

PKCS#11 Wrapper – JSign Signing Integration Document

JSign is a versatile, platform-independent tool used for code signing of Windows executable files, installers, and scripts. Because it is written in Java, the same tool signs Windows artifacts from a Linux, macOS, or Windows machine.

JSign reads its signing key from a keystore, and by giving it a keystore type of PKCS11 together with Encryption Consulting's PKCS#11 wrapper configuration, that keystore becomes your HSM. The private key never leaves the HSM and every signing operation is recorded centrally.

Section 1 covers the steps common to every platform. After that, follow only the track for your operating system — Sections 2 and 3 for Linux (Ubuntu), Sections 4 and 5 for macOS, or Sections 6 and 7 for Windows.

Prerequisites: Access to the CodeSign Secure portal, administrative rights on the client machine, and a file to sign such as an executable, installer, or script.

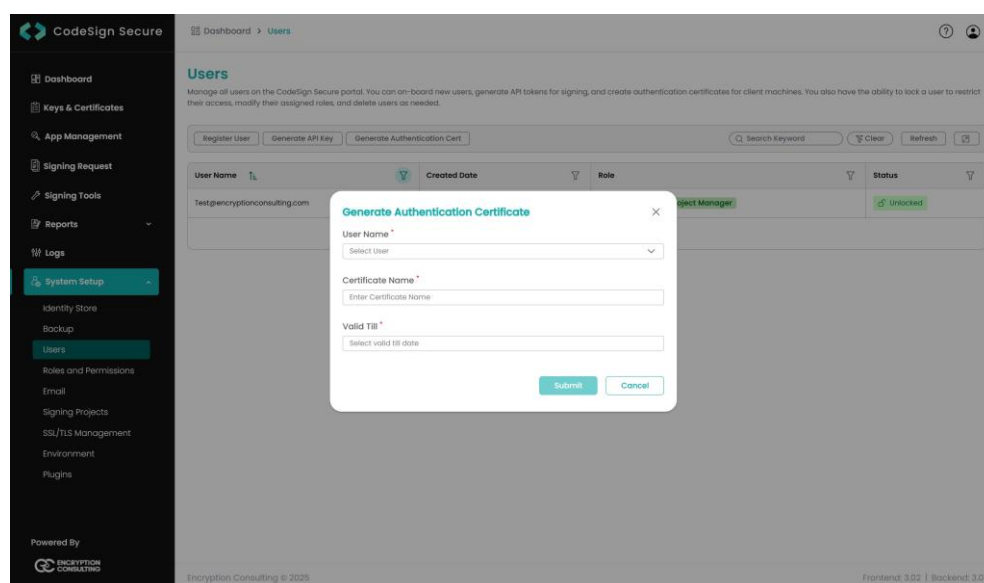
1. Common Setup

Every platform needs the same two things before JSign will work: an authentication certificate that identifies your machine to CodeSign Secure, and the two configuration files that ship inside the PKCS#11 Wrapper download.

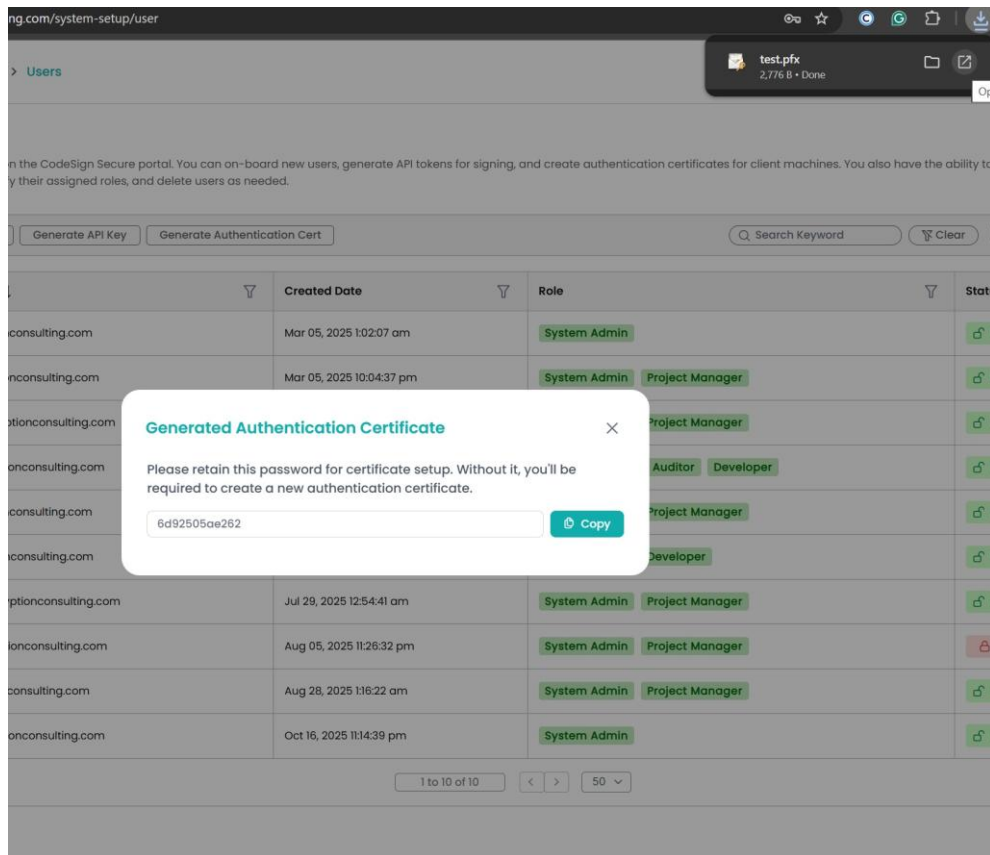
Steps:

Step 1: Generate a P12 Authentication Certificate

- Generate a P12 Authentication certificate from the System Setup > User > Generate Authentication Certificate dropdown in the CodeSign Secure portal.



- Enter the certificate name, user name, and expiry date. A certificate is generated and downloaded, and its password is displayed.



NOTE: The password is displayed only once, so copy and store it safely. Both the certificate path and its password are entered into `ec_pkcs11client.ini` in your platform track.

Step 2: Understand the Two Configuration Files

- The PKCS#11 Wrapper download contains two files that must be edited before use. The same fields apply on every platform.
 - pkcs11properties.cfg** – Describes the PKCS#11 token to JSig. Update it with the correct library path for your platform. This is also the file passed to JSig as `--keystore`.
 - ec_pkcs11client.ini** – Points the wrapper at your CodeSign Secure server and at the authentication certificate from Step 1. The fields to update are listed below.

Field	Description
name	The path location of the Log xml file you downloaded with the PKCS11 Wrapper tool (EC_PKCS11_CLIENT-LogConfig.xml).
url	Verify the server url of your CodeSign Secure portal.
username	The username of your CodeSign Secure account.

Field	Description
passcode	The secret passcode which was used to set up the CodeSign Secure portal.
path	The path location of your SSL Authentication certificate.
passwd	The password of your SSL Authentication certificate.

NOTE: When using Luna HSM, the slot number must also be added to the pkcs11properties.cfg file. Each platform track below shows this variation.

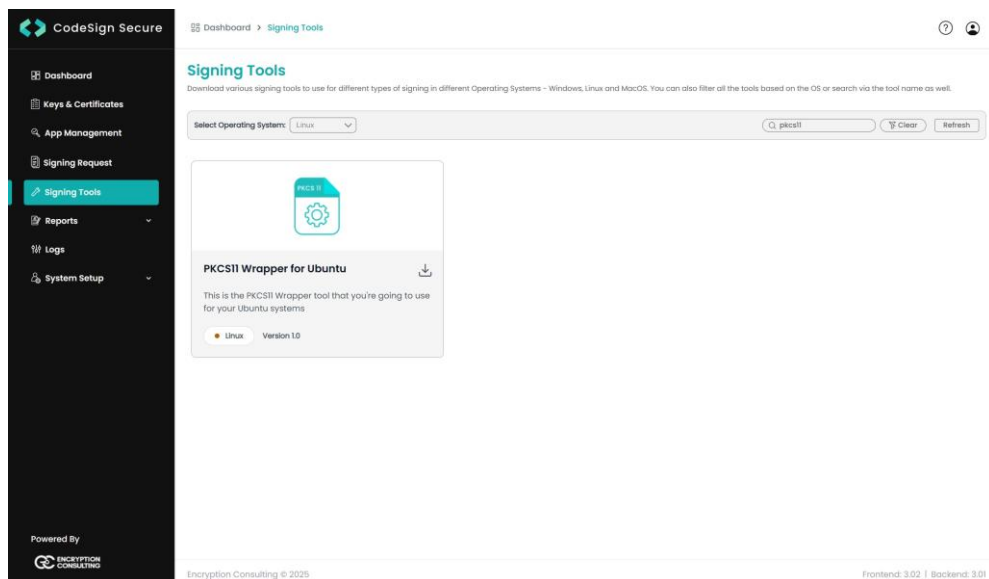
2. Linux (Ubuntu): Install and Configure

To perform signing using JSign Tool and our PKCS11 Wrapper in a Linux (Ubuntu) machine, you would need to follow the below steps.

Steps:

Step 1: Download the PKCS#11 Wrapper for Ubuntu

- Go to EC CodeSign Secure v3.02's Signing Tools section and download the PKCS11 Wrapper for Ubuntu.



Step 2: Edit the Configuration Files

- Go to your Ubuntu client system and edit the configuration files (ec_pkcs11client.ini and pkcs11properties.cfg) downloaded in the PKCS11 Wrapper.

```

aryan@test:~/PKCS_Wrapper_Ubuntu$ cat ec_pkcs11client.ini
[logconfig_file]
name=/home/aryan/PKCS_Wrapper_Ubuntu/EC_PKCS11_CLIENT-LogConfig.xml

[server_url]
url=https://devcodesignsecure.encryptionconsulting.com

[pfxfile_path]
path=/home/aryan/PKCS_Wrapper_Ubuntu/Pkcs11Test.pfx

[pfxfile_passwd]
passwd=de48b6924cb8

[login_token]
username=Aryan@encryptionconsulting.com
passcode=8@!$U3QXS5Xtu2vf
aryan@test:~/PKCS_Wrapper_Ubuntu$ cat pkcs11properties.cfg
name=signingmanager
library=/home/aryan/PKCS_Wrapper_Ubuntu/ec_pkcs11client.so
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}
aryan@test:~/PKCS_Wrapper_Ubuntu$ █

```

NOTE: When using Luna HSM, add the slot number in the pkcs11properties.cfg file as shown below.

```

aryan@test:~/PKCS_Wrapper_Ubuntu$ cat pkcs11properties.cfg
name=signingmanager
library=/home/aryan/PKCS_Wrapper_Ubuntu/ec_pkcs11client.so
slot=3
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}

```

Step 3: Set the EC_INI_FILE_PATH Environment Variable

- Now we need to add the environment variable for the pkcs11 client library.
- Run the below command to open the bashrc file:

```
nano ~/.bashrc
```

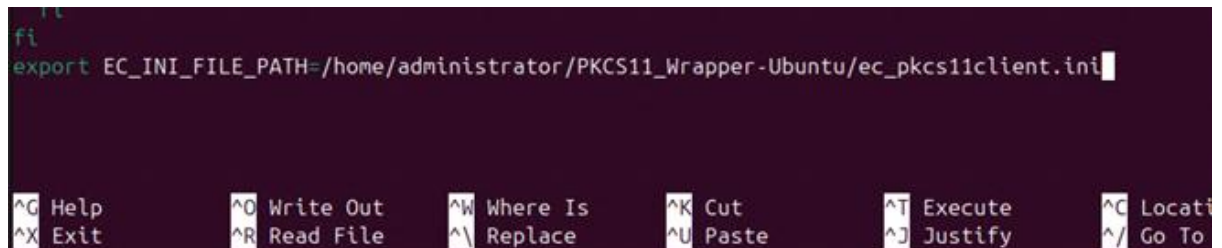
```

administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ nano ~/.bashrc
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ █

```

- Now add the EC_INI_FILE_PATH variable with the path of the .ini at the end of the bashrc file, like:

```
export EC_INI_FILE_PATH=\
/home/administrator/PKCS11_Wrapper-Ubuntu/ec_pkcs11client.ini
```



- Press Ctrl+X, then enter Y, and then press Enter to save.
- Reload the environment variables:

```
source ~/.bashrc
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ source ~/.bashrc
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$
```

- Check whether the variable has been set correctly:

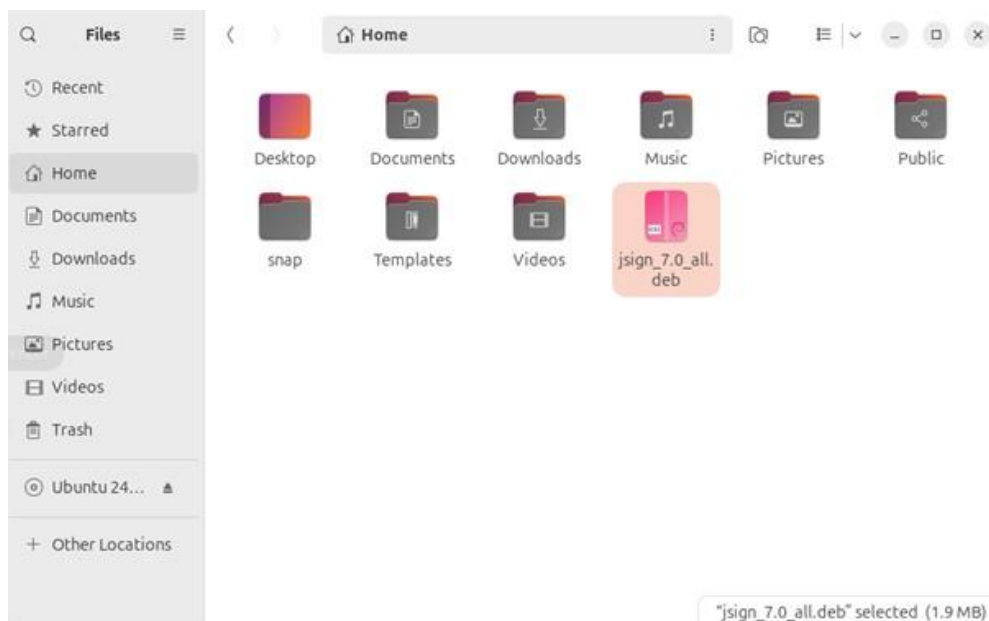
```
echo $EC_INI_FILE_PATH
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ echo $EC_INI_FILE_PATH
/home/administrator/PKCS11_Wrapper-Ubuntu/ec_pkcs11client.ini
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$
```

NOTE: If you cannot see the file location from the echo command, open a new terminal and try again.

Step 4: Install the JSign Tool

- Now, you will need to download the latest version of the JSign tool (DEB package) on your Ubuntu machine using this [link](#).



- Install the debian package using the below command:

```
sudo dpkg --install jsign_7.0_all.deb
```

```
aryan@JSIgn-new:~$ sudo dpkg --install jsign_7.0_all.deb
(Reading database ... 170459 files and directories currently installed.)
Preparing to unpack jsign_7.0_all.deb ...
Unpacking jsign (7.0) over (7.0) ...
Setting up jsign (7.0) ...
Processing triggers for man-db (2.12.0-4build2) ...
aryan@JSIgn-new:~$
```

- To check whether JSIgn has been properly installed or not, run the following command:

```
jsign
```



```

aryan@JSign-new:~$ jsign
usage: jsign [COMMAND] [OPTIONS] [FILE] [PATTERN] [@FILELIST]...

Sign and timestamp Windows executable files, Microsoft Installers (MSI), Cabinet
files (CAB), Catalog files (CAT), Windows packages (APPX/MSIX), Microsoft Dynamics
365 extension packages, NuGet packages and scripts (PowerShell, VBScript, JScript,
WSF)

commands: sign (default), timestamp, extract, remove, tag

sign:
  -s,--keystore <FILE>      The keystore file, the SunPKCS11 configuration file, the
                             cloud keystore name, or the card/token name
  --storepass <PASSWORD>    The password to open the keystore
  --storetype <TYPE>        The type of the keystore
                             File based:
                             - JKS: Java keystore (.jks files)
                             - JCEKS: SunJCE keystore (.jceks files)
                             PKCS12: Standard PKCS#12 keystore (.p12 or .pfx files)

```

Step 5: Install and Activate Java 17

- You will also need to install Java (Java 17 or lower) on your Ubuntu machine for JSign to work with our PKCS11 Wrapper.
- Run the following command to install Java 17 on your Ubuntu machine:

```
sudo apt install openjdk-17-jdk
```

```

aryan@ Sign-new:~$ sudo apt install openjdk-17-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm17t64
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  fonts-dejavu-extra libatk-wrapper-java libatk-wrapper-java-jni libice-dev
  libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev
  openjdk-17-jdk-headless openjdk-17-jre openjdk-17-jre-headless x11proto-dev
  xorg-sgml-doctools xtrans-dev

```

- To use Java 17, you will need to set it as the active version.

```
sudo update-alternatives --config java
```

```

aryan@ Sign-new:~$ sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                                    Priority    Status
  -----
* 0            /usr/lib/jvm/java-21-openjdk-amd64/bin/java            2111      auto mode
  1            /usr/lib/jvm/java-17-openjdk-amd64/bin/java            1711      manual mode
  2            /usr/lib/jvm/java-21-openjdk-amd64/bin/java            2111      manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/java to provide /usr/bin/java
(java) in manual mode
aryan@JSign-new:~$ S

```

- Now to check whether Java has been installed properly or not, run:

```
java -version
```

```
aryan@ Sign-new:~$ java -version
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu124.04, mixed mode, sharing)
aryan@ Sign-new:~$
```

Step 6: Install the Remaining Packages

- You would also need to install some other pre-requisite packages to perform signing using JSign.

```
sudo apt-get install curl
sudo apt-get install liblog4cxx-dev
sudo apt-get install liblog4cxx12
```

NOTE: If the “sudo apt-get install liblog4cxx12” command gives an error, install the package directly for your architecture using the commands below.

- For amd architecture:

```
wget http://archive.ubuntu.com/ubuntu/pool/universe/l/log4cxx/\
liblog4cxx12_0.12.1-4_amd64.deb
sudo dpkg -i liblog4cxx12_0.12.1-4_amd64.deb
```

- For arm architecture:

```
wget http://ports.ubuntu.com/pool/universe/l/log4cxx/\
liblog4cxx12_0.12.1-4_arm64.deb
sudo dpkg -i liblog4cxx12_0.12.1-4_arm64.deb
```

- We are now ready with our Ubuntu environment to perform the signing using JSign tool and PKCS11 Wrapper.

3. Linux (Ubuntu): Sign the File

The signing command must be run from the directory containing your configuration, so that the relative paths in the command resolve correctly.

Steps:

Step 1: Change to the Working Directory

- For signing, change the working directory of the terminal to that folder which contains your “ec_pkcs11client.ini” file.

```

aryan@JSign-new:~$ cd PKCS_Wrapper_Ubuntu/
aryan@JSign-new:~/PKCS_Wrapper_Ubuntu$ ls
ec_pkcs11client.ini      ec_pkcs11client.so      Pkcs11Test.pfx.pfx
EC_PKCS11_CLIENT-LogConfig.xml  pkcs11properties.cfg
aryan@JSign-new:~/PKCS_Wrapper_Ubuntu$

```

Step 2: Run the Signing Command

- Now, run the signing command from this directory.

```

<Path of JSig tool> --keystore <Path of pkcs11properties.cfg> \
--storepass NONE \
--storetype PKCS11 \
--alias <Key alias of the signing certificate> \
<Path of the file to be signed>

```

- A sample command is provided below:

```

jsign --keystore pkcs11properties.cfg \
--storepass NONE \
--storetype PKCS11 \
--alias gpg2 \
build_project.ps1

```

```

aryan@JSign-new:~/PKCS_Wrapper_Ubuntu$ jsign --keystore pkcs11properties.cfg --storepass NONE --storetype
PKCS11 --alias gpg2 build_project.ps1
aryan@JSign-new:~/PKCS_Wrapper_Ubuntu$

```

NOTE: On Ubuntu the DEB package installs a jsign launcher on the PATH, so the command begins with jsign rather than with java -jar.

Step 3: Option Reference

- The following are the options and their meaning in the command:

Option	Description
--keystore	The path of the pkcs11properties.cfg file, which directs JSig to the PKCS#11 wrapper instead of to a keystore file.
--storepass	The keystore password. This is NONE because access is controlled by the wrapper configuration.

Option	Description
<code>--storetype</code>	The keystore type, which must be PKCS11.
<code>--alias</code>	The key alias of the signing certificate in CodeSign Secure.
<code><file path></code>	The file to be signed, given as the final argument with no option name.

Step 4: Confirm the Signing Request

NOTE: There is no verification command for the JSign tool.

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the wrapper is recorded for audit purposes. Confirm that a signing request appears for the certificate you used.

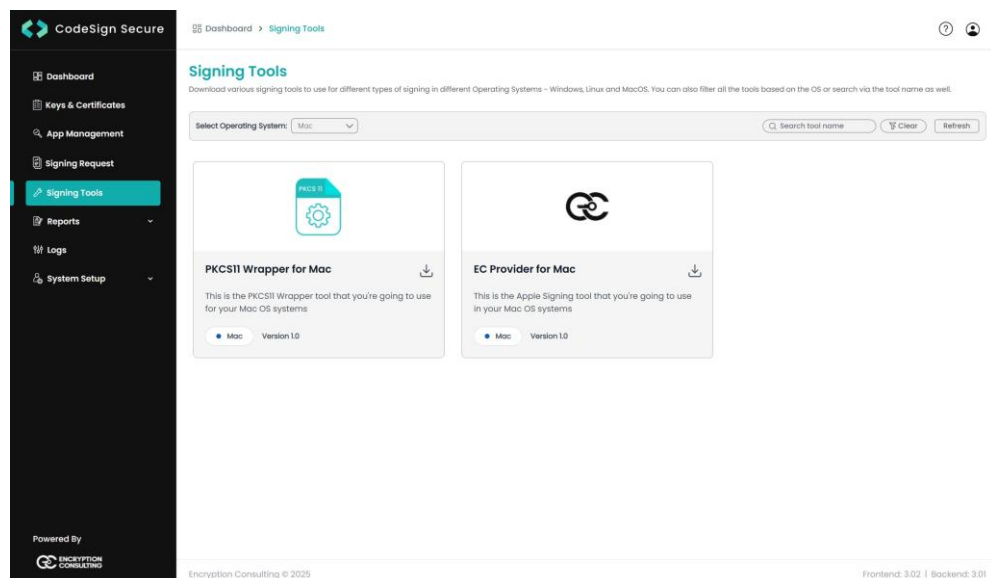
4. macOS: Install and Configure

To perform signing using JSign Tool and our PKCS11 Wrapper in a MacOS machine, you would need to follow the below steps.

Steps:

Step 1: Download the PKCS#11 Wrapper

- Download our PKCS11 Wrapper Tool from the CodeSign Secure portal in the Signing Tools section in your MacOS.



Step 2: Update the Configuration Files

- Go to your PKCS11 Wrapper directory (../PKCS11_Wrapper-Mac) and update the “pkcs11properties.cfg” and “ec_pkcs11client.ini” with the required details.
- For “pkcs11properties.cfg”, update the library path.

NOTE: When using Luna HSM, add the slot number in the pkcs11properties.cfg file as shown below.

```
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % cat pkcs11properties.cfg
name=signingmanager
library=/Users/subhayuroy/PKCS11_Wrapper-Mac/ec_pkcs11client.dylib
slot=3
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %
```

- For the “ec_pkcs11client.ini” file, update the six fields listed in Section 1, Step 2.
- The initialization and configuration files should look like this with all the correct file paths and settings.

```
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % cat ec_pkcs11client.ini
[logconfig_file]
name=/Users/subhayuroy/PKCS11_Wrapper-Mac/EC_PKCS11_CLIENT-LogConfig.xml

[server_url]
;url=https://lunacodesignsecure.encryptionconsulting.com
url=https://devcodesignsecure.encryptionconsulting.com
;url=http://127.0.0.1:6666

[pfxfile_path]
path=/Users/subhayuroy/PKCS11_Wrapper-Mac/AuthCert-nCipher.pfx

[pfxfile_passwd]
passwd=f4550251e262

[login_token]
username=Sam@encryptionconsulting.com
passcode=80!$U3QXS5Xtu2vf
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % cat pkcs11properties.cfg
name=signingmanager
library=/Users/subhayuroy/PKCS11_Wrapper-Mac/ec_pkcs11client.dylib
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %
```

Step 3: Install and Activate Java 17

- You will also need to install Java (Java 17 or lower) on your MacOS machine for JSig to work with our PKCS11 Wrapper.
- To install Java 17 on your MacOS machine, run the following command:

```
brew install openjdk@17
```

- Find the location where Java 17 is installed on your machine:

```
brew info openjdk@17
```

- Now set Java 17 as the active version. For Zsh:

```
nano ~/.zshrc
```

- For Bash:

```
nano ~/.bash_profile
```

- After running the above command, add these lines:

```
export PATH=<Path of Java 17 bin folder>:$PATH  
export JAVA_HOME=<Path of Java 17 bin folder>
```



```
pkcs11-client-Development — nano ~/.bash_profile — 121x34  
UW PICO 5.09 File: /Users/puneet/.bash_profile  
export PATH=/usr/local/Cellar/openjdk@17/17.0.14/bin:$PATH  
export JAVA_HOME=/usr/local/Cellar/openjdk@17/17.0.14/bin
```

NOTE: JSign is launched with `java -jar`, which resolves through `PATH`, so this setting is sufficient for signing. Note that `JAVA_HOME` is conventionally set to the JDK home directory rather than to its bin subfolder, so if other Java tooling on the machine relies on `JAVA_HOME` you may wish to point it one level higher.

- Reload the environment variables using the below commands. For Zsh:

```
source ~/.zshrc
```

- For Bash:

```
source ~/.bash_profile
```

Step 4: Install the Remaining Packages

- You would also need to install some other pre-requisite packages to perform signing using JSign.

```
brew install log4cxx
brew install curl
```

Step 5: Download the JSign Tool

- Now, you will need to download the latest version of the JSign tool (JAR package) on your MacOS machine using this [link](#).



- To check whether JSign has been properly installed or not, run the following command:

```
java -jar <Path of JSign Jar Package>
```

- We are now ready with our MacOS environment to perform the signing using JSign tool and PKCS11 Wrapper.

5. macOS: Sign the File

As on Linux, the signing command is run from the directory containing your configuration files. On macOS the JAR is invoked through java rather than through an installed launcher.

Steps:

Step 1: Change to the Working Directory

- For signing, change the working directory of the terminal to that folder which contains your “ec_pkcs11client.ini” file.

Step 2: Run the Signing Command

- Now, run the signing command from this directory.

```
java -jar <Path of JSigntool> \
  --keystore <Path of pkcs11properties.cfg> \
  --storepass NONE \
  --storetype PKCS11 \
  --alias <Key alias of the signing certificate> \
  <Path of the file to be signed>
```

- A sample command is provided below:

```
java -jar jsigntool-7.0.jar \
  --keystore pkcs11properties.cfg \
  --storepass NONE \
  --storetype PKCS11 \
  --alias gpg2 \
  build_project.ps1
```



Step 3: Option Reference

- The following are the options and their meaning in the command:

Option	Description
--keystore	The path of the pkcs11properties.cfg file, which directs JSigntool to the PKCS#11 wrapper instead of to a keystore file.
--storepass	The keystore password. This is NONE because access is controlled by the wrapper configuration.
--storetype	The keystore type, which must be PKCS11.
--alias	The key alias of the signing certificate in CodeSign Secure.
<file path>	The file to be signed, given as the final argument with no option name.

Step 4: Confirm the Signing Request

NOTE: There is no verification command for the JSigntool.

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the wrapper is recorded for audit purposes.

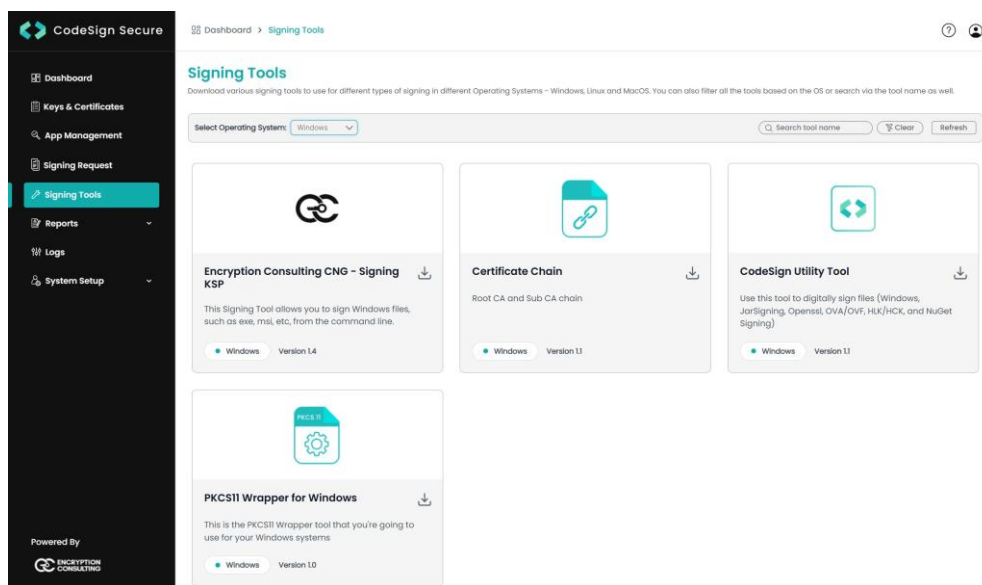
6. Windows: Install and Configure

To perform signing using JSign Tool and our PKCS11 Wrapper in a Windows machine, you would need to follow the below steps.

Steps:

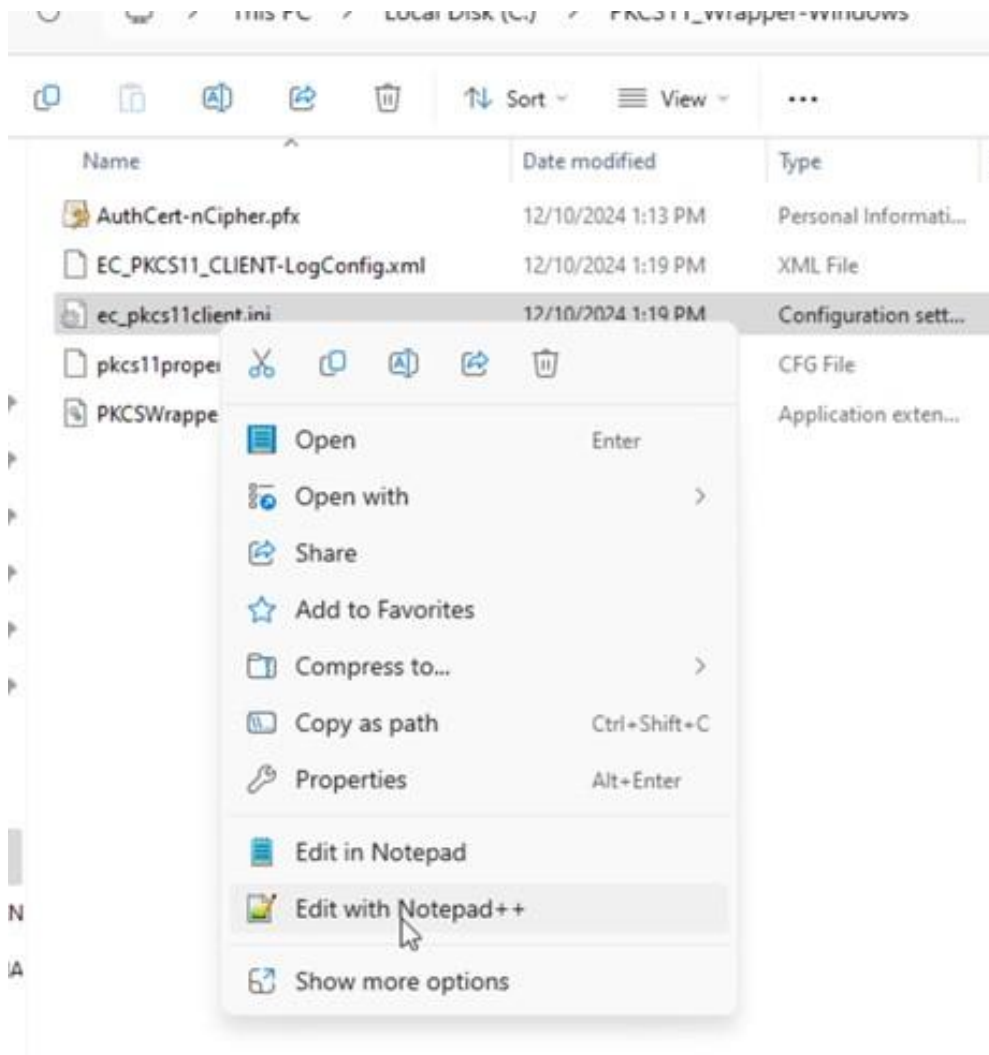
Step 1: Download the PKCS#11 Wrapper for Windows

- Go to EC CodeSign Secure v3.02's Signing Tools section and download the PKCS11 Wrapper for Windows.

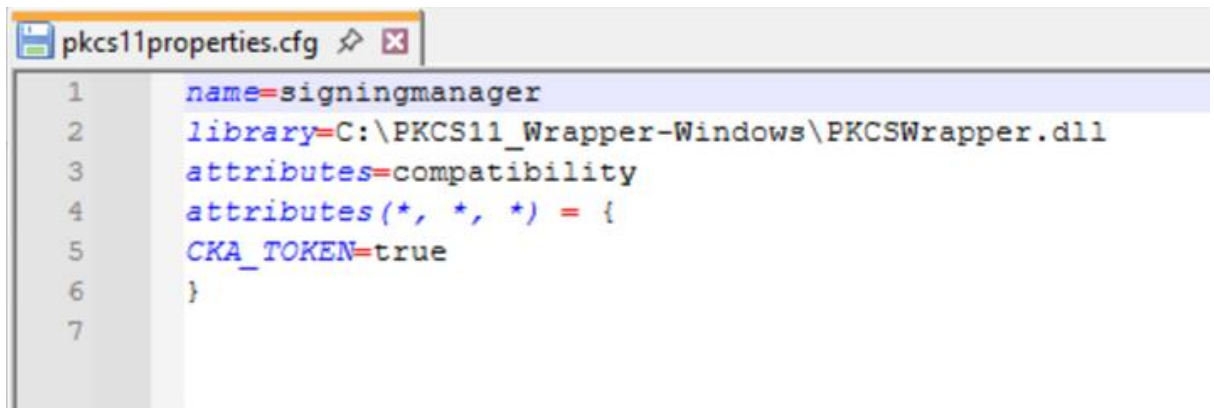


Step 2: Edit the Configuration Files

- Go to your Windows client system and edit the configuration files (ec_pkcs11client.ini and pkcs11properties.cfg) downloaded in the PKCS11 Wrapper.

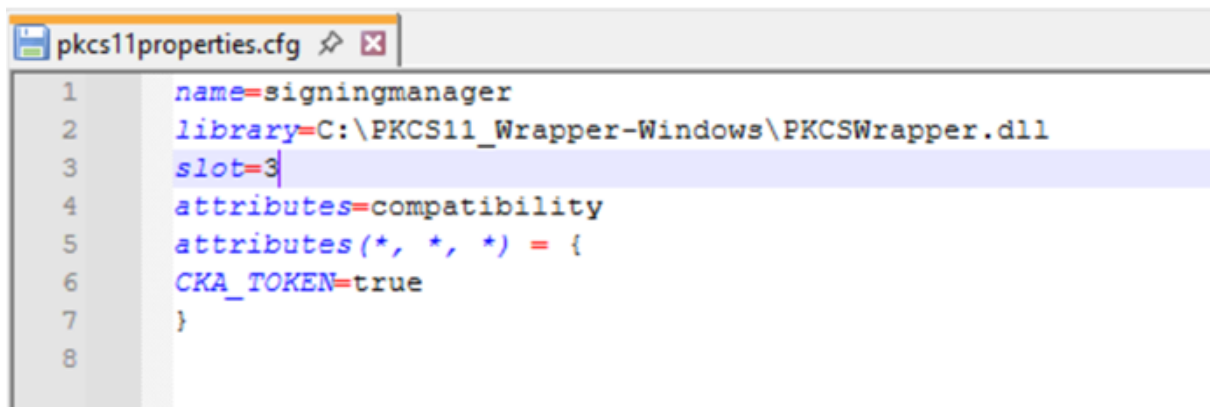


```
ec_pkcs11client.ini
1  [logconfig_file]
2  name=C:\PKCS11_Wrapper-Windows\EC_PKCS11_CLIENT-LogConfig.xml
3
4  [server_url]
5  url=https://devcodesignsecure.encryptionconsulting.com
6
7  [pfxfile_path]
8  path=C:\PKCS11_Wrapper-Windows\Pkcs11Test.pfx
9
10 [pfxfile_passwd]
11 passwd=bb881220628e
12
13 [login_token]
14 username=Aryan@encryptionconsulting.com
15 passcode=8@!$U3QXS5Xtu2vf
16
17
```



```
pkcs11properties.cfg
1  name=signingmanager
2  library=C:\PKCS11_Wrapper-Windows\PKCSWrapper.dll
3  attributes=compatibility
4  attributes(*, *, *) = {
5  CKA_TOKEN=true
6  }
7
```

NOTE: When using Luna HSM, add the slot number in the pkcs11properties.cfg file as shown below.



```
pkcs11properties.cfg
1  name=signingmanager
2  library=C:\PKCS11_Wrapper-Windows\PKCSWrapper.dll
3  slot=3
4  attributes=compatibility
5  attributes(*, *, *) = {
6  CKA_TOKEN=true
7  }
8
```

Step 3: Install and Activate Java 22

- Now, let us install some prerequisites in your client system to run the PKCS11 Wrapper.
- Install Java 22 from [Oracle's official site](#), and follow the instructions in the msi file.

Oracle Java SE Development Kit 22.0.2 archive downloads

Platform	Size	Download Link
Linux x64 RPM Package	185.89 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_linux-x64_bin.rpm
macOS Arm 64 Compressed Archive	179.81 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-arm64_bin.tar.gz
macOS Arm 64 DMG Installer	179.27 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-arm64_bin.dmg
macOS x64 Compressed Archive	181.99 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-x64_bin.tar.gz
macOS x64 DMG Installer	181.43 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-x64_bin.dmg
Windows x64 Compressed Archive	184.16 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_windows-x64_bin.zip
Windows x64 Installer	164.35 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_windows-x64_bin.exe
Windows x64 MSI Installer	165.09 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_windows-x64_bin.msi

Java SE Development Kit 22.0.1
This software is licensed under the Oracle No-Fee Terms and Conditions License.

Java(TM) SE Development Kit 22.0.2 (64-bit) - Setup

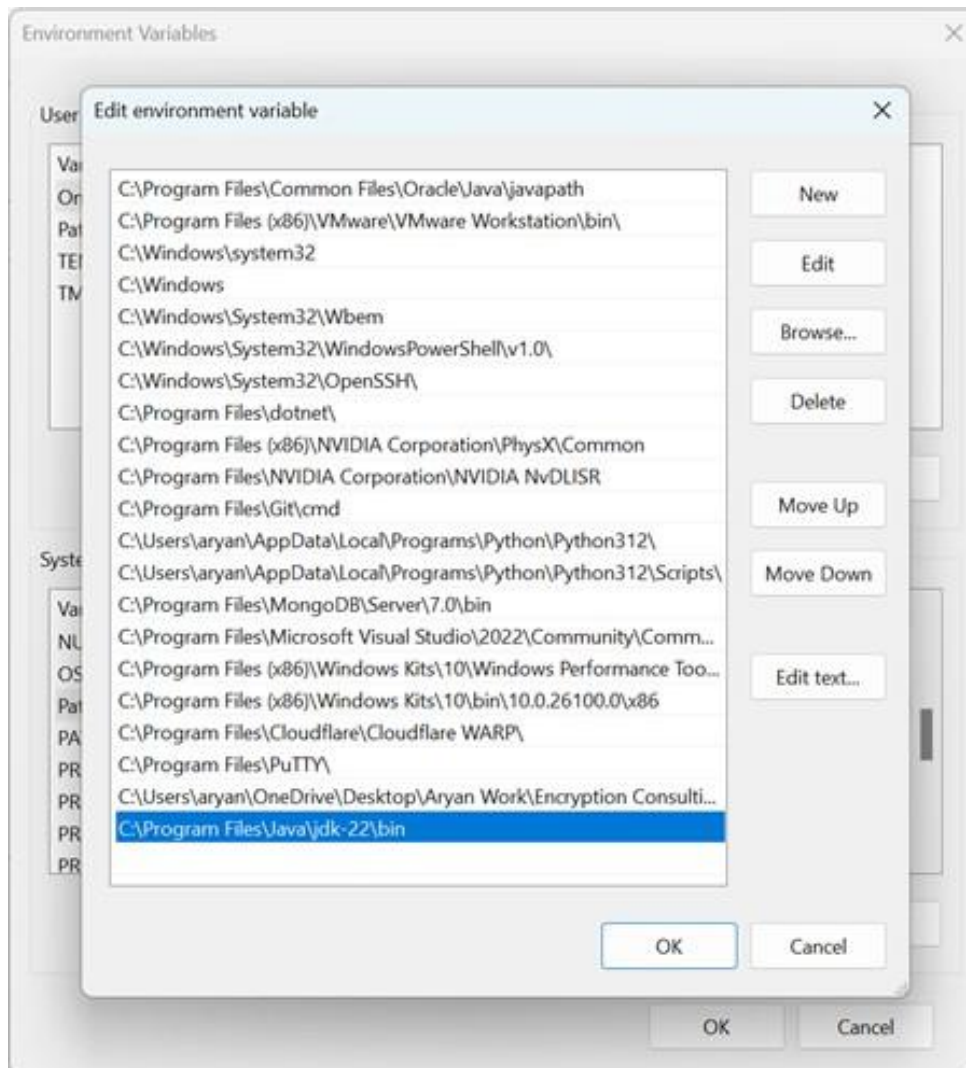
Welcome to the Installation Wizard for Java SE Development Kit 22.0.2

This wizard will guide you through the installation process for the Java SE Development Kit 22.0.2.

Next > Cancel



- Set Java 22 as the active version by storing the bin path in the PATH variable.



NOTE: The Linux and macOS tracks call for Java 17 or lower, while this track installs Java 22. Confirm which version your JSign release supports before deviating from the version shown for your platform.

Step 4: Download the JSign Tool

- Now, you will need to download the latest version of the JSign tool (JAR package) on your Windows machine using this [link](#).



- To check whether JSign has been properly installed or not, run the following command:

```
java -jar <Path of JSign Jar Package>
```

- We are now ready with our Windows environment to perform the signing using JSign tool and PKCS11 Wrapper.

7. Windows: Sign the File

As on the other platforms, the signing command is run from the directory containing your configuration files.

Steps:

Step 1: Change to the Working Directory

- For signing, change the working directory of the terminal to that folder which contains your “ec_pkcs11client.ini” file.

```
C:\>cd PKCS11_Wrapper-Windows

C:\PKCS11_Wrapper-Windows>dir
Volume in drive C has no label.
Volume Serial Number is 865B-2F9D

Directory of C:\PKCS11_Wrapper-Windows

02/06/2025  11:48 AM    <DIR>          .
02/06/2025  11:56 AM               6,122 build_project.ps1
02/04/2025  06:00 PM               248 ec_pkcs11client.ini
02/04/2025  05:36 PM               525 EC_PKCS11_CLIENT-LogConfig.xml
02/06/2025  11:56 AM          843,332 ec_pkcs11_client.log
02/04/2025  05:36 PM      2,104,682 jsign-7.0.jar
02/04/2025  06:08 PM               136 pkcs11properties.cfg
02/04/2025  05:39 PM               2,800 Pkcs11Test.pfx
02/06/2025  11:47 AM      2,285,056 PKCSWrapper.dll
               8 File(s)      5,242,901 bytes
               1 Dir(s)      384,016,384 bytes free
```

Step 2: Run the Signing Command

- Now, run the signing command from this directory.

```
java -jar <Path of JSign tool> ^
  --keystore <Path of pkcs11properties.cfg> ^
  --storepass NONE ^
  --storetype PKCS11 ^
  --alias <Key alias of the signing certificate> ^
  <Path of the file to be signed>
```

- A sample command is provided below:

```
java -jar jsign-7.0.jar ^
  --keystore pkcs11properties.cfg ^
```



```
--storepass NONE ^  
--storetype PKCS11 ^  
--alias gpg2 ^  
build_project.ps1
```

```
C:\PKCS11_Wrapper-Windows>java -jar jsign-7.0.jar --keystore pkcs11properties.cfg --storepass NONE --storetype PKCS11 --alias gpg2 build_project.ps1  
---Initializing curl-----slot description-----  
---api name ---C_OpenSession  
-----Session Handle----- 2257  
----- utc time -----2257  
Mechanism: CKM_SHA3_224_RSA_PKCS_PSS not found in mapping.  
Mechanism: CKM_SHA3_256_RSA_PKCS_PSS not found in mapping.  
Mechanism: CKM_SHA3_384_RSA_PKCS_PSS not found in mapping.  
Mechanism: CKM_SHA3_512_RSA_PKCS_PSS not found in mapping.  
Mechanism: CKM_SHA3_224_RSA_PKCS not found in mapping.  
Mechanism: CKM_SHA3_256_RSA_PKCS not found in mapping.  
Mechanism: CKM_SHA3_384_RSA_PKCS not found in mapping.  
Mechanism: CKM_SHA3_512_RSA_PKCS not found in mapping.  
Mechanism: CKM_EC_MONTGOMERY_KEY_PAIR_GEN not found in mapping.  
Mechanism: CKM_EC_EDWARDS_KEY_PAIR_GEN not found in mapping.
```

NOTE: The caret characters are Command Prompt line continuations. You may keep them or join the command onto a single line.

Step 3: Option Reference

- The following are the options and their meaning in the command:

Option	Description
--keystore	The path of the pkcs11properties.cfg file, which directs JSign to the PKCS#11 wrapper instead of to a keystore file.
--storepass	The keystore password. This is NONE because access is controlled by the wrapper configuration.
--storetype	The keystore type, which must be PKCS11.
--alias	The key alias of the signing certificate in CodeSign Secure.
<file path>	The file to be signed, given as the final argument with no option name.

Step 4: Confirm the Signing Request

NOTE: There is no verification command for the JSign tool.

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the wrapper is recorded for audit purposes.