



White Paper

# Code Signing Digital IDs for Sun<sup>®</sup> Java<sup>™</sup> Signing

---

# Code Signing Digital IDs for Sun Java Signing

*Realizing the Possibilities of Internet Software Distribution*

## What Is Code Signing?

When customers buy software in a store, the source of that software is obvious. Customers can tell who published the software, and they can see whether the package has been opened. These factors enable customers to make decisions about what software to purchase and how much to “trust” those products.

When customers download software from the Internet, the most they see is a message warning them about the dangers of using the software. The Internet lacks the subtle information provided by packaging, shelf space, and shrink-wrap. Without any assurance of the software’s integrity, and without knowing who published the software, it’s difficult for customers to know how much to trust software downloaded from the Internet.

The solution to these issues is Sun Java Object Signing coupled with **VeriSign® Code Signing Digital IDs**. Object Signing, through the use of digital signatures, enables software developers to include information about themselves and their code with their software.

When customers download software signed with a Sun Java Code Signing Digital ID issued by VeriSign, they can be assured of:

**Content Source:** End users can confirm that the software really comes from the publisher who signed it.

**Content Integrity:** End users can verify that the software has not been altered or corrupted since it was signed.

Users benefit from this software accountability because they know who published the software and that the code hasn’t been tampered with. In the extreme case that software performs unacceptable or malicious activity on their computers, users can also pursue recourse against the publisher. This accountability and potential recourse serve as a strong deterrent to the distribution of harmful code.

Developers and Web masters benefit from Object Signing because it builds trust in their names and makes it more difficult to falsify their products. By signing their code, developers build a trusted relationship with users, who learn they can download software signed by that publisher or Web site with confidence. With Sun Java Signing, developers can create exciting Web pages using signed Java applets, plug-ins, or other executables. And users can make educated decisions about what software they want to download.

## Who Needs a Code Signing Digital ID?

Any software publisher planning to distribute code or content over the Internet or through an extranet risks impersonation and tampering. VeriSign Code Signing Digital IDs for Sun Java Signing protect against these hazards.

VeriSign offers **Code Signing Digital IDs** designed for commercial software developers: companies and other organizations that publish software. This class of Digital ID provides assurance regarding an organization's identity and legitimacy, much like a business license, and is designed to represent the level of assurance provided today by retail channels for software.

## What Does Object Signing Look Like to End Users?

Applications that run Java Applets and applications on the Java Runtime Environment (JRE) come with security features that recognize Object Signing. For example, when users visit a Web page that uses Java applets to provide animation or sound, their browsers download code to their machines to achieve the desired effects. While this may provide substantial value, users run the risk of downloading viruses or other unwanted code.

When an application running on the JRE encounters a software component that is trying to gain access to the user's machine, it automatically checks to see if there is a recognized digital signature with that software. If the code is signed with Sun Java Signing, JRE prompts the application to display a warning dialog box similar to the following:



Image 1: The Java Plug-in Security Warning dialog box launched by the WebStart launcher.

The warning dialog box informs the end user:

Of the true identity of the publisher

Of the type of access requested by the software

That VeriSign has authenticated the identity of the code signer.

The end user can choose to grant or deny the requested privileges, or to view the Digital ID used to sign the code. The warning dialogs also provide the user with an estimated level of the risk (high, medium, or low) associated with the requested privileges and a means for viewing the details included in the software developer's Digital ID.

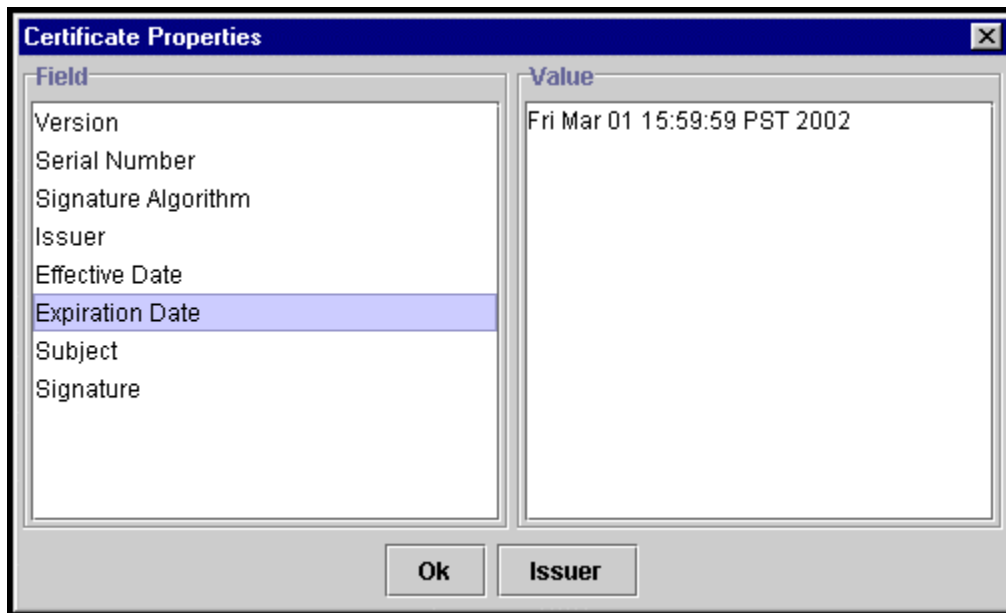


Image 2: The Certificate Properties dialog box launched by the WebStart launcher.

# Technical Background

## What is a Digital ID?

A Digital ID (also known as a digital certificate or Code Signing Digital ID) is a form of electronic credential for the Internet. Similar to a driver's license, employee ID card, or business license, a Digital ID is issued by a trusted third party to establish the identity of the ID holder. The third party who issues the Digital ID is known as a Certification Authority (CA).

Digital ID technology is based on the theory of public key cryptography. In public key cryptography systems, every entity has two complementary keys—a public key and a private key—which function only when they are held together. Public keys are widely distributed to users, while private keys are kept safe and only used by their owner. Any code digitally signed with the publisher's private key, can be successfully verified only with the complementary public key. Another way to look at this is that code that is successfully verified using the publisher's public key (which is sent along with the digital signature) can only have been digitally signed using the publisher's private key (thus authenticating the source of the code), and has not been tampered with. For more information on public keys and private keys, please see the VeriSign white paper, "Introduction to Public Key Cryptography.")

The purpose of a Digital ID is to reliably link a public/private key pair with its owner. When a CA such as VeriSign issues Digital IDs, it verifies that the owner is not claiming a false identity. Just as when a government issues you a passport it is officially vouching for the fact that you are who you say you are, when a CA issues you a Digital ID, it is putting its name behind the statement that you are the rightful owner of your public/private key pair.

A Digital ID is valid only for the period of time specified by VeriSign. The ID contains information about its beginning and expiration dates. VeriSign can also revoke (cancel) any Digital ID it has issued and maintains a list of revoked Code Signing IDs. This list of revoked Code Signing Digital IDs, called a Code Signing Digital ID Revocation List (CRL), is published by VeriSign so anyone can determine the validity of any Digital ID.

## **Certification Authorities**

Certification Authorities, such as VeriSign, are organizations that issue Digital IDs to applicants whose identities they are willing to vouch for. Each Digital ID is linked to the Code Signing Digital ID of the CA that signed it.

As the Internet's leading Certification Authority, VeriSign has the following responsibilities:

- Publishing the criteria for granting, revoking, and managing Digital IDs
- Granting Digital IDs to applicants who meet the published criteria
- Managing Digital IDs (for example, enrolling, renewing, and revoking them)
- Storing VeriSign's root keys in an exceptionally secure manner.
- Verifying evidence submitted by applicants
- Providing tools for enrollment
- Accepting the liability associated with these responsibilities

## **How Does Object Signing Work with VeriSign Code Signing Digital IDs?**

Sun Java Signing relies on industry-standard cryptography techniques such as X.509 v3 Code Signing IDs and PKCS #7 and #10 signature standards. These well-proven cryptography protocols ensure a robust implementation of code signing technology.

Object Signing uses digital signature technology to assure users of the origin and integrity of software. In digital signatures, the private key generates the signature and the corresponding public key validates it. To save time, the Object Signing protocols use a cryptographic digest, which is a one-way hash of the document.

**Note:** Java Runtime Environment (JRE) version 1.3 or newer must be installed on the end user's machine.

### **The Code Signing Process:**

1. Publisher creates code.
2. Publisher obtains a Sun Java Code Signing Digital ID from VeriSign.
3. Using the Java 2 Software Development Kit, the publisher:
  - Creates a hash of the code, using an algorithm such as MD5 or SHA
  - Encrypts the hash using their private key
  - Creates a package containing the code, the encrypted hash, and the publisher's Code Signing Digital ID
4. The end-user's Java Runtime Environment (JRE) encounters the package.

5. The end-user's JRE examines the publisher's Code Signing Digital ID. Using the VeriSign root public key that is already embedded in the end-user JRE trusted root store, the end-user JRE verifies the authenticity of the Code Signing Digital ID (which is itself signed by the VeriSign root private key).
6. Using the publisher's public key contained within the publisher's Digital ID, the end-user JRE decrypts the signed hash.
7. The end-user JRE runs the code through the same hashing algorithm as the publisher, creating a new hash.
8. The end-user JRE compares the two hashes. If they are identical, the JRE sends a message stating that the content has been verified by VeriSign. The end user is assured that the code was signed by the publisher identified in the Digital ID and that the code hasn't been altered since it was signed.

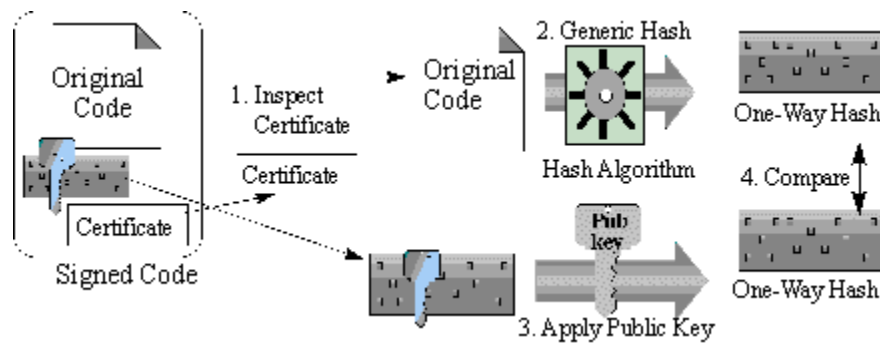


Image 3: Verifying the Code Signing Digital ID on the end user's computer.

The entire process is transparent to end users, who see only a message that the content was signed by its publisher and verified by VeriSign.

## The Six Steps to Signing Code

These instructions provide an overview of obtaining and using Sun Java Signing and a Code Signing Digital ID from VeriSign.

### Step 1: Download the Java 2 Software Development Kit (SDK).

The Java2 SDK for the Solaris SPARC/x86, Linux86, and Microsoft Windows platforms is available free of charge from [java.sun.com](http://java.sun.com).

You will be using the following tools to apply for your Code Signing Digital ID and sign your code: keytool, jar, and jarsigner.

### Step 2: Generate a public/private key pair.

Enter the following code, specifying an alias for your keystore, to generate a public/private key pair.

```
C:\>C:\jdk1.3\bin\keytool -genkey -keyalg rsa -alias MyCert
```

---

In this string, the keystore alias is MyCert.

Keytool responds with prompts to enter a password for your keystore, your name, organization, and address information. The public/private key pair generated by keytool is saved to your keystore and will be used to sign Java Applets and Applications.

**Note:** Your private key is never sent to VeriSign, so if you lose it, you will be unable to sign code. If your private key is lost or stolen, please contact VeriSign to cancel your Code Signing Digital ID.

### Step 3: Generate a Code Signing Digital ID Signing Request (CSR).

Enter the following code to generate a CSR:

```
C:\>C:\jdk1.3\bin\keytool -certreq -alias MyCert
```

---

In this string, keytool is requested to create a CSR for the key pair in the keystore MyCert.

After prompting you to enter the password for your keystore, keytool will generate a CSR.

```
-----BEGIN NEW CODE SIGNING ID REQUEST-----
MIIBtjCCAR8CAQAwZjELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAkNBMRIwE
AYDVQQHEw1DdXB1cnRpbm8xGTAXBgNVBAoTEFN1biBNaWNyb3N5c3R1bX
MxFjAUBgNVBAStDUphdmEgU29mdHdhcmUxEzARBgNVBAMTC1N0YW5sZXk
gSG8wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALTgU8PovA4y59eb
oPjY65BwCSc/zPqtOZKJlaW4WP+UhmebE+T2Mho7P5zXjGf7elo3tV5uI
3vzgGfnhgpf73EoMow8EJhly4w/YsXKqeJEqqvNogzAD+qUv7Ld6dLOv0
CO5qvpmbAO6mfaI1XAgx/4xU/6009jVQe0TgIoocB5AgMBAAGgADANBgk
qhkiG9w0BAQQFAAObgQAWmLrkifKiUYtd4ykhBtPWSwW/IKkgyfIuNMML
dF1DH8neSnXf3ZLI32f2yXvs7u3/xn6chnTXh4HYCJoGYOAbB3WNbAoQR
i6u6TLL0vgv9pMNUo6v1qB0xly1faizjimVYBwLhOenkA3Bw7S8UIVfdv
84c09dFUGcr/Pfrl3GtQ==
-----END NEW CODE SIGNING ID REQUEST-----
```

---

This string is an example of a CSR generated using keytool. A CSR contains a copy of the requestor's public key and a hash of the data entered in step 2 signed with the requestor's private key.

Copy the CSR and paste it into the [VeriSign Sun Java Signing Digital ID application form](http://www.verisign.com/products/signing/index.html), accessible at <http://www.verisign.com/products/signing/index.html>.

When your request is approved, VeriSign will attach your Sun Java Code Signing Digital ID to your confirmation e-mail.

Upon receipt, the attached Code Signing Digital ID is saved to a file on your computer.

A Code Signing Digital ID is a "trust path" or "chain" back to the VeriSign root certificate. This "trust path" allows your code to be validated on any standard JRE without installing any additional files.

**Note:** VeriSign takes a number of steps to verify your identity. For commercial publishers, VeriSign does a considerable amount deal of background checking. As a result, it will take approximately 3-5 business days to verify your information and issue a Code Signing Digital ID.

#### **Step 4: Import your Sun Java Signing Code Signing Digital ID.**

Enter the following code, with the path to your Code Signing Digital ID, to import the chain into your keystore.

```
C:\>C:\jdk1.3\bin\keytool -import -alias MyCert -file
VSSStanleyNew.cer
```

---

In this string, keytool is requested to import the Code Signing Digital ID "VSSStanleyNew.cer" into the keystore MyCert.

### Step 5: Bundle your applet into a JAR file.

Use `jar` to bundle your Applets or applications as a JAR file.

```
C:>C:\jdk1.3\bin\jar cvf C:\TestApplet.jar .
```

---

This string creates a JAR file `C:\TestApplet.jar`. The JAR file contains all the files under the current directory and its sub-directories.

Jar responds with:

```
added manifest
adding: TestApplet.class (in = 94208) (out= 20103)(deflated 78%)
adding: TestHelper.class (in = 16384) (out= 779)(deflated 95%)
```

### Step 6: Sign your applet.

Use `jarsigner` to sign the JAR file, using the private key you saved in your keystore.

```
C:\>C:\jdk1.3\bin\jarsigner C:\TestApplet.jar MyCert
```

At the prompt, enter the password to your keystore.

Jarsigner hashes your Applet or Application, and stores the hash in the JAR file created in step 5 with a copy of your Code Signing Digital ID.

Verify the output of your signed JAR file.

```
C:>C:\jdk1.3\bin\jarsigner -verify -verbose -certs
d:\TestApplet.jar
```

---

This string verifies that the files have been saved to the JAR file and that the signature is correct.

When the signed JAR file is downloaded, the Java Runtime Environment will display your Digital ID to the user. If the file is tampered with in any way after it has been signed, the user will be notified and given the option of refusing installation.

For more in-depth instructions on the use of the Java 2 Software Development Kit, please see the [Java™ 2 Platform, Standard Edition, v 1.3 Documentation](http://www.javasoft.com/j2se/1.3/docs.html), available at <http://www.javasoft.com/j2se/1.3/docs.html>.

## Conclusion

Sun Microsystems and VeriSign are committed to making the Internet a secure and viable platform for commerce and the distribution of content. With Sun Java Object Signing and a VeriSign Code Signing ID, your code will be as safe and trustworthy to your customers as it would be if you shrink-wrapped it and sold it off a store shelf.

For more information about VeriSign Code Signing Digital IDs for Sun Java Signing, including pricing, availability, and Frequently Asked Questions, please visit [www.verisign.com/developers](http://www.verisign.com/developers).



©2001 VeriSign, Inc. All rights reserved. VeriSign and the VeriSign logo and other trademarks, service marks, and logos are trademarks and service marks or registered trademarks and service marks of VeriSign, Inc. and its subsidiaries in the United States and in foreign countries. Sun and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in foreign countries. 6/01