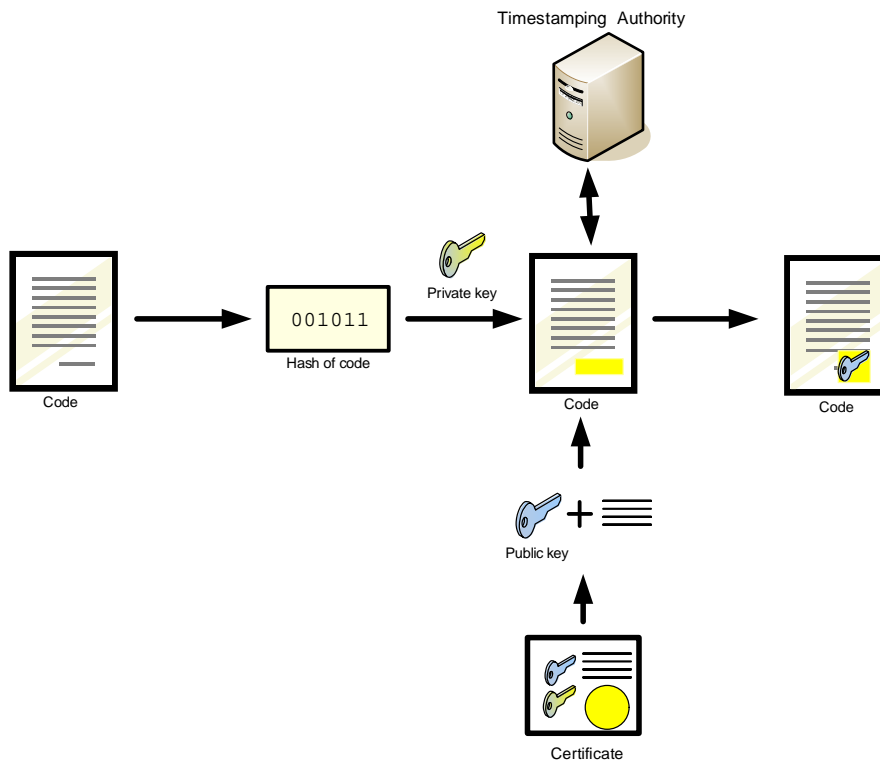
The timestamp and second signature are applied by a timestamping authority (TSA). Timestamping Authorities are usually maintained by a third party (such as Entrust) that can ensure the validity of the timestamp.

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.
- The signature is sent to the timestamping authority (TSA).
 - The TSA adds a timestamp 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 Java code

JAR signing process



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 reacts by warning the user or rejecting the code, depending on the level of security being used.

Verifying the timestamp

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

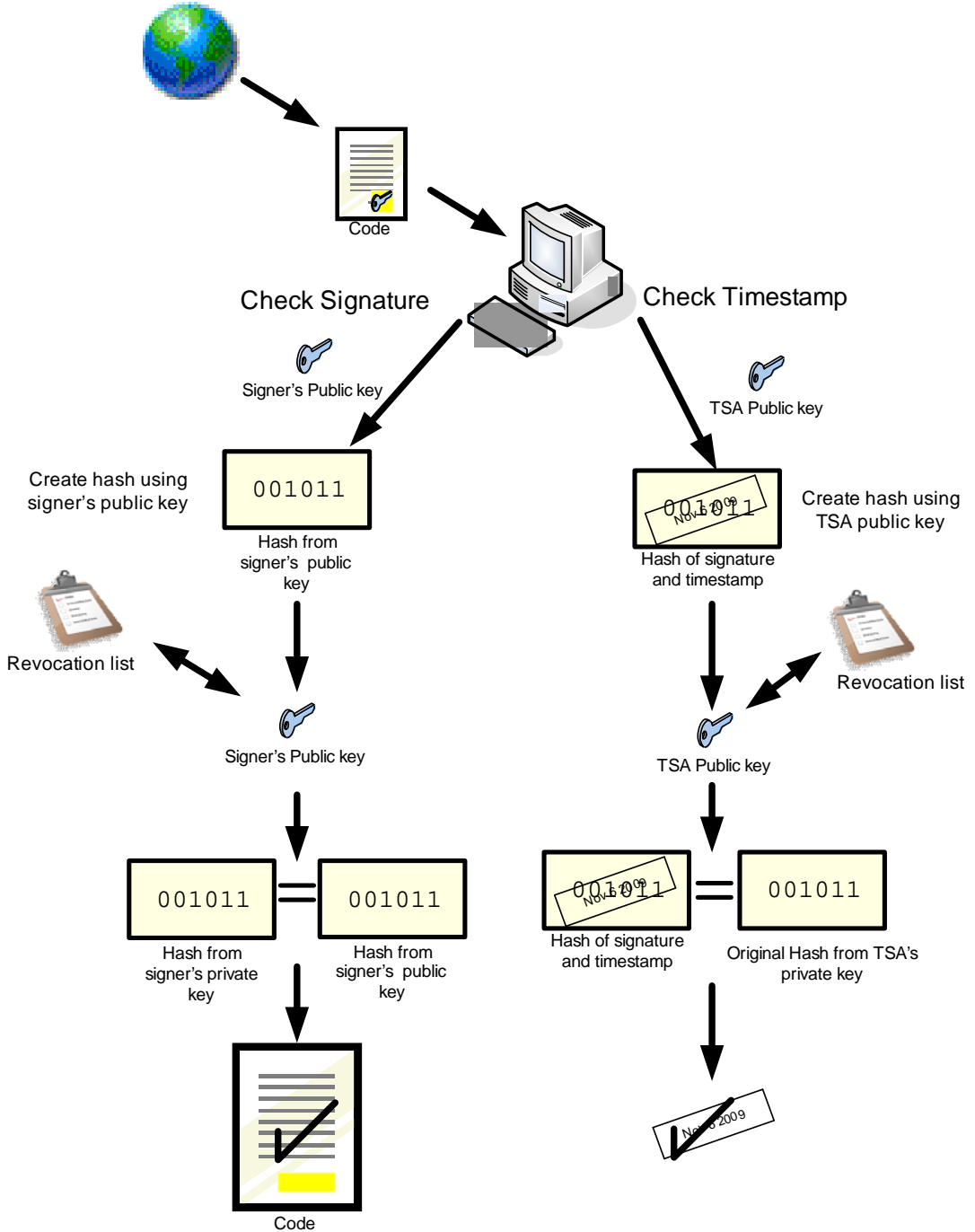
- The timestamp is added to the bundled signature information and the combined signature and timestamp are hashed.
- The Timestamping Authority's public key is applied to the timestamped signature block revealing the hash calculated by the TSA.
- The validity of the TSA's public key is verified by checking 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 ensure that it is valid.
- The two hashes are compared. If the hashes are identical, 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



Obtaining and using a code signing certificate

If you do not have a Java code signing certificate, you must obtain one before you can sign JAR files. To buy and use an Entrust certificate:

- Create a certificate signing request (CSR) as outlined in [“Creating a certificate signing request \(CSR\)”](#).
- Buy the certificate from Entrust using the Web site mentioned in [“Obtaining a certificate from Entrust”](#).

It takes time to verify the information that you include with your certificate request. Please allow three to five days for Entrust to verify your information.

- Import the certificate into your keystore as outlined in [“Importing the certificate into your keystore”](#).
- If you have not already done so, create a JAR file of your work as outlined in [“Creating JAR files”](#).
- Sign the file using your certificate as outlined in [“Signing a JAR file”](#).

Required software

Download and install the Java developer's kit (JDK). The JDK is available free of charge from java.sun.com.

Creating a certificate signing request (CSR)

To obtain a certificate you must create a certificate signing request using software installed on your computer. The JDK includes the keytool application that can be used to create a CSR.

To be verifiable, each certificate must carry specific information about the person applying for the certificate. This information is incorporated into the CSR and used by the CA to create the applicant's certificate.

To create a CSR

- 1 If you have not already done so, create a key store using the keytool application from the JDK.

- a Open a **Command Prompt** window.

- b Type the following command into the window:

```
<path to keytool>\keytool -genkey -keyalg rsa -alias  
<friendly_name>
```

Where:

<path to keytool> is the path to the keytool executable in the JDK.

- genkey is the generate key option.
- keyalg rsa indicates the rsa algorithm
- alias <friendly_name> is the friendly name that are using for your certificate

For example:

```
C:\JDKS\jdk1.6.0_03\bin\keytool -genkey -keyalg rsa
-alias example.com
```

You must supply information required to create the DN used in the keystore.

```
C:\>C:\JDKS\jdk1.6.0_03\bin\keytool -genkey -keyalg rsa -alias example.com
Enter keystore password:
Re-enter new password:
What is your first and last name?
 [Unknown]: Alice Gray
What is the name of your organizational unit?
 [Unknown]: Development
What is the name of your organization?
 [Unknown]: example.com
What is the name of your City or Locality?
 [Unknown]: Ottawa
What is the name of your State or Province?
 [Unknown]: Ontario
What is the two-letter country code for this unit?
 [Unknown]: CA
Is CN=Alice Gray, OU=Development, O=example.com, L=Ottawa, ST=Ontario, C=CA
ect?
 [no]: y
```

2 Create a certificate signing request (CSR).

- a Type the following command into the window:

```
<path to keytool>\keytool -certreq -alias <friendly_name>
```

For example:

```
C:\JDKS\jdk1.6.0_03\bin\keytool -certreq -alias MyAlias
```

- b Enter the key store password at the prompt.

Keytool creates a string similar to the following example.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBsTCCARoCAQAwcTELMAkGA1UEBhMCQ0ExEDA0BgNVBAGTB09udGFyaW8xDzANBg
NVBACTBk90dGF3YTEUMBIGAlUEChMLZxhxbXBsZS5jb20xFDASBgNVBASTC0RldmVs
b3BtZW50MRMwEQYDVQQDEWpBbG1jZSBHcmF5MIGfMA0GCsqGSIb3DQEBAQUAA4GNAD
CBiQKBgQCaxmlZnlKq3nZ0VbIsNMwHgdNilnkQlkQTpSzglM3fdUZmF9oRjNjYftBmi
gLSXMhoypi4YkOgRLZT5wnnwh1H2UPWpQk3hCNyK01chNsmTrZKnJEzku24dGXZ/H6
UXlyvGM3bugw+My0/UFeb4J/1DEN/P3Z3QVJ7614Squ+zrawIDAQABoAAwDQYJKoZI
hvcNAQEFBQADgYEADjxeKxzNKQ9LTY+FUXdVv3Lc35jq5Og2082xhzmyfnYBUkOgOY
KiQL+gY2Gm/IkLYceqDfbhBHI6WE1maMtLntbpFXgh9n/C9gLHTbPSQJbGXku0985i
J8O6au4DOPoxegBH+Zflb5mztChzD7+sKVYYI65/XatUtYv3+w1LnKk=
```

-----END NEW CERTIFICATE REQUEST-----

- 3 Copy the certificate signing request to a file. Include the `Begin New Certificate Request` and `End New Certificate Request` lines. You will use the string to obtain a certificate from Entrust.

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 are available to users who have registered for the Entrust Certificate Management System (CMS). For information about enrolling in the CMS see the *Entrust Certificate Management System Enrollment Guide*. For information about buying and managing code signing certificates see the *Entrust Certificate Management System User Guide*.

Importing the certificate into your keystore

Use the `keytool` application (part of the JDK) to import the certificate into your keystore. Type:

```
<Path to keytool>\keytool -import -alias <alias of your certificate> -file <certificate name>
```

Where:

`<Path to keytool>` is the location for the `keytool` executable in the JDK.

`-import` specifies that `keytool` should import the certificate.

`-alias` is the friendly name of the certificate.

`-file` indicates the certificate being imported.

For example:

```
C:\JDKS\jdk1.6.0_03\bin\keytool -import -alias MyAlias -file Entrust_CS_Cert
```

Creating JAR files

Before you can sign a Java applet's files, you must bundle them into a JAR file. The Java SDK contains a tool for creating JAR bundles. It can be used from the command prompt.

If you are familiar with creating TAR bundles you will find creating a JAR bundle is very similar.

Creating and checking the JAR file

- 1 Create a directory (or directory and subdirectories—depending on the directory structure you require) for your applet's files and place them in that location.

- 2 From the directory (or the top level of the directory structure that you created), run the jar tool. For example:

```
C:\JDKS\jdk1.6.0_03\bin\jar cvf C:\test.jar
```

Where:

- C:\JDKS\jdk1.6.0_03\bin\ is the location of the jar tool (jar.exe)
- cvf stands for:
 - create (create file)
 - verbose (see all messages from the tool)
 - file (use the specified filename and location)
- C:\test.jar is the name and location that you specify for the resulting JAR file.

This command creates the JAR file C:\test.jar in the location specified.

- 3 Optionally, verify the newly created JAR file using the jar tool. For example,

```
C:\JDKS\jdk1.6.0_03\bin\jar tvf C:\test.jar
```

Where:

- C:\JDKS\jdk1.6.0_03\bin\ is the location of the jar tool executable (jar.exe).
- tvf stands for:
 - test (test file)
 - verbose (see all messages from the tool)
 - file (use the specified filename and location)

Testing ensures that the JAR file has the correct content and structure.

Signing a JAR file

Use the JDK keytool and jarsigner applications to sign your code. These instructions explain how to sign a JAR file.

To sign a JAR file

- 1 Optionally, check that the keytool application (the JDK application used to sign the applet), can read your keystore.

a Open a command prompt.

b Enter the command:

```
<path to keytool>\keytool -list <Certificate_Name>
```

For example:

```
\JDKS\jdk1.6.0_03\jre\bin\keytool -list EntrustCert
```

c Enter the password when prompted.

The keytool lists information about entries in your keystore.

- 2** To sign the JAR file, you need the friendly name of the key entry that you are using. If you do not know the friendly name of the key entry use the following procedure.

- a** To obtain the friendly name, use keytool's list and verbose options:

```
<path to keytool>\keytool -list <certificate_name> -v
```

For example:

```
\JDKS\jdk1.6.0_03\jre\bin\keytool -list EntrustCert -v
```

- b** Enter the password when prompted.

Keytool lists information about the keystore including the friendly name.

- 3** To sign the JAR file, use the jarsigner tool included in the JDK.

- a** Enter the following command:

```
<path to jarsigner>\jarsigner <certificate name> <path to  
JAR file>\<JAR file name.jar> <friendly name of key entry>
```

For example:

```
\JDKS\jdk1.6.0_03\bin\jarsigner EntrustCert  
\work\MyProject\MyApplet.jar MyAlias
```

- b** Enter the password when prompted.

You have successfully signed a JAR file.