gemalto
security to be free

HSMoD Service

JAVA CODE SIGNER INTEGRATION GUIDE

Document Information

Product Version	1.7
Document Part Number	007-013897-001
Release Date	21 November 2018

Revision History

Revision	Date	Reason
Rev. A	17 August 2017	For initial release 1.1.0
Rev. B	19 September 2017	For release 1.1.1
Rev. C	14 November 2017	For release 1.2
Rev. D	05 February 2018	For release 1.3
Rev. E	02 March 2018	For HSM on Demand release 1.3
Rev. F	05 April 2018	For release 1.4
Rev. G	07 May 2018	For HSM on Demand release 1.4
Rev. H	10 June 2018	For release 1.5
	12 September 2018	For release 1.6
	21 November 2018	For release 1.7

SafeNet Data Protection on Demand1.7

JAVA CODE SIGNER INTEGRATION GUIDE

Contents

Overview

This guide demonstrates to Administrators how to sign Java artifacts using an encryption key generated on an HSM.

Java code signing is used for signing Java applications for desktops, digitally signing .jar files and Netscape Object signing recognized by Java Runtime Environment (JRE). In Java, the process for setting up your Code Signing Certificate consists of creating a keystore and a Certificate Signing Request (CSR) and then, installing your code signing certificate file to the keystore 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 with its certificate, so that it is available for use by anyone who has access to the key. 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 an HSM on Demand Service.

Using 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.

> improved performance by off-loading cryptographic operations from the signing servers.

This document contains the following sections:

Third Party Application Details

This integration guide uses the following third party applications:

> Java JDK 8

Supported Platforms

The following platforms are tested with HSMoD Service:

Platforms	Java Version
RHEL 64-bit	JDK 8
Windows Server 2016	JDK 8

Preparing for the Integration

Before you proceed with the integration, ensure you have completed the following:

> "Provision the HSMoD Service" below

Provision the HSMoD Service

The HSM on Demand Service provides your client machine with access to 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 must provision your HSM on Demand service by adding the service, downloading the service client package and initializing the HSM. Provisioning your HSM on Demand service entails:

> "Adding a Service" below

> "Adding a Service Client" below

> "Initializing the HSM" on page 8

Adding a Service

1. Under the **Services** tab, select the **Add New Service** page. Click **Deploy** on the service tile for the service you wish to add.

   NOTE Click **Deploy** on the HSM on Demand Service tile for your integration.

2. Review the "Terms of Services DPoD." Enable the **I have read and accept the Terms of Service above** check box and then click **Next**.
3. On the **Add <service_type> Service** page, enter a name for the Service in the **Service Name** field. You can optionally allow non-FIPS approved algorithms by selecting the **Allow non-FIPS approved algorithms** check box. Click **Next**.

   CAUTION! You cannot alter the FIPS setting after creating the service. You must decide if the service should allow or disallow non-FIPS approved algorithms before clicking **Finish** in the next step.

4. Review the configuration summary page. If acceptable, click **Finish**. If you would like to make changes to the configuration, click **Go Back**.

   When completed, the new service is listed under **My Services** and a **Create Service Client?** window displays.
5. Click **Create Service Client**.

Adding a Service Client

1. In the **Create Service Client** window enter a name for the service client in the **Service Client Name** field.

> **NOTE** If the **Create Service Client** window is not available, navigate to the **Services** tab and click the name of the Service you would like to generate a client for in the **My Services** table. On the Service Details page, click **New Service Client**.

2. Select **Create Service Client**.

   A new HSM service client package is created and provided for downloading on your client system.

   > **NOTE** The HSM service client package is a zip file that contains system information needed to connect your client system to an existing HSM on Demand service. The HSM service client package should download immediately on creation. If it does not, or you lose access to your HSM service client package it can be accessed or reacquired through the **My Services** table.

3. Transfer the service client package to your client system. You can use SCP, PSCP, WinSCP, FTPS, or any other secure file transfer tool.
4. Unzip the service client package.

   For Linux, enter:

   ```
   unzip <service_client_package>.zip
   ```

   For Windows, using the Windows GUI or an unzip tool unzip the file:

   ```
   <service_client_package>.zip
   ```

   > **NOTE** For more information about the service client package contents see .

5. Extract the cvclient-min file.

   > **NOTE** Extract the cvclient-min file in the directory where you extracted the <service_client_package>.zip. **Do not** extract to a new cvclient-min directory.

   For Linux, untar the cvclient-min.tar

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

   For Windows, unzip the cvclient-min.zip.
6. Set the environment variable.

   For Linux, execute:

   ```
   source ./setenv
   ```

   For Windows, right click setenv.cmd and select **Run as Administrator**.

   > **NOTE** If you encounter the error dll load failed with GetLastError() 126 move the contents of the cvclient_min folder up one directory and execute setenv.

7. Start LunaCM.

For Linux, execute the following from the directory where you extracted the cvclient-min.tar file.

```
./bin/64/lunacm
```

For Windows, execute the following from the directory where you unzipped the cvclient-min.zip file.

```
lunacm
```

Initializing the HSM

1. Set the active slot to the service partition.

   ```
   lunacm:>slot set -slot <slot_number>
   ```

   > **NOTE** Execute slot list in LunaCM to identify the slot number associated with your service.

2. Initialize the application partition. During this process you will create the partition's Security Officer (SO), set the SO password, and specify the cloning domain.

   ```
   lunacm:> partition init -label <service_label>
   ```

3. Optional: If you wish to transfer key material to or from a PED-authenticated Luna partition, you initialize the SafeNet Data Protection On Demand partition using the red PED domain key.

   a. For DPoD deployments, contact customer support to obtain the necessary PED drivers so that your HSM client can communicate with the PED.

   b. Attach the PED locally to the client computer, insert the red cloning domain PED key, and initialize the partition, including the option to set the cloning domain from the red PED key. Execute:

   ```
   lunacm:> partition init -label <cryptovisor_partition_label> -importpeddomain
   ```

4. Log in as the partition's Security Officer:

   ```
   lunacm:>role login -name Partition SO
   ```

5. Initialize the Crypto Officer role:

   ```
   lunacm:>role init -name Crypto Officer
   ```

6. Log out of the partition Security Officer role and log in as the Crypto Officer.

   ```
   lunacm:>role logout
   lunacm:>role login -name Crypto Officer
   ```

7. You must change the Crypto Officer password immediately on the initial log in. Failure to due so will result in a password error on subsequent logins.

   ```
   lunacm:>role changepw -name Crypto Officer
   ```

8. Initialize the Crypto User role:

   ```
   lunacm:>role init -name Crypto User
   ```

9. Log out of the partition Crypto Officer role and log in as the Crypto User.

   ```
   lunacm:>role logout
   lunacm:>role login -name Crypto User
   ```

10. You must change the Crypto User password immediately on the initial log in. Failure to do so will result in a password error on subsequent logins.

    ```
    lunacm:>role changepw -name Crypto User
    ```

This completes initializing the HSM on Demand Service. The Crypto Officer and Crypto User roles can now be used to integrate applications with the HSMoD service to perform cryptographic operations

Constraints on HSMoD Services

Please take the following limitations into consideration when integrating your application software with an HSM on Demand Service.

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.

Integrating Java Code Signing with an HSM on Demand Service

This document provides detailed instructions to use the Java keytool utility to generate signing keys and certificates using the HSM on Demand Service, and then use those keys to sign a **.jar** file.

Ensure that the Java Development Kit (JDK) is installed on your local system, and the bin directory is accessible by users on the system. We recommend adding the JDK Bin folder to the PATH environment, alternatively, you can run the commands from the /bin/ directory.

This integration contains the following topics:

Configuring the java.security file

You must update the **java.security** configuration file to use the SafeNet security providers and the HSM on Demand Service.

To configure the java.security file

1. Open the Java security configuration file **java.security** in a text editor. The file is available at <JDK_ installation_directory>**/jre/lib/security**.
2. Update the Luna Providers in the **java.security** file so they appear as follows:

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

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

Enabling the HSM on Demand Service keystore

You must configure the Java Code Signing utility to use the keystore located on the HSM on Demand Service.

To enable the HSM on Demand Service keystore

1. Copy the **LunaAPI.dll** (Windows) or the **libLunaAPI.so** (UNIX) and the **LunaProvider.jar** files from the <Luna_installation_directory>**/jsp/lib** to the Java extension folder located at <JDK_installation_

directory>**/jre/lib/ext**.

2. Set the environment variables for JAVA_HOME and PATH.

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

> **NOTE** We recommend setting the PATH variable in Windows environments using the System Environments menu.

3. Create a file called lunastore and add the following line:

```
tokenlabel:<service_label>
```

Acquire encryption keys for signing the .jar file

There are two methods for acquiring the encryption key on the HSM on Demand Service for signing the .jar files. You can migrate an existing keystore into the HSM on Demand Service, or generate a new keypair for use on the HSM on Demand Service..

Migrating the JKS keystore for signing the .jar file

If the JKS keystore is already in use, you can migrate the keys from the JKS keystore to the HSM on Demand Service to better secure the encryption keys. The following procedure details the steps required to migrate signing keys from the JKS local keystore to the keystore on the HSM on Demand Service.

To migrate the JKS keystore for signing the .jar file

1. Migrate the JKS keystore into the <product>.

   # keytool –importkeystore –srckeystore <source_key_store> **–destkeystore lunastore –srcstoretype jks –deststoretype luna**

   Respond to the prompts from the system to complete the migration.

2. Verify the contents of the lunastore.

   # keytool –list –v –keystore lunastore –storetype luna

Generating the encryption key for signing the .jar file

You can generate a new keypair directly on the HSM on Demand Service keystore.

To generate the encryption key for signing the .jar file

1. Generate a keypair using the Java keystore utility and respond to the prompts to configure the keypair.

   # keytool –genkeypair –alias <key_label> **–keyalg RSA –sigalg SHA256withRSA –keypass** <key_password> **-keysize 2048 –keystore lunastore –storepass** <HSMoD_password> **-storetype luna**

   When you complete the process, a new key pair will be generated on the registered HSM on Demand Service.

2. Verify that the private key exists on the HSM on Demand Service.

   # keytool –list –v –storetype luna –keystore lunastore

The system will prompt the user to enter the keystore password. This is the password for the HSM on Demand Service. On correct entry, it will display the existing keys.

3. Generate a certificate request from the private key in the keystore.

   # keytool –certreq –alias <key_label> **–sigalg SHA256withRSA –file**<cert_request_file> **–storetype luna –keystore lunastore**.

   The system will prompt the user to enter the keystore password. This is the password for the HSM on Demand Service.

4. Deliver the CSR file to your local Certification Authority (CA). Request the CA authenticates the request and returns a signed certificate or certificate chain. Save the reply and the CA root certificate in the current working directory.

5. Import the CA root certificate into the keystore.

   # keytool –trustcacerts –importcert –alias rootca –file <root_certificate> **–keystore lunastore –storetype luna**

6. Import the signed certificate reply, or certificate chain, into the keystore.

   # keytool –trustcacerts –importcert –alias <key_label> **–file** <certificate_file_or_chain> **-keystore lunastore –storetype luna**

7. Verify the files exist in the keystore.

   # keytool –list –v –storetype luna –keystore lunastore

> **NOTE** Ensure you keep a record of the location of your keystore file following the creation of the CSR. You will need this location as it contains your private key, and it will be needed when you install the Code Signing Certificate.

Signing the .jar file

You can sign the .jar files using the encryption keys stored in the HSM on Demand Service keystore.

To sign the .jar file

1. Move the ***.jar** file inside the current working directory.

2. Sign the ***.jar** file. Execute:

   # jarsigner –keystore lunastore –storetype luna –signedjar <signed_jar_to_be_generated> <jar_to_be_signed> <private_key_label> **-tsa** <time_stamping_URL>

   On correct execution, the system will prompt the user to enter the keystore password. This is the password for the HSM on Demand Service.

3. Verify the signed ***.jar** file.

   # jarsigner –verify <signed_jar_file> **-verbose –certs**

   If the .jar file is verified, a message similar to the following is returned.

```
S      565 Tue Apr 17 09:42:36 PDT 2018 META-INF/MANIFEST.MF
       [entry was signed on 4/16/18 9:12 PM]
       X.509, CN=Administrator, CN=Users, DC=hlk, DC=com
```

```
      [certificate is valid from 4/16/18 12:02 PM to 4/16/19 12:02 PM]
      X.509, CN=hlk-CA, DC=hlk, DC=com
      [certificate is valid from 4/5/18 10:25 AM to 4/5/23 10:35 AM]
      647 Tue Apr 17 09:42:36 PDT 2018 META-INF/LUNAKEY.SF
      5869 Tue Apr 17 09:42:36 PDT 2018 META-INF/LUNAKEY.RSA
      0 Thu Dec 02 10:41:40 PST 2010 META-INF/
m     506 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer$1.class
m     533 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer$2.class
m     1567 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer$3.class
m     3905 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer.class
s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore
i = at least one certificate was found in identity scope
-Signed by "CN=Administrator, CN=Users, DC=hlk, DC=com"
Digest algorithm : SHA256
Signature algorithm : SHA256withRSA, 2048-bit key
Timestamped by “CN=GlobalSign TSA for Standard – G2, O=GMO GlobalSign Pte Ltd, C=SG” on Tue
Apr 17 04:12:52 UTC 2018
Timestamp digest algorithm: SHA-256
Timestamp signature algorithm: SHA1withRSA, 2048-bit key
Jar verified.
```