Signing Software with Netscape Signing Tool 1.0

Recycled and Recyclable Paper

Netscape Compass Server 3.0

©Netscape Communications Corporation 1995-1998
All Rights Reserved

Printed in USA
99 98 97 10 9 8 7 6 5 4 3 2 1

Netscape Communications Corporation, 501 East Middlefield Road, Mountain View, CA 94043

Signing Software with Netscape Signing Tool 1.0

This document describes how to use version 1.0 of the Netscape Signing Tool (`signtool` on the command line) to digitally sign software, including binary files intended for distribution via SmartUpdate, Java class files, and JavaScript scripts. Version 1.0 includes all the capabilities of, and is fully compatible with, previous versions of the Netscape Signing Tool (.50 and .60).

Note This document describes version 1.0 only. For information about version .60 (`zigbert` on the command line)--for example, if you need to maintain scripts based on version .60--see Signing Software with Netscape Signing Tool .60. For new projects, Netscape recommends using version 1.0.

Chapter

1

Introduction to the Netscape Signing Tool

This chapter reviews basic concepts that you need to understand before you begin using version 1.0 of the Netscape Signing Tool to sign files or JavaScript scripts. If you are already familiar with object-signing concepts, go straight to Chapter 2, "Using the Netscape Signing Tool."

Important Version .60 of the Netscape Signing Tool (`zigbert` on the command line) has been superseded by version 1.0 (`signtool` on the command line). For information about version .60 of the Netscape Signing Tool, see Signing Software with Netscape Signing Tool.60.You may need this information, for example, to maintain Page Signer Perl scripts based on version .60. Such scripts are unnecessary with version 1.0 of the Netscape Signing Tool. However, Perl scripts written for use with version .60 are still supported in version 1.0.

Sections in this chapter:

For a complete introduction to Object Signing, see Netscape Object Signing: Establishing Trust for Downloaded Software on Netscape DevEdge.

What Is the Netscape Signing Tool?

The Netscape Signing Tool is a stand-alone command-line tool that creates digital signatures and uses the Java Archive (JAR) format to associate them with files in a directory.

The Netscape Signing Tool is available for Windows 95 and NT, Solaris, and IRIX. It is intended for use by system administrators using Netscape Mission Control and by developers who want to distribute software electronically over the Internet.

This document describes how to use Netscape Signing Tool 1.0 to sign Java applets, JavaScript scripts, plug-ins, and other files and how to package the signed objects in a **JAR archive** (also called a **JAR file**), which is a digital envelope for a compressed collection of files. Communicator client software uses JAR archives to install or update software automatically.

Electronic software distribution over any network involves potential security problems. To help address some of these problems, you can associate digital signatures with the files in a JAR archive. **Digital signatures** allow Communicator to perform two operations that are important to end users:

- confirm the identity of the individual, company, or other entity whose digital signature is associated with the files
- check whether the files have been tampered with since being signed

You do not need to understand the technical details of JAR archives or digital signatures to use the Netscape Signing Tool. However, you do need some familiarity with the concepts described in the rest of this chapter. If you are already familiar with basic object-signing concepts, go straight to Chapter 2, "Using the Netscape Signing Tool."

What's New in Version 1.0

In addition to the capabilities supported by earlier versions of the Netscape Signing Tool, version 1.0 supports includes new options that allow you to

- List the available signing certificates and their issuing CAs (see Chapter 2, "Using the Netscape Signing Tool").

- Create a JAR file automatically without the use of a separate ZIP utility (see Chapter 2, "Using the Netscape Signing Tool").
- Generate test object-signing certificates (see Chapter 4, "Generating Test Object-Signing Certificates").
- Create multiple JAR files automatically for a directory containing HTML files with JavaScript archive tags (see Chapter 5, "Signing Inline JavaScript Scripts").
- Perform cryptographic operations in FIPS-140-1 validated mode with the aid of Netscape's FIPS-validated cryptographic module. (see Chapter 7, "Netscape Signing Tool and FIPS-140-1").
- Use DSA keys to sign and verify JAR files.

For a complete list of the Netscape Signing Tool command-line options, see Chapter 3, "SignTool Syntax and Options."

JAR Format and JAR Archives

The **Java Archive (JAR) format** is a set of conventions for associating digital signatures, installer scripts, and other information with files in a directory. Signing tools such as the Netscape Signing Tool allow you to sign files using the JAR format and package them as a single JAR file. JAR files are used by Communicator client software to support automatic software installation, user-controlled access to local system resources by Java applets, and other features that help address potential security problems.

The **JAR file type** is a registered Internet MIME type based on the standard cross-platform ZIP archive format. A JAR file functions as a digital envelope for a compressed collection of files. The JAR file type is distinct from the JAR format, which is simply a way of organizing information in a directory.

Because the JAR format doesn't require a digital signature to be stored physically inside the file with which it is associated, JAR files are extremely flexible. You can use the Netscape Signing Tool to sign any files, including Java class files, Communicator plug-ins, or any other kind of document or application. You can also use version 1.0 of the Netscape Signing Tool to sign inline JavaScript scripts.

You must create a JAR file if you want to take advantage of Communicator's SmartUpdate feature. Communicator can automatically locate, download, and install components, plug-ins, and Java classes on a user's machine, thus freeing the user from this chore. Automatic software installation also helps both software developers who want to distribute software and updates over the Internet and system administrators using Mission Control to manage a corporate intranet.

For example, if you are a software developer, you should test your code to make sure it is virus-free before signing it. Similarly, if you are a network administrator, you should make sure, before signing any code, that it comes from a reliable source and will run correctly with the software installed on the machines to which you are distributing it.

Public-Key Cryptography and Certificates

Public-key cryptography involves a pair of keys associated with each individual or entity that needs to send secure messages electronically: a public key and a private key. Each public key is published, but the corresponding private key is kept secret. Messages encrypted with your private key can be decrypted only with your public key, and messages encrypted with your public key can be decrypted only with your private key.

You can freely distribute a public key, and as long as you keep the corresponding private key safe, only you will be able to read messages encrypted using the public key. You can also use your private key to sign a message with your digital signature, which allows Communicator software (with the aid of your public key) to confirm that the message was signed with your private key and that it hasn't been tampered with since being signed.

To obtain a public key and a corresponding private key for your own use, you must request a certificate from a certificate authority. **Certificate authorities (CAs)** are trusted entities, such as VeriSign, that issue certificates after verifying the identities of the persons or organization requesting them. When you request a certificate, your copy of Communicator generates both a public key and the corresponding private key and sends the public key to the certificate authority. The certificate authority gives you a certificate, which is a digital document that binds a particular public key to an individual or entity. Certificates help prevent the use of fake public keys for impersonation.

A certificate is like a driver's license, a passport, or any other personal ID that provides generally recognized proof of a person's identity. A certificate always includes a public key and the name of the entity it identifies. Most certificates also include an expiration date, the name of the certificate authority that issued the certificate, a serial number, and other information. Most importantly, a certificate includes the digital signature of the certificate authority.

Every digital signature points to a certificate that validates the public key of the signer. Communicator can check the validity of not only the signer's public key but also the certificate issuer's public key, the public key of the authority that issued the certificate issuer's certificate, and so on. This process of checking the certificate hierarchy continues until Communicator reaches a certificate authority that is included in its list of accepted CAs. If Communicator can't successfully traverse the certificate hierarchy and identify a CA included in its list, it won't accept the original digital signature.

In addition to allowing the exchange and verification of encrypted and signed messages, public-key cryptography and certificates facilitate digital signing of files using tools such as the Netscape Signing Tool. However, before you can use the Netscape Signing Tool to sign files, you must have an object-signing certificate, which is a special certificate whose associated private key is used to create digital signatures using Netscape object-signing technology. For information about obtaining your own signing certificate, see Object-Signing Tools on Netscape DevEdge.

Java applets that require special access to local system resources must be signed with the private key associated with an object-signing certificate that Communicator client software can validate. They must also use a set of classes, called the Capabilities classes, that add facilities to and refine the control provided by the standard class `java.SecurityManager.` The Capabilities classes let Java applets explicitly request the kind of access they need. For information about the Capabilities classes, see Java Capabilities API on Netscape DevEdge.

When you receive an object-signing certificate for your own use, it is automatically installed in your copy of the Communicator client software. Communicator supports the public-key cryptography standard known as PKCS #12, which governs key portability. You can, for example, move an object-signing certificate and its associated private key from one computer to another on a credit-card-sized device called a smart card. For information about using the Netscape Signing Tool with smart cards, see Chapter 6, "Using the Netscape Signing Tool with Smart Cards."

For more information about public-key cryptography and object signing, see Netscape Object Signing: Establishing Trust for Downloaded Software on Netscape DevEdge.

Chapter

2

Using the Netscape Signing Tool

This chapter describes how to use version 1.0 of the Netscape Signing Tool to create digital signatures for files in a directory and to associate the signatures with the files according to the JAR format. Netscape Signing Tool 1.0 also provides an option that automatically creates a JAR file containing the directory; this option was not implemented in earlier versions. For maximum flexibility, and for compatibility with scripts that used earlier versions of the Netscape Signing Tool, you can still use a ZIP utility to create the JAR file.

For a complete list of the Netscape Signing Tool command-line options, see Chapter 3, "SignTool Syntax and Options."

Sections in this chapter:

Getting Ready to Use the Netscape Signing Tool

Before using the Netscape Signing Tool, you must have the `signtool` executable in your path environment variable. You must also have an object-signing certificate.

Netscape Signing Tool 1.0 includes an option that allows you to generate an object-signing certificate for testing purposes. For information about using this

`.netscape` directory, put the existing `key3.db` and `cert7.db` files in some other directory before replacing them with the versions that include the object-signing certificate you want to use with the Netscape Signing Tool.

If you are using Unix, set up an alias to call `signtool`, or place it in your path.

If you are using Windows 95 or NT, the `signtool` executable doesn't know where your certificates are, so either put the `key3.db` and `cert7.db` files in the current directory and use "`-d.`" or use `-d` to point to the directory in which they are located.

Warning Keep copies of the `key3.db` and `cert7.db` files somewhere separate from the copies you use with the `signtool` executable. This ensures that you won't lose your certificates if you accidentally damage the files. §

Listing Available Certificates

You use the `-L` option to list the nicknames for all available certificates and check which ones are signing certificates, as shown in this Unix example:

```
% signtool -L
using certificate directory: /u/jsmith/.netscape
S Certificates
- ------------
  BBN Certificate Services CA Root 1
  IBM World Registry CA
  VeriSign Class 1 CA - Individual Subscriber - VeriSign, Inc.
  GTE CyberTrust Root CA
  Douglas J. Nicolson's Netscape Communications Corporation ID
  Uptime Group Plc. Class 4 CA
* Verisign Object Signing Cert
  Integrion CA
  GTE CyberTrust Secure Server CA
  AT&T Directory Services
* test object signing cert
  Uptime Group Plc. Class 1 CA
  VeriSign Class 1 Primary CA
- ------------
Certificates that can be used to sign objects have *'s to their left.
%
```

In the above example, two signing certificates are displayed: `Verisign Object Signing Cert` and `test object signing cert`.

You use the `-l` option to get a list of signing certificates only, including the signing CA for each, as shown in this Unix example:

```
% signtool -l
using certificate directory: /u/jsmith/.netscape
Object signing certificates
---------------------------------------
Verisign Object Signing Cert
    Issued by: VeriSign, Inc. - Verisign, Inc.
    Expires: Tue May 19, 1998
test object signing cert
    Issued by: test object signing cert (Signtool 1.0 Testing
Certificate (960187691))
    Expires: Sun May 17, 1998
---------------------------------------
For a list including CAs, use "signtool -L"
```

Signing a File

To sign a file using the Netscape Signing Tool, follow these steps:

1. **Create an empty directory.**

```
% mkdir signdir
```

2. **Put some file into it.**

```
% echo boo > signdir/test.f
```

3. **Specify the name of your object-signing certificate and sign the directory.**

   If you are using Unix, this example assumes you have put your `.db` files in the
   `~/.netscape` directory, as explained in Setting Up Your Certificate.

```
% signtool -k MySignCert -Z testjar.jar signdir

using key "MySignCert"
using certificate directory: /u/jsmith/.netscape
Generating signdir/META-INF/manifest.mf file..
--> test.f
adding signdir/test.f to testjar.jar
Generating signtool.sf file..
Enter Password or Pin for "Communicator Certificate DB":
```

4. **At the prompt, type the password to your private-key database.**

If it accepts the password, signtool responds as follows:

```
adding signdir/META-INF/manifest.mf to testjar.jar
adding signdir/META-INF/signtool.sf to testjar.jar
adding signdir/META-INF/signtool.rsa to testjar.jar
tree "signdir" signed successfully
```

5. **Test the archive you just created.**

```
% signtool -v testjar.jar

using certificate directory: /u/jsmith/.netscape
archive "testjar.jar" has passed crypto verification.

           status   path
     ------------   -------------------
         verified   test.f
```

You can also use the Netscape Signing Tool from within a script to automate some aspects of signing. For example, here's a Windows script that starts with an unsigned JAR file, unpackages it, signs it, and then repackages it:

```
rem Expand the jar file into a new directory
unzip -qq myjar.jar -d signjar
del myjar.jar
rem Sign everything in the new directory and recompress
signtool -k MySignCert -Z myjar.jar signdir
```

Using the Netscape Signing Tool with a ZIP Utility

To use the Netscape Signing Tool with a ZIP utility, you must have the utility in your path environment variable. You should use the `zip.exe` utility rather than `pkzip.exe`, which cannot handle long file names.

You can use a ZIP utility instead of the `-Z` option to package a signed archive into a JAR file after you have signed it:

```
% cd signdir
% zip -r ../myjar.jar *
  adding: META-INF/ (stored 0%)
  adding: META-INF/manifest.mf (deflated 15%)
  adding: META-INF/signtool.sf (deflated 28%)
  adding: META-INF/signtool.rsa (stored 0%)
  adding: text.txt (stored 0%)
%
```

Tips and Techniques

- If you are storing JAR files or their components in CVS, store them as binary files.
- If you are signing metadata only and not files, you still need to create a blank directory for the Netscape Signing Tool to sign.
- When using the Windows NT version of the Netscape Signing Tool, always use a relative path.
- The command `signtool -L` should list your object-signing certificate with an asterisk (*) beside it. If it doesn't, you cannot sign files.
- If you are having problems using the JAR file from within Navigator, check `signtool -v` on your final archive.
- Don't run the Netscape Signing Tool while Navigator is running.
- If you see the error `Unknown issuer`, you need to get the certificate of the certificate authority that issued your signing certificate. Alternatively, you may have the certificate authority's certificate, but it may not be trusted for object signing.
- If you see the error `Issuer not trusted`, open Communicator on the system you used when you obtained the certificate and follow these steps:

  1. Click the Security button in a Navigator window.
  2. Click Signers under Certificates in the left frame.
  3. Select the CA that issued your signing certificate.
  4. Click the Edit button.
  5. Select the "Accept this Certificate Authority for Certifying software developers" checkbox.
  6. Click OK.

  You then need to transfer the `cert7.db` file to the appropriate directory on the system on which you are running the Netscape Signing Tool, as described in Setting Up Your Certificate.

Chapter

3

SignTool Syntax and Options

This chapter summarizes the syntax and options for Netscape Signing Tool 1.0.

Sections in this chapter:

Syntax

To run Netscape Signing Tool 1.0, type

`signtool` *options*

where *options* can be any sequence of the options listed in this chapter.

Note Version 1.0 of the Netscape Signing Tool is compatible with earlier versions, including .50 and .60. If you rename the executable to `zigbert.exe`, you may continue to use the term `zigbert` rather than `signtool` with all the options listed here. This may be desirable to ensure compatibility with scripts written for earlier versions.

Each argument for each `signtool` option must be in quotes if it contains any spaces or other nonalphanumeric characters.

Options

Options for `signtool` are defined as follows:

`-b` *basename*	Specifies the base filename for the `.rsa` and `.sf` files in the `META-INF` directory (required by The JAR Format). For example, `-b signatures` causes the files to be named `signatures.rsa` and `signatures.sf`. The default is `signtool`. The `-b` option is available in Netscape Signing Tool 1.0 and later versions only.
`-c#`	Specifies the compression level for the `-J` or `-Z` option. The symbol # represents a number from 0 to 9, where 0 means no compression and 9 means maximum compression. The higher the level of compression, the smaller the output but the longer the operation takes. If the `-c#` option is not used with either the `-J` or the `-Z` option, the default compression value used by both the `-J` and `-Z` options is 6.
`-d` *certdir*	Specifies your certificate database directory; that is, the directory in which you placed your `key3.db` and `cert7.db` files. To specify the current directory, use "`-d.`" (including the period). The Unix version of `signtool` assumes `~/.netscape` unless told otherwise. The NT version of `signtool` always requires the use of the -d option to specify where the database files are located.
`-e` *extension*	Tells `signtool` to sign only files with the given extension; for example, use `-e".class"` to sign only Java class files.

`-G` *nickname*	Generates a new private-public key pair and corresponding object-signing certificate with the given nickname. The newly generated keys and certificate are installed into the key and certificate databases in the directory specified by the `-d` option. With the NT version of the Netscape Signing Tool, you must use the `-d` option with the `-G` option. With the Unix version of the Netscape Signing Tool, omitting the `-d` option causes the tool to install the keys and certificate in the Communicator key and certificate databases. If you are installing the keys and certificate in the Communicator databases, you must exit Communicator before using this option; otherwise, you risk corrupting the databases. In all cases, the certificate is also output to a file named `x509.cacert`, which has the MIME-type `application/x-x509-ca-cert`. Unlike certificates normally used to sign finished code to be distributed over a network, a test certificate created with `-G` is not signed by a recognized certificate authority. Instead, it is self-signed. In addition, a single test signing certificate functions as both an object-signing certificate and a CA. When you are using it to sign objects, it behaves like an object-signing certificate. When it is imported into browser software such as Communicator, it behaves like an object-signing CA and cannot be used to sign objects. The `-G` option is available in Netscape Signing Tool 1.0 and later versions only. In version 1.0, it produces only RSA certificates with 512-byte keys. For more information about the use of the `-G` option, see Chapter 4, "Generating Test Object-Signing Certificates."
`-i` *scriptname*	Specifies the name of an installer script for SmartUpdate. This script installs files from the JAR archive in the local system after SmartUpdate has validated the digital signature. For more details, see the description of `-m` that follows. The `-i` option provides a straightforward way to provide this information if you don't need to specify any metadata other than an installer script.

`-j` *directory*	Specifies a special JavaScript directory. This option causes the specified directory to be signed and tags its entries as inline JavaScript. This special type of entry does not have to appear in the JAR file itself. Instead, it is located in the HTML page containing the inline scripts. When you use `signtool -v`, these entries are displayed with the string `NOT PRESENT`.
`-J`	Signs a directory of HTML files containing JavaScript and creates as many archive files as are specified in the HTML tags. Even if `signtool` creates more than one archive file, you need to supply the key database password only once. The `-J` option is available in Netscape Signing Tool 1.0 and later versions only. The `-J` option cannot be used at the same time as the `-Z` option. If the `-c#` option is not used with the `-J` option, the default compression value is 6.
`-k` *key* `...` *directory*	Specifies the nickname (*key*) of the certificate you want to sign with and signs the files in the specified directory. The directory to sign is always specified as the last command-line argument. Thus, it is possible to write `signtool -k MyCert -d . signdir` You may have trouble if the nickname contains a single quotation mark. To avoid problems, escape the quotation mark using the escape conventions for your platform. It's also possible to use the `-k` option without signing any files or specifying a directory. For example, you can use it with the `-l` option to get detailed information about a particular signing certificate.
`-l`	Lists signing certificates, including issuing CAs. If any of your certificates are expired or invalid, the list will so specify. This option can be used with the -k option to list detailed information about a particular signing certificate. The `-l` option is available in Netscape Signing Tool 1.0 and later versions only.

`-L`	Lists the certificates in your database. An asterisk appears to the left of the nickname for any certificate that can be used to sign objects with `signtool`.

`-m` *metafile*	Specifies the name of a metadata control file. Metadata is signed information attached either to the JAR archive itself or to files within the archive. This metadata can be any ASCII string, but is used mainly for specifying an installer script.
	
	The metadata file contains one entry per line, each with three fields:
	
	field #1: file specification, or `+` if you want to specify global metadata (that is, metadata about the JAR archive itself or all entries in the archive)
	field #2: the name of the data you are specifying; for example: `Install-Scrip`t
	field #3: data corresponding to the name in field #2
	
	For example, the `-i` option uses the equivalent of this line:
	
	`+ Install-Script: script.js`
	
	This example associates a MIME type with a file:
	
	`movie.qt MIME-Type: video/quicktime`
	For information about the way installer script information appears in the manifest file for a JAR archive, see The JAR Format on Netscape DevEdge.

`-M`	Lists the PKCS #11 modules available to `signtool`, including smart cards. The `-M` option is available in Netscape Signing Tool 1.0 and later versions only. For information on using the Netscape Signing Tool with smart cards, see Chapter 6, “Using the Netscape Signing Tool with Smart Cards.” For information on using the `-M` option to verify FIPS-140-1 validated mode, see Chapter 7, “Netscape Signing Tool and FIPS-140-1.”

`-o`	Optimizes the archive for size. Use this *only* if you are signing very large archives containing hundreds of files. This option makes the manifest files (required by the JAR format) considerably smaller, but they contain slightly less information.
`-p` *password*	Specifies a password for the private-key database. Note that the password entered on the command line is displayed as plain text.
`-v` *archive*	Displays the contents of an archive and verifies the cryptographic integrity of the digital signatures it contains and the files with which they are associated. This includes checking that the certificate for the issuer of the object-signing certificate is listed in the certificate database, that the CA's digital signature on the object-signing certificate is valid, that the relevant certificates have not expired, and so on.
`-w` *archive*	Displays the names of signers of any files in the archive.
`-x` *directory*	Excludes the specified directory from signing.
`-z`	Tells `signtool` not to store the signing time in the digital signature. This option is useful if you want the expiration date of the signature checked against the current date and time rather than the time the files were signed.
`-Z` *jarfile*	Creates a JAR file with the specified name. You must specify this option if you want `signtool` to create the JAR file; it does not do so automatically. If you don't specify `-Z`, you must use an external ZIP tool to create the JAR file. The `-Z` option cannot be used at the same time as the `-J` option. If the `-c#` option is not used with the `-Z` option, the default compression value is 6.

Chapter

4

Generating Test Object-Signing Certificates

Netscape Signing Tool 1.0 allows you to create object-signing certificates for testing purposes. This chapter describes how to create and use such test certificates.

Sections in this chapter:

Unlike certificates normally used to sign finished code to be distributed over a network, the test certificates created with Netscape Signing Tool are not signed by a recognized certificate authority. Instead, they are self-signed. In addition, a single test signing certificate functions as both an object-signing certificate and a CA. When you are using it to sign objects, it behaves like an object-signing certificate. When it is imported into browser software such as Communicator, it behaves like an object-signing CA.

Generating the Keys and Certificate

The `signtool` option `-G` generates a new public-private key pair and certificate. It takes the nickname of the new certificate as an argument. The newly generated keys and certificate are installed into the key and certificate databases in the directory specified by the `-d` option. With the NT version of the Netscape Signing Tool, you must use the `-d` option with the `-G` option. With the Unix version of the Netscape Signing Tool, omitting the `-d` option causes the tool to install the keys and certificate in the Communicator key and certificate databases. In all cases, the certificate is also output to a file named `x509.cacert`, which has the MIME-type `application/x-x509-ca-cert`.

Warning If you intend to install the new key pair and certificate in the Communicator database, you must exit Communicator before using the Netscape Signing Tool to generate the object-signing certificate. Otherwise, you risk corrupting your certificate and key databases.

Certificates contain standard information about the entity they identify, such as the common name and organization name. The Netscape Signing Tool prompts you for this information when you run the command with the `-G` option. However, all of the requested fields are optional for test certificates. If you do not enter a common name, the tool provides a default name. In the following example, the user input is in boldface:

```
% signtool -G MyTestCert
using certificate directory: /u/someuser/.netscape
Enter certificate information. All fields are optional. Acceptable
characters are numbers, letters, spaces, and apostrophes.
certificate common name: Test Object Signing Certificate
organization: Netscape Communications Corp.
organization unit: Server Products Division
state or province: California
country (must be exactly 2 characters): US
username: someuser
email address: someuser@netscape.com
Enter Password or Pin for "Communicator Certificate DB": [Password will
not echo]
generated public/private key pair
certificate request generated
certificate has been signed
certificate "MyTestCert" added to database
Exported certificate to x509.raw and x509.cacert.
%
```

The certificate information is read from standard input. Therefore, the information can be read from a file using the redirection operator (<) in some operating systems. To create a file for this purpose, enter each of the seven input fields, in order, on a separate line. Make sure there is a newline character at the end of the last line. Then run `signtool` with standard input redirected from your file as follows:

```
% signtool -G MyTestCert <inputfile
```

The prompts show up on the screen, but the responses will be automatically read from the file. The password will still be read from the console unless you use the `-p` option to give the password on the command line.

Importing Your Test Certificate Into Communicator

It's possible to install the newly generated certificate and keys directly into the Communicator databases if you so specify with the `-d` option (or, with Unix versions of the Netscape Signing Tool only, if you omit the `-d` option). If you do install the certificate and keys in the Communicator database when you create your test certificate, the certificate will thenceforth be trusted as a CA.

If you do not install the certificate and keys in Communicator at the time you create them, or if you want to install them in additional copies of Communicator on other machines, you can use the `x509.cacert` file that `signtool` generates automatically when it creates the certificate. This file contains the certificate in base-64-encoded form. The file can be read into any copy of Communicator if the file is posted on a web page.

To make the `x509.cacert` file accessible from a web page, follow these steps:

1. **Create a link to the `x509.cacert` file in an HTML document. For example:**

   ```
   <a href="x509.cacert">Click Here to Import My Object-Signing Test
   Certificate</a>
   ```

2. **Make sure your web server exports the file as MIME-type `application/x-x509-ca-cert`.** Netscape Enterprise Server 3.0 is configured this way by default, and any server can be made to export the certificate correctly. To arrange this, you or your system administrator must associate this MIME-type with the file extension `.cacert`. Depending on your web server, this may involve editing a configuration file or using an administration tool.

After completing these steps, anyone who can access the web page can import your certificate by clicking the link. Doing so causes a security dialog box to appear that guides the user through the process of installing the certificate as a trusted CA. After completing the installation and accepting the certificate for certifying software developers (that is, as a CA for object-signing certificates), the user can run Java applets signed by your test certificate.

It is important to note that importing the certificate in this way does not enable anyone else to use your certificate for object-signing. Your private signing key is not stored in the `x509.cacert` file. Importing the certificate only allows other people to use software that you have signed.

Chapter

5

Signing Inline JavaScript Scripts

This chapter describes how to use the Netscape Signing Tool to sign inline JavaScript scripts and package the digital signature and related information in a JAR file.

Important These capabilities were available in version .60 of the Netscape Signing Tool only through the use of the Page Signer Perl script. Although version 1.0 still supports such Perl scripts, the capabilities of Page Signer are provided through the `-J` option for `signtool`, described in Chapter 2, "Using the Netscape Signing Tool." If you prefer to continue using the Page Signer Perl script to maintain compatibility with other scripts, you may do so, but Netscape recommends that you use the `signtool` options for new projects.

Sections in this chapter:

For more information about JavaScript and signed scripts, see JavaScript Security Model in Communicator 4.x on Netscape DevEdge.

Modifying the HTML Page

The `<SCRIPT>` tag for a signed script must include two attributes:

- An ARCHIVE attribute whose value is the name of the JAR file that contains the script's digital signature. If you do not include an ARCHIVE attribute, Communicator uses the ARCHIVE attribute from an earlier script on the same page.
- Either an ID attribute whose value is a string that associates the script with a digital signature in the JAR file, or an SRC attribute that retrieves a script from the JAR file.

If you use the SRC attribute, the script itself must be stored in the JAR file as well as the script's digital signature. If you keep all your scripts in separate files, you don't need to use the -J option; just use the -k option to sign the whole directory. If you use both inline scripts and separate scripts identified by the SRC attribute in the same page, the -J option automatically extracts and signs the inline scripts.

In addition to inline scripts and JavaScript files, you can sign event handler scripts and JavaScript entities. You cannot sign javascript: URLs. You must identify handler scripts and JavaScript entities within the HTML file as follows:

- To sign an event handler, you add an ID attribute for the event handler to the tag containing the event handler. In addition, the HTML page must also contain a signed inline script preceding the event handler. That <SCRIPT> tag must supply the ARCHIVE attribute.
- To sign a JavaScript entity, you do not do anything special to the entity. Instead, the HTML page must also contain a signed inline script preceding the JavaScript entity. That <SCRIPT> tag must supply the ARCHIVE and ID attributes.

For more details on signed JavaScript and examples, see JavaScript Security Model in Communicator 4.x on Netscape DevEdge.

To request access to specific targets, such as UniversalFileRead, you use the Java Capabilities API from within JavaScript. For more information about the Java Capabilities classes, see Java Capabilities API. For information about the system targets you can use this API to access, see Netscape System Targets. Both of these documents are on Netscape DevEdge.

Signing Scripts

This section demonstrates how to use the Netscape Signing Tool to sign a directory of HTML and JavaScript files. The HTML file contains an inline JavaScript script, as well as a reference to the JavaScript source file. The directory is named `signdir`. To begin with, this directory contains two files, `installation.js` and `test.html`.

This Unix example explores the contents of the `signdir` directory:

```
% cd signdir
% ls
installation.js  test.html
% cat test.html
<HTML>
<HEAD>
<SCRIPT ARCHIVE="handler.jar" ID=1>
function getsExpandedPrivileges() {
   // Request privilege
   netscape.security.PrivilegeManager.enablePrivilege
      ("UniversalBrowserRead");
      return history[0] != "";
}
</SCRIPT>
</HEAD>
<BODY onLoad="alert(getsExpandedPrivileges() ? 'Pass' : 'FAIL');" ID=2>
<SCRIPT SRC="installation.js"></SCRIPT>
</BODY>
</HTML>

% cat installation.js
function cAlert(x) {
      if(!this.silent) {
                alert(x);
      }
}
% cd ..
```

This command uses `signtool` to specify a signing certificate and to sign the contents of the `signdir` directory:

```
% signtool -k MySignCert -J signdir
using key "MySignCert"
using certificate directory: /u/jsmith/.netscape
Generating inline signatures from HTML files in: signdir
Processing HTML file: test.html
  entry: handler.arc/1
  (stashing a copy of installation.js -> handler.arc)
```

```
signing: signdir/handler.jar
Generating signdir/handler.arc/META-INF/manifest.mf file..
--> 1
adding signdir/handler.arc/1 to signdir/handler.jar
--> installation.js
adding signdir/handler.arc/installation.js to signdir/handler.jar
Generating signtool.sf file..
Enter Password or Pin for "Communicator Certificate DB":
adding signdir/handler.arc/META-INF/manifest.mf to signdir/handler.jar
adding signdir/handler.arc/META-INF/signtool.sf to signdir/handler.jar
adding signdir/handler.arc/META-INF/signtool.rsa to signdir/handler.jar
jarfile "signdir/handler.jar" signed successfully
```

The `ARCHIVE="handler.jar"` tag in the inline script causes the Netscape Signing Tool to create a new directory called `handler.arc`

```
function getsExpandedPrivileges() {
// Request privilege
netscape.security.PrivilegeManager.enablePrivilege
("UniversalBrowserRead");
return history[0] != "";
}
</SCRIPT>
```

- The file named `installation.js` was copied to the `handler.arc` directory from the `signdir` directory during the signing operation.
- The `META-INF` directory contains the digital signature files and related information organized according to the JAR Format. You don't need to concern yourself with the contents of this directory or the details of the JAR format to use the Netscape Signing Tool.

Note that the contents of the `handler.arc` directory are no longer needed, because they have been copied into the `handler.jar` file.

The `signtool` option `-v` verifies that the signing and archiving operations were successful:

```
% signtool -v handler.jar
using certificate directory: /u/jsmith/.netscape
archive "handler.jar" has passed crypto verification.

          status   path
    ------------   -------------------
        verified   1
        verified   installation.js
```

Chapter

6

Using the Netscape Signing Tool with Smart Cards

This chapter describes how to use smart cards from within the Netscape Signing Tool to digitally sign files.

Sections in this chapter:

What Is a Smart Card?

A **smart card** (sometimes called a **token**) is a credit-card-sized card, a key, or other easily removable device that can be used for cryptographic operations and for storing certificates. Smart cards are portable and must be physically inserted in an appropriate smart card reader attached to a computer for use with Communicator software running on that computer. Smart cards extend the private-key protection provided by Communicator, since private keys stored on the card require the card's presence as well as the password to the private-key database.

Navigator and the Netscape Signing Tool support PKCS #11, a cryptographic standard developed to support services provided by smart cards. Before purchasing a smart card for use with Communicator, you should ensure that your vendor provides a PKCS #11 driver that has been tested with Communicator on your platform. Tested brands include Litronic Netsign and Datakey's SignaSURE.

Setting Up a Smart Card

Connect the smart card reader according to the manufacturer instructions. You may need to reset the smart card to a default state using the manufacturer's configuration utility. Not all smart cards require this step.

Smart cards designed for use with Communicator come with a software driver that you should install in your computer according to the manufacturer's instructions. You can then add the driver (also called a **cryptographic module**) to Communicator as follows:

1. Make sure the smart card is inserted in the smart card reader.
2. Click the Security button near the top of a Navigator window.
3. Click Cryptographic Modules in the left frame.
4. Click the Add button.
5. Type an appropriate name for the module you want to add in the box labeled Security Module Name.
6. Type the name of the driver that was supplied with your smart card in the box labeled Security Module File. For Windows systems, this is a dynamic linked library (DLL). You don't have to type the entire path, but you may.
7. Click OK.
8. If Communicator asks for it, type the smart card password.
9. Select the module you've just installed and click the View/Edit button.
10. Make sure the displayed information is correct for the smart card you just installed.

11. Select the name of the smart card.

12. Click the More Info button and examine that information as well.

13. If the state of the smart card (shown near the bottom of the More Info window) is Not Logged In, click OK and then click the Login button. Otherwise, just click OK. (Logging in allows you to install your signing certificate on the smart card. The smart card doesn't have to be logged in within Communicator for you to use it with the Netscape Signing Tool.)

14. Click OK again.

After you have activated the smart card, use Communicator to visit the web site for the certificate authority (CA) you want to use and request a signing certificate.

When you submit your information to the certificate authority, Communicator asks you to select the card or database you wish to use to generate your private key. You should select the name of your smart card.

Your system then generates a public-private key pair and submits your request to the CA. When you receive the certificate, it is installed directly onto the card and travels with that smart card. However, you will be unable to use the certificate unless the smart card is inserted in the appropriate reader and you have entered its password correctly.

Using the -M Option to List Smart Cards

You can use the `-M` option to list the PKCS #11 modules, including smart cards, that are available to `signtool`:

```
% signtool -d "c:\netscape\users\jsmith" -M
using certificate directory: c:\netscape\users\<username>
Listing of PKCS11 modules
-----------------------------------------------
   1. Netscape Internal PKCS #11 Module
           (this module is internally loaded)
           slots: 2 slots attached
           status: loaded
     slot: Communicator Internal Cryptographic Services Version 4.0
    token: Communicator Generic Crypto Svcs
     slot: Communicator User Private Key and Certificate Services
    token: Communicator Certificate DB
```

```
  2. CryptOS
          (this is an external module)
DLL name: core32
   slots: 1 slots attached
  status: loaded
    slot: Litronic 210
   token:
  3. Datakey SignaSURE
          (this is an external module)
DLL name: dkck232.dll
   slots: 1 slots attached
  status: loaded
   slot: Datakey Reader
   token: <username>
  -----------------------------------------------
```

Using the Netscape Signing Tool and a Smart Card to Sign Files

Chapter 7

Netscape Signing Tool and FIPS-140-1

This chapter describes how to use the Netscape Signing Tool in FIPS-140-1 validated mode. FIPS 140-1 is a U.S. government standard for implementations of cryptographic modules--that is, hardware or software that encrypts and decrypts data or performs other cryptographic operations (such as creating or verifying digital signatures). Many products sold to the U.S. government must comply with one or more of the FIPS standards.

Sections in this chapter:

For general information on FIPS standards and Netscape FIPS-140-1 validation, see the FIPS 140-1 FAQ.

Using FIPS-140 Mode

Netscape Signing Tool is FIPS-140-1 validated when it uses the FIPS-validated Netscape cryptographic module. The FIPS module can be activated and deactivated from within Communicator. Communicator stores the module choice in the security module database (called `secmod.db` on Windows platforms and `secmodule.db` on Unix platforms). This database is stored in the same directory as your certificate database (`cert7.db`) and key database (`key3.db`), as indicated by the `-d` option of the Netscape Signing Tool.

Before using the Netscape Signing Tool in FIPS-validated mode, you must use Navigator to switch to FIPS mode. For information on how to do this, see Operating Netscape Navigator in FIPS PUB-140-1 Compliant Mode.

After switching the Navigator cryptographic module to FIPS mode, you have two choices:

- Use the same security module database from Netscape Signing Tool (by specifying the same directory with the `-d` option).
- Make a copy of Communicator's security module database and place it in Netscape Signing Tool's database directory.

Verifying FIPS Mode

Use the `-M` option to verify that you are using the FIPS-140-1 module.

This Unix example shows that Netscape Signing Tool is using a non-FIPS module:

```
% signtool -d "c:\netscape\users\jsmith" -M
using certificate directory: c:\netscape\users\jsmith
Listing of PKCS11 modules
-----------------------------------------------
  1. Netscape Internal PKCS #11 Module
          (this module is internally loaded)
          slots: 2 slots attached
          status: loaded
    slot: Communicator Internal Cryptographic Services Version 4.0
   token: Communicator Generic Crypto Svcs
    slot: Communicator User Private Key and Certificate Services
   token: Communicator Certificate DB
-----------------------------------------------
```

This Unix example shows that Netscape Signing Tool is using a FIPS-140-1 module:

```
% signtool -d "c:\netscape\users\jsmith" -M
using certificate directory: c:\netscape\users\jsmith
Enter Password or Pin for "Communicator Certificate DB": [password will
not echo]
```

```
Listing of PKCS11 modules
-----------------------------------------------
  1. Netscape Internal FIPS PKCS #11 Module
          (this module is internally loaded)
          slots: 1 slots attached
          status: loaded
    slot: Netscape Internal FIPS-140-1 Cryptographic Services
   token: Communicator Certificate DB
-----------------------------------------------
```