THALES

Java Code Signing

INTEGRATION GUIDE

THALES LUNA HSM

THALES DATA PROTECTION ON DEMAND

Document Information

Document Part Number	007-000083-001
Revision	C
Release Date	5 October 2020

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

You can use Java Code Signing to sign Java applications for desktops, digitally sign .jar files, and enable Netscape Object Signing recognized by Java Runtime Environment (JRE). In Java, the process for setting up the Code Signing Certificate consists of creating a keystore and a Certificate Signing Request (CSR) and then, installing the code signing certificate on the keystore from where the CSR was generated. The Java platform enables one to digitally sign .jar files. The signer signs the .jar file using a private key. The corresponding public key is placed in the .jar file together with its certificate, so that it is available for use by anyone who wants to verify the signature. When the .jar file is signed, the user can timestamp the signature.

This guide demonstrates how to complete Java Code Signing using a signing key generated on a Luna HSM or an HSM on Demand service. Using a Luna HSM or an HSM on Demand service to generate the RSA keys for Java code signing provides the following benefits:

- Secure generation, storage, and protection of the signing private keys on FIPS 140-2 level 3 validated hardware.
- Full life cycle management of the keys.
- Access to the HSM audit trail*.
- Significant performance improvements by off-loading cryptographic operations from signing servers.

*HSMoD services do not have access to the secure audit trail

Supported Platforms

Thales Luna HSM: Thales Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Thales Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. The Luna HSM on premise offerings include the Luna Network HSM, PCIe HSM, and Luna USB HSMs. The following platforms are supported:

- Windows
- Linux

Thales DPoD: Thales Data Protection on Demand (DPoD) is a cloud-based platform that provides on-demand HSM and Key Management services through a simple graphical user interface. With DPoD, security is simple, cost effective and easy to manage because there is no hardware to buy, deploy, and maintain. As an Application Owner, you click and deploy services, generate usage reports, and maintain just the services you need. The following platforms are supported:

- Windows
- Linux

Prerequisites

Complete the following tasks before you begin the installation:

Configure the Luna HSM

Set up and configure the Luna HSM device for your system. Refer to the *Thales Luna HSM Product Documentation* for help.

1. Ensure the HSM is setup, initialized, provisioned, and ready for deployment.
2. Create a partition on the HSM for use by Java Code Signing.
3. If you are using a Thales Luna Network HSM, register a client for the system and assign the client to a partition to create an NTLS connection.
4. Initialize the Crypto Officer and Crypto User roles for the initialized partition.
5. Verify that the partition is successfully registered and configured. The command to see the registered partition is:

```
lunacm.exe (64-bit) v10.2.0-111. Copyright (c) 2020 SafeNet. All rights
reserved.
Available HSMs:
Slot Id ->              0
Label ->                JCS
Serial Number ->        1280780175877
Model ->                LunaSA 7.3.0
Firmware Version ->     7.3.0
Configuration ->        Luna User Partition With SO (PW) Key Export With
Cloning Mode
Slot Description ->     Net Token Slot
FM HW Status ->         FM Ready
Current Slot Id: 0
```

> **NOTE:** Refer to the *Thales Luna Network HSM Product Documentation* for detailed steps for creating NTLS connection, initializing the partition, and initializing the user roles.

Controlling User Access to the HSM

> **NOTE:** This section is applicable only for Linux users.

By default, only the root user has access to the HSM. You can specify a set of non-root users that are permitted to access the HSM by adding them to the **hsmusers** group. The client software installation automatically creates the **hsmusers** group. The **hsmusers** group is retained when you uninstall the client software, allowing you to upgrade the software while retaining your **hsmusers** group configuration.

Adding a user to hsmusers group

To allow non-root users or applications access to the HSM, assign the users to the **hsmusers** group. The users you assign to the **hsmusers** group must exist on the client workstation.

1. Ensure that you have **sudo** privileges on the client workstation.
2. Add a user to the hsmusers group.

   ```
   # sudo gpasswd --add <username> hsmusers
   ```

   Where `<username>` is the name of the user you want to add to the hsmusers group.

Removing a user from hsmusers group

1. Ensure that you have sudo privileges on the client workstation.
2. Remove a user from the hsmusers group.

   ```
   # sudo gpasswd -d <username> hsmusers
   ```

   Where `<username>` is the name of the user you want to remove from the **hsmusers** group. You must log in again to see the change.

   NOTE: The user you delete will continue to have access to the HSM until you reboot the client workstation.

Setting up Thales Luna HSM High-Availability (HA)

Refer to the *Thales Luna Network HSM Product Documentation* for HA steps and details regarding configuring and setting up two or more HSM appliances on Windows and UNIX systems. You must enable the HAOnly setting in HA for failover to work so that if primary stop functioning for some reason, all calls automatically routed to secondary till primary starts functioning again.

Provision HSM on Demand Service

This service enables your client machine to access an HSM application partition for storing cryptographic objects used by your applications. Application partitions can be assigned to a single client, or multiple clients can be assigned to, and share, a single application partition. You need to provision your application partition by initializing the following roles:

- **Security Officer (SO)** - Responsible for setting the partition policies and for creating the Crypto Officer.
- **Crypto Officer (CO)** - Responsible for creating, modifying, and deleting crypto objects within the partition. The CO can use the crypto objects and create an optional, limited-capability role called Crypto User that can use the crypto objects but cannot modify them.
- **Crypto User (CU)** – An optional role that can use crypto objects while performing cryptographic operations.

  NOTE: Refer the “Thales Data Protection on Demand Application Owner Quick Start Guide” for configuring the HSM on Demand service and creating a service client.

> The HSM service client package is a zip file containing system information needed to connect your client machine to an existing HSM on Demand service.

To Configure DPoD HSM on Demand service with/without Luna Client

HSM on Demand Service can be configured in the following scenarios:

- **User wants to use DPoD Client to access service partition:** Execute steps 1, 2, 3, 4 and 10 below.
- **User wants to use Luna Client to access the service partition:** Execute steps 1-10 below.
- **User wants to use existing Luna Client to access the service partition in Hybrid mode with Luna Partition:** Execute steps 1-10 below.

> **NOTE:** Last two scenarios are supported for Universal Client only, from Luna Client v10.1.0 onwards.

To configure DPoD HSM on Demand service:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the DPoD client install directory.

   [Windows]

   ```
   cvclient-min.zip
   ```

   [Linux]

   ```
   cvclient-min.tar
   # tar -xvf cvclient-min.tar
   ```

4. Run the provided script to create a new configuration file containing information required by the HSMoD service.

   [Windows]

   Right-click **setenv.cmd** and select **Run as Administrator**.

   [Linux]

   Source the **setenv** script.

   ```
   # source ./setenv
   ```

> **NOTE:** Run the LunaCM utility available in the DPoD client and verify the service partition is listed. If you need to configure DPoD service partition with existing Luna Client follow further steps.

5. Copy the server and partition certificates from the DPoD client directory to your Luna client certificates directory:

   DPoD Certificates:

   ```
   server-certificate.pem
   partition-ca-certificate.pem
   partition-certificate.pem
   ```

   LunaClient Certificate Directory:

   [Windows default]

   ```
   C:\Program Files\Safenet\Lunaclient\cert\
   ```

   [Linux default]

   ```
   /usr/safenet/lunaclient/cert/
   ```

6. Open the configuration file from the DPoD client directory and copy the **XTC** and **REST** section.

   [Windows]

   ```
   crystoki.ini
   ```

   [Linux]

   ```
   Chrystoki.conf
   ```

7. Edit the Luna Client configuration file and add the **XTC** and **REST** section. In both sections you need to change only server and partition certificates path from step 5. Do not change any other entries provided in **XTC** and **REST** section.

   ```
   [XTC]
   . . .
   PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
   PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
   . . .

   [REST]
   . . .
   SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
   . . .
   ```

8. Edit the following entry from the **Misc** section and update the correct path for the **plugins** directory:

   ```
   Misc]
   PluginModuleDir=<LunaClient_plugins_directory>
   ```

   ```
   [Windows Default]
   ```

   ```
   C:\Program Files\Safenet\Lunaclient\plugins\
   ```

   ```
   [Linux Default]
   ```

   ```
   /usr/safenet/lunaclient/plugins/
   ```

   Save the configuration file. If you wish, you can now safely delete the extracted DPoD client directory.

9. Reset the **ChrystokiConfigurationPath** environment variable and point back to the location of the Luna Client configuration file.

   [Windows]

   In the Control Panel, search for "environment" and select **Edit the system environment variables**. Click **Environment Variables**. In both list boxes for the current user and system variables, edit **ChrystokiConfigurationPath** and point to the **crystoki.ini** file in the Luna client install directory.

   [Linux]

   Either open a new shell session, or export the environment variable for the current session pointing to the location of the **Chrystoki.conf** file:

   ```
   # export ChrystokiConfigurationPath=/etc/
   ```

10. Run the **LunaCM** utility and verify the service partition is listed. If you already have a Luna Partition before configuring the DPoD service partition, both Luna and DPoD service partition will be listed.

Constraints on HSM on Demand Services

HSM on Demand Service in FIPS mode

HSMoD services operate in a FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your HSM on Demand service. The FIPS mode is enabled by default. Refer to the *Mechanism List* in the *SDK Reference Guide* for more information about available FIPS and non-FIPS algorithms.

Verify HSM on Demand <slot value>

LunaCM commands work on the current slot. If there is only one slot, then it is always the current slot. If you are completing an integration using HSMoD services, you need to verify which slot on the HSMoD service you send commands to. If there is more than one slot, then use the slot set command to direct a command to a specified slot. You can use slot list to determine which slot numbers are in use by which HSMoD service.

Using Thales HSM in FIPS Mode

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for a FIPS-compliant HSM. If you are using the Luna HSM or HSM on Demand service in FIPS mode, you have to make the following change to the configuration file:

```
[Misc]

RSAKeyGenMechRemap=1
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM or HSMoD is in FIPS mode.

> **NOTE:** For Universal Client, above setting is not required. This setting is applicable for Luna Client 7.x only.

NOTE: This remapping is automatic if you are using Luna HSM Client 10.1 and above, and the configuration file entry is ignored.

Install Java Development Kit

Ensure that the Java Development Kit (JDK) is installed on your server or local computer. You can run the commands in this instruction wherever you have the keytool command available.

Signing Jar Files using Keys Generated on Thales HSM

This section demonstrates the method to use Java keytool utility for generating signing keys and certificate on Thales HSMs and Java jarsigner utility to sign the JAR file.

After creating your Certificate Signing request (CSR), ensure that you keep track of your keystore file as it contains the private key. In addition, you require the keystore file to install your Code Signing Certificate.

Configure Java for Luna Keystore

To generate the signing keys on Luna HSM using Java, you need to configure java.security configuration file and a Luna keystore file.

To configure java.security file for JDK 7/8:

1. Edit the Java Security Configuration file java.security located in the directory <JDK_installation_directory>/jre/lib/security.
2. Add the Luna Provider to the java.security file as shown below:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.safenetinc.luna.provider.LunaProvider
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=sun.security.ec.SunEC
security.provider.5=com.sun.net.ssl.internal.ssl.Provider
security.provider.6=com.sun.crypto.provider.SunJCE
security.provider.7=sun.security.jgss.SunProvider
security.provider.8=com.sun.security.sasl.Provider
security.provider.9=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.10=sun.security.smartcardio.SunPCSC
```

3. Save the changes to the **java.security** file.

To configure java.security file for JDK 11

1. Edit the Java Security Configuration file **java.security** located in the directory **<JDK_installation_directory>/conf/security**.
2. Add the Luna Provider to the **java.security** file as shown below:

```
security.provider.1=SUN
security.provider.2=com.safenetinc.luna.provider.LunaProvider
security.provider.3=SunRsaSign
security.provider.4=SunEC
security.provider.5=SunJSSE
security.provider.6=SunJCE
security.provider.7=SunJGSS
security.provider.8=SunSASL
security.provider.9=XMLDSig
security.provider.10=SunPCSC
security.provider.11=JdkLDAP
security.provider.12=JdkSASL
security.provider.13=SunPKCS11
```

3. Save the changes to the **java.security** file.

To create HSM keystore

1. Copy the **LunaAPI.dll** (Windows) or **libLunaAPI.so** (UNIX) and **LunaProvider.jar** file from the **<Luna_installation_directory>/jsp/lib** folder to the **<JDK_installation_directory>/jre/lib/ext**.

   NOTE: When using JDK 11, skip the step 1 as JDK 11 do not required to copy the files.

2. Set the environment variables for **JAVA_HOME** and **PATH**.

   ```
   # export JAVA_HOME=<JDK_installation_directory>
   # export PATH=$JAVA_HOME/bin:$PATH
   ```

   NOTE: For Windows, set the JDK bin folder in PATH variable under System Environments.

3. Create a file named **lunastore** (it could be any user defined name) and add the following entry where `<partition_label>` would be your Luna HSM partition name.

   ```
   tokenlabel:<partition_label>
   ```

4. Save the file in the current working directory.

Generate Signing Key and Certificate on Luna Keystore

Keytool utility provided by JDK will be used to generate signing keys and certificate on Luna HSM. To Generate Signing Keys:

1. Generate a key signing key and certificate using the Java **keytool** utility in the Luna keystore. This will generate the key pair in the Thales Luna HSM or HSM on Demand service.

   For JDK 7/8

```
# keytool -genkeypair -alias lunakey -keyalg RSA -sigalg SHA256withRSA -
keypass userpin1 -keysize 2048 -keystore lunastore -storepass userpin1 -
storetype luna
```

For JDK 11

```
# keytool -genkeypair -alias lunakey -keyalg RSA -keysize 2048 -sigalg
SHA256withRSA -keypass userpin1 -keystore lunastore -storetype Luna -
storepass userpin1 -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

When you run the command it will prompt for the following details:

```
What is your first and last name?
  [Unknown]:  Java Code Signing
What is the name of your organizational unit?
  [Unknown]:  RSA-HSM
What is the name of your organization?
  [Unknown]:  Thales
What is the name of your City or Locality?
  [Unknown]:  Noida
What is the name of your State or Province?
  [Unknown]:  Uttar Pradesh
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=Java Code Signing, OU=RSA-HSM, O=Thales, L=Noida, ST=Uttar Pradesh,
C=IN correct?
  [no]:  yes
```

A new key pair will be generated on the Luna HSM or HSMoD service.

> **NOTE:** The command above used “userpin1” as storepass which is the partition Crypto Officer Pin you set when initializing the CO role for the partition.

2. Verify that the private key is in the Luna HSM or HSMoD service.

For JDK 7/8

```
# keytool -list -v -storetype luna -keystore lunastore
```

For JDK 11

```
# keytool -list -v -keystore lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

Enter the keystore password, when prompted.

```
Enter keystore password:
Keystore type: LUNA
```

```
Keystore provider: LunaProvider
Your keystore contains 1 entry
Alias name: lunakey
Creation date: Aug 26, 2020
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Java Code Signing, OU=RSA-HSM, O=Thales, L=Noida, ST=Uttar
Pradesh, C=IN
Issuer: CN=Java Code Signing, OU=RSA-HSM, O=Thales, L=Noida, ST=Uttar
Pradesh, C=IN
Serial number: 21103d61
Valid from: Wed Aug 26 18:24:00 EDT 2020 until: Tue Nov 24 17:24:00 EST
2020
Certificate fingerprints:
SHA1: 80:5B:F5:12:88:A4:B5:45:F6:43:51:46:08:C4:10:2C:FF:88:BA:44
SHA256:
03:6D:86:7F:8B:28:DC:E3:62:F2:32:3C:1C:9C:D7:5A:5C:7C:67:55:5E:EE:BC:98:16:
40:BA:56:1A:43:E8:94
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
*******************************************
*******************************************
```

3. Generate a certificate request for a signing key in the Luna keystore.

   For JDK 7/8

   ```
   # keytool -certreq -alias lunakey -sigalg SHA256withRSA -file certreq_file
   -storetype luna -keystore lunastore
   ```

   For JDK 11

   ```
   # keytool -certreq -alias lunakey -sigalg SHA256withRSA -file certreq_file
   -keystore lunastore -storetype Luna -providerpath
   "/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
   com.safenetinc.luna.provider.LunaProvider -J-
   Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
   J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
   ```

   Enter the keystore password, when prompted. File **certreq_file** will be generated in the current directory.

4. Submit the CSR file to your Certification Authority (CA). Have the CA authenticate the request with the Code Signing template and return a signed certificate or a certificate chain. Save the reply and the root certificate of the CA in the current working directory.

5. Import the CA's Root certificate and signed certificate or certificate chain in to the keystore.

   For JDK 7/8

   To import the CA root certificate, execute the following:

   ```
   # keytool -trustcacerts -importcert -alias rootca -file root.cer -keystore
   lunastore -storetype luna
   ```

   To import the signed certificate reply or certificate chain, execute the following:

```
# keytool -importcert -trustcacerts -alias lunakey -file signing.p7b -
  keystore lunastore -storetype luna
```

For JDK 11

To import the CA root certificate, execute the following:

```
# keytool -trustcacerts -importcert -alias rootca -file root.cer -keystore
lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

To import the signed certificate reply or certificate chain, execute the following:

```
# keytool -trustcacerts -importcert -alias lunakey -file signing.p7b -
keystore lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

Where **root.cer** and **signing.p7b** are the CA Root Certificate and Signed Certificate Chain, respectively.

6. Verify the keystore contents in the Luna HSM or HSMoD service.

For JDK 7/8

```
# keytool -list -v -storetype luna -keystore lunastore
```

For JDK 11

```
# keytool -list -v -keystore lunastore -storetype Luna -providerpath
"/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
com.safenetinc.luna.provider.LunaProvider -J-
Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
```

Enter the keystore password, when prompted.

```
Enter keystore password:

Keystore type: LUNA

Keystore provider: LunaProvider

Your keystore contains 2 entries

Alias name: lunakey

Creation date: Aug 26, 2020

Entry type: PrivateKeyEntry

Certificate chain length: 2

Certificate[1]:

Owner: CN=Administrator, CN=Users, DC=intgca, DC=com

Issuer: CN=intgca-WIN-Q5F45F5JLU0-CA, DC=intgca, DC=com
```

```
Serial number: 4c000000b9daae1c328f7218900000000000b9

Valid from: Wed Aug 26 12:13:36 EDT 2020 until: Thu Aug 26 12:13:36 EDT
2021

Certificate fingerprints:

SHA1: 73:E5:F3:28:4D:D1:39:05:57:2C:03:6F:EF:20:9F:E6:97:36:3C:71

SHA256:
89:C3:FE:3F:3D:AD:7D:0F:6A:CE:1A:C6:9C:77:D8:15:48:F3:77:2A:09:F4:6C:24:F6:
7F:CE:04:3D:F7:AC:1C

Signature algorithm name: SHA256withRSA

Subject Public Key Algorithm: 2048-bit RSA key

Version: 3

Certificate[2]:

Owner: CN=intgca-WIN-Q5F45F5JLU0-CA, DC=intgca, DC=com

Issuer: CN=intgca-WIN-Q5F45F5JLU0-CA, DC=intgca, DC=com

Serial number: 1ccec3bcead7d9af42e932b59c487218

Valid from: Wed Feb 19 15:00:23 EST 2020 until: Wed Feb 19 15:10:23 EST
2025

Certificate fingerprints:

SHA1: 8C:34:D5:3D:53:2F:D3:4F:C7:32:A0:1B:7C:7C:BF:AB:C7:36:4F:89

SHA256:
A3:9F:BD:FE:5B:99:C7:B9:D9:A8:3C:C6:CB:78:5B:A8:BC:A0:FC:09:18:98:EE:BD:A5:
A7:C1:B7:F0:8E:48:7E

Signature algorithm name: SHA256withRSA

Subject Public Key Algorithm: 2048-bit RSA key

Version: 3

*******************************************

*******************************************

Alias name: rootca

Creation date: Aug 26, 2020

Entry type: trustedCertEntry

Owner: CN=intgca-WIN-Q5F45F5JLU0-CA, DC=intgca, DC=com

Issuer: CN=intgca-WIN-Q5F45F5JLU0-CA, DC=intgca, DC=com

Serial number: 1ccec3bcead7d9af42e932b59c487218

Valid from: Wed Feb 19 15:00:23 EST 2020 until: Wed Feb 19 15:10:23 EST
2025

Certificate fingerprints:
```

```
SHA1: 8C:34:D5:3D:53:2F:D3:4F:C7:32:A0:1B:7C:7C:BF:AB:C7:36:4F:89

SHA256:
A3:9F:BD:FE:5B:99:C7:B9:D9:A8:3C:C6:CB:78:5B:A8:BC:A0:FC:09:18:98:EE:BD:A5:
A7:C1:B7:F0:8E:48:7E

Signature algorithm name: SHA256withRSA

Subject Public Key Algorithm: 2048-bit RSA key

Version: 3

*******************************************
```

Sign and Verify Jar File

Jarsigner utility provided by JDK will be used to sign the jar files using signing keys on Luna HSM and verification will be done by the certificate embed in the signed jar file. To Sign/Verify Jar Files:

1. Move the jar file into the current working directory and sign the jar file. The system will prompt you for the Luna keystore password for signing the jar file.

For JDK 7/8

```
# jarsigner -keystore lunastore -storetype luna -signedjar
<name_of_signedjar_to_be_generated>    <jar_to_be_signed>
<alias_of_private_key> -tsa <time_stamping_authority_url>
```

Example

```
# jarsigner -keystore lunastore -storetype luna -signedjar signedsample.jar
 sample.jar lunakey -tsa
 http://timestamp.globalsign.com/scripts/timestamp.dll

 Enter Passphrase for keystore:

 jar signed.

 The signer certificate will expire on 2021-08-26.

 The timestamp will expire on 2027-06-23.
```

For JDK 11

```
 # jarsigner -keystore lunastore -storetype luna -signedjar
 <name_of_signedjar_to_be_generated> <jar_to_be_signed>
 <alias_of_private_key> -tsa <time_stamping_authority_url> -providername
 <lunaprovider> -providerclass <luna class path> -J-Djava.library.path=<luna
 JSP lib path> -J-cp -J<lunaprovider jar file>
```

Example

```
# jarsigner -keystore lunastore -storetype luna -signedjar signedsample.jar
 sample.jar lunakey -tsa
 http://timestamp.globalsign.com/scripts/timestamp.dll -providername
 LunaProvider -providerclass com.safenetinc.luna.provider.LunaProvider -J-
 Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
 J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar

 Enter Passphrase for keystore:
```

```
jar signed.

The signer certificate will expire on 2021-08-26.

The timestamp will expire on 2027-06-23.
```

2. Verify the signed jar file. Execute the following command. It will display the jar verified message if the operation is successful.

```
# jarsigner -verify signedsample.jar -verbose -certs

sm X    569 Wed Aug 26 18:44:48 EDT 2020 META-INF/MANIFEST.MF

        646 Wed Aug 26 18:44:48 EDT 2020 META-INF/LUNAKEY.SF

        6007 Wed Aug 26 18:44:48 EDT 2020 META-INF/LUNAKEY.RSA

        0 Thu Dec 02 10:41:40 EST 2010 META-INF/

 sm X   506 Mon May 21 00:09:04 EDT 2007  JSmoothPropertiesDisplayer$1.class

 sm X   533 Mon May 21 00:09:04 EDT 2007  JSmoothPropertiesDisplayer$2.class

 sm X   1567 Mon May 21 00:09:04 EDT 2007 JSmoothPropertiesDisplayer$3.class

 sm X   3905 Mon May 21 00:09:04 EDT 2007 JSmoothPropertiesDisplayer.class

        s = signature was verified

        m = entry is listed in manifest

        k = at least one certificate was found in keystore

        X = not signed by specified alias(es)

- Signed by "CN=Administrator, CN=Users, DC=intgca, DC=com"

        Digest algorithm: SHA-256

        Signature algorithm: SHA256withRSA, 2048-bit key

Timestamped by "CN=GlobalSign TSA for Standard - G2, O=GMO GlobalSign Pte
Ltd, C=SG" on Wed Aug 26 17:43:16 UTC 2020

        Timestamp digest algorithm: SHA-256

        Timestamp signature algorithm: SHA1withRSA, 2048-bit key jar
        verified.

The signer certificate will expire on 2021-08-26.

The timestamp will expire on 2027-06-23.
```

The .jar file is signed and verified. The private key and signing certificate are securely stored on the Luna HSM or HSM on Demand service. This completes the Java Code Signing integration with Luna HSM.

Migrating JKS keystore to Luna HSM Keystore

If the JKS keystore is already in use and the user wants to migrate the JKS keystore onto the Luna HSM or HSM on Demand service, complete the following procedure. Ensure that environment variables for JAVA_HOME and PATH are set.

1. Ensure that Luna HSM partition or HSMoD service is configured to use.

2. Configure Java to use Thales HSM as described in the section Configure Java for Luna Keystore.
3. Migrate the JKS keystore (examplestore) to lunastore.

   For JDK 7/8

   ```
   # keytool -importkeystore -srckeystore keystore.jks -destkeystore lunastore
   -srcstoretype jks -deststoretype luna
   ```

   For JDK 11

   ```
   # keytool -importkeystore -srckeystore keystore.jks -destkeystore lunastore
   -srcstoretype jks -deststoretype luna -providerpath
   "/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
   com.safenetinc.luna.provider.LunaProvider -J-
   Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
   J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
   ```

   When you run the command it will prompt for the following details:

   ```
   Importing keystore keystore.jks to lunastore...
   Enter destination keystore password:
   Enter source keystore password:
   Existing entry alias rootca exists, overwrite? [no]:  yes
   Entry for alias rootca successfully imported.
   Entry for alias jkskey successfully imported.
   Import command completed:  2 entries successfully imported, 0 entries
   failed or cancelled
   ```

   NOTE: When prompted for destination keystore password, provide partition Crypto Officer Pin that you had set when initializing the CO role for the partition.

4. Verify the keystore contents in the Luna HSM or HSMoD service.

   For JDK 7/8

   ```
   # keytool -list -v -storetype luna -keystore lunastore
   ```

   For JDK 11

   ```
   # keytool -list -v -keystore lunastore -storetype Luna -providerpath
   "/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar" -providerclass
   com.safenetinc.luna.provider.LunaProvider -J-
   Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -J-cp -
   J/usr/safenet/lunaclient/jsp/lib/LunaProvider.jar
   ```

   Enter the keystore password, when prompted. After successfully migrating the signing keys from JKS keystore to lunastore, you can sign/verify the jar files using the signing keys migrated on the Luna HSM or HSM on Demand service.

Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, refer to the documentation. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.