THALES

# Keycloak

## INTEGRATION GUIDE
## THALES LUNA HSM

## Document Information

| | |
|---|---|
| **Document Part Number** | 007-001280-001 |
| **Revision** | B |
| **Release Date** | 30 June 2022 |

## Trademarks, Copyrights, and Third-Party Software

# CONTENTS

# Overview

Keycloak is an open source identity and access management solution aimed at modern applications and services. It makes it easy to secure applications and services with little or no code. It provides advanced features such as user federation, identity brokering, and social login. Keycloak is based on standard protocols and provides support for OpenID Connect, OAuth 2.0, and SAML.

This guide demonstrates how to generate the Keycloak Realm signing keys on Luna HSM. Realm signing key is used to sign the access token and XML documents between the authentication server and the application. Using a Luna HSM to generate the signing keys for Keycloak 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.

# Certified Platforms

This integration is tested on the following platforms:

| HSM Type | Platforms Tested | JDK Version | Keycloak Version |
|---|---|---|---|
| Luna HSM | RHEL 8 | Open JDK 11 | 15.1.0, 13.0.1 |
| Luna HSM | RHEL 7 | Open JDK 8 | 13.0.1, 12.0.1, 11.0.1, 10.0.1, 9.0.2 |

**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.

**NOTE:** The integration with Keycloak requires Luna Client v10.4 or above.

**NOTE:** Obtain a **patch** from Thales Support containing Keycloak Luna plugin that will enable the Keyclock to use Luna Keystore and signing keys generated on Luna HSM.

**NOTE:** Although README attached to the **patch** refers to a single supported version of Keycloak, refer to the table provided at the beginning of the Certified Platforms section for the Keycloak versions that are actually certified.

# Prerequisites

Complete the following tasks before you begin the installation:

## Configure Luna HSM

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

1. Ensure that the HSM is setup, initialized, provisioned, and ready for deployment.
2. Create a partition on the HSM for use by Keycloak.
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 (64-bit) v10.4.0-417. Copyright (c) 2021 SafeNet. All rights
reserved.


        Available HSMs:

        Slot Id ->              0
        Label ->                INTG01
        Serial Number ->        1312109861412
        Model ->                LunaSA 7.7.0
        Firmware Version ->     7.7.0
        Bootloader Version ->   1.1.2
        Configuration ->        Luna User Partition With SO (PW) Key Export
                                With Cloning Mode
        Slot Description ->     Net Token Slot
        FM HW Status ->         Non-FM

        Current Slot Id: 0
```

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

### Control user access to 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.

### Add 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.

### Remove 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

### Set 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 or UNIX systems. You must enable the HAOnly setting in HA for failover to work so that if the primary stop functioning for some reason, all calls are automatically routed to secondary till the primary starts functioning again.

## Download Keycloak Luna SPI Patch

Keycloak does not provide support for Luna HSM Keystore, but facilitates development of a plugin using Signature Provider Interface (SPI) that leverages the Luna HSM Keystore. Contact Thales Customer Support to obtain Keycloak Luna plugin. Thales Customer Support will provide you a patch containing Keycloak Luna plugin that you can use to enable Keyclock to use Luna Keystore and signing keys generated on Luna HSM.

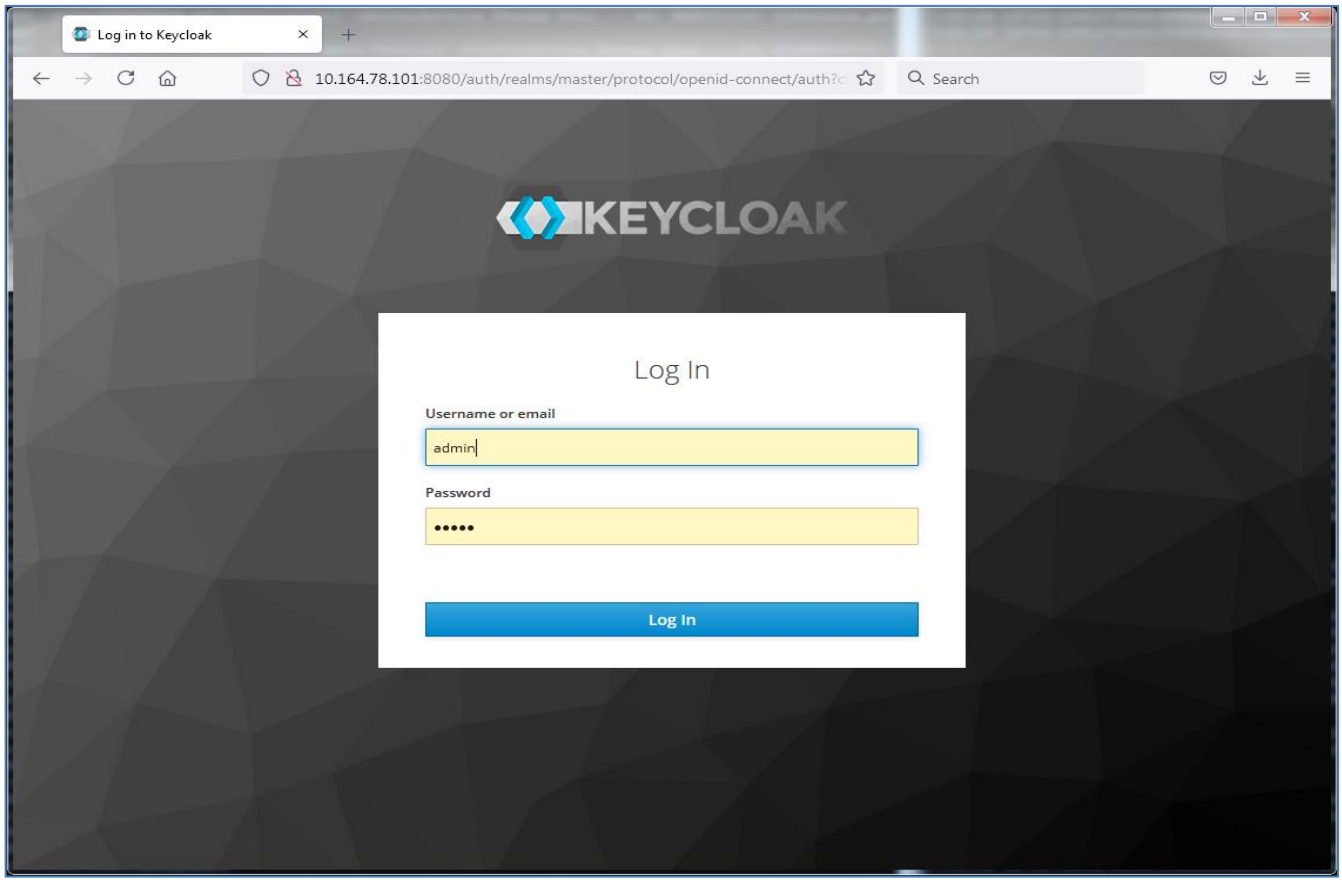
> **NOTE:** README attached to the patch refers to a single supported version of Keycloak but this guide takes precedence as to what actual versions of Keycloak are supported.

## Install Java Development Kit

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

## Set up Keycloak Server

To log in to the Keycloak Admin console, ensure that the Keycloak server is installed and running and you have created the initial Admin user. Consult the Keycloak documentation for detailed instructions about installing and setting up a Keycloak server: https://www.keycloak.org/documentation.




# Configuring Keycloak to Use HSM-Generated Signing Keys

This section demonstrates the steps required for generating signing keys and certificate on Luna HSM and configuring the Keycloak Realm to use the HSM generated key for token signing and XML document signing. Following are the steps involved in achieving Keycloak and Luna HSM integration:

> Configure Java for Luna Keystore

> Generate signing keys and certificate on Luna Keystore

> Configure Keycloak for Luna Keystore

## Configure Java for Luna Keystore

To generate the signing keys on Luna HSM using Java, configure Java to use the Luna Provider.

1. Set the environment variables for JAVA_HOME and PATH.

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

```
[root@keycloak ~]#
[root@keycloak ~]# export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.222.b03-1.el7.x86_64
[root@keycloak ~]# export PATH=$JAVA_HOME/bin:$PATH
[root@keycloak ~]#
```

2. Copy the libLunaAPI.so and LunaProvider.jar file from the <Luna_installation_directory>/jsp/lib directory to the <JDK_installation_directory>/jre/lib/ext directory.

   **NOTE:** Skip this step, if using JDK 11, as JDK 11 doesn't support to copy the provider.

3. Edit the Java Security Configuration file $JAVA_HOME/conf/security/java.security and add the Luna Provider to the java.security file. Adjust the provider list, as shown below:

   **For JDK 11:**

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

   **For JDK 8:**

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

4. Save and close the java.security file.

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

   ```
   tokenlabel:<partition_label>
   ```

6. Save the lunastore file in the current working directory, let's say /opt.

## Generate signing key and certificate on Luna Keystore

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

1. Generate a signing key and certificate using the Java keytool utility leveraging Luna keystore. This will generate the key pair in Luna HSM.

   **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
   ```

   **For JDK 8:**

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

   When the above command is run, you need to provide certificate details. A new key pair will be generated on the Luna HSM.

   > **NOTE:** The command above uses "userpin1" as storepass which is the partition's Crypto Officer pin set during initializing the CO role for the partition. You can use the same password for keypass.

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

   **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
   ```

   **For JDK 8:**

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

   Enter the keystore password, when prompted. A file named certreq_file will be generated in the current directory.

3. Submit the CSR file to your Certification Authority (CA). The CA will 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.

4. Import the CA's root certificate and signed certificate or certificate chain to the keystore.

   **For JDK 11:**

   To import the CA root certificate, execute the following command:

   ```
   # 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 command:

   ```
   # keytool -importcert -trustcacerts -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
   ```

   **For JDK 8:**

   To import the CA root certificate, execute the following command:

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

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

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

   Where root.cer and signing.p7b are the CA Root Certificate and Signed Certificate Chain, respectively. Enter the keystore password, when prompted.

5. Verify the keystore contents in the Luna HSM.

   **For JDK 11:**

   To display the contents in lunastore, execute the following command:

   ```
   # 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
   ```

   **For JDK 8:**

   To display the contents in lunastore, execute the following command:

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

```
[root@keycloak opt]# keytool -list -v -storetype luna -keystore lunastore
Enter keystore password:
Keystore type: LUNA
Keystore provider: LunaProvider

Your keystore contains 2 entries

Alias name: lunakey
Creation date: Jun 2, 2021
Entry type: PrivateKeyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=Administrator, CN=Users, DC=thaleshsm, DC=com
Issuer: CN=ORGCA, DC=thaleshsm, DC=com
Serial number: 14000000a59ad15742ad96b4590000000000a5
Valid from: Wed Jun 02 16:01:31 IST 2021 until: Thu Jun 02 16:01:31 IST 2022
Certificate fingerprints:
         MD5:  75:40:1B:95:D3:B3:F8:05:5F:42:A4:99:41:87:06:D3
         SHA1: F1:25:DA:70:40:89:17:39:5C:05:41:FE:E3:36:DB:FC:0E:07:BE:D4
         SHA256: CE:7F:DA:B2:9E:75:40:0B:09:09:8D:91:EB:0D:8E:50:E3:74:30:91:90:E8:99:BC:2E:CD:65:6D:CA:D4:B5:FD
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
Certificate[2]:
Owner: CN=ORGCA, DC=thaleshsm, DC=com
Issuer: CN=ORGCA, DC=thaleshsm, DC=com
Serial number: 6532c42d59f069b245089dc5d129b2c1
Valid from: Tue Feb 09 15:25:40 IST 2021 until: Mon Feb 09 15:35:38 IST 2026
Certificate fingerprints:
         MD5:  70:0A:DE:C6:4F:65:F9:28:A2:42:9B:87:72:1E:5D:82
         SHA1: 91:AF:11:9C:3F:BA:E9:CB:19:A4:09:3E:B1:06:3C:BA:BC:96:CE:56
         SHA256: 90:D9:D1:32:0D:77:99:F1:43:4A:37:39:A4:8F:51:CC:80:D3:93:58:6D:02:EF:96:54:69:2B:92:0A:2D:E6:61
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3


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


Alias name: rootca
Creation date: Jun 2, 2021
Entry type: trustedCertEntry

Owner: CN=ORGCA, DC=thaleshsm, DC=com
Issuer: CN=ORGCA, DC=thaleshsm, DC=com
Serial number: 6532c42d59f069b245089dc5d129b2c1
Valid from: Tue Feb 09 15:25:40 IST 2021 until: Mon Feb 09 15:35:38 IST 2026
Certificate fingerprints:
         MD5:  70:0A:DE:C6:4F:65:F9:28:A2:42:9B:87:72:1E:5D:82
         SHA1: 91:AF:11:9C:3F:BA:E9:CB:19:A4:09:3E:B1:06:3C:BA:BC:96:CE:56
         SHA256: 90:D9:D1:32:0D:77:99:F1:43:4A:37:39:A4:8F:51:CC:80:D3:93:58:6D:02:EF:96:54:69:2B:92:0A:2D:E6:61
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3


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

## Configure Keycloak for Luna Keystore

Keycloak does not provide support for Luna HSM Keystore, but facilitates development of the plugin using Signature Provider Interface (SPI) that leverages the Luna HSM Keystore. Contact Thales Customer Support to obtain Keycloak Luna plugin. Thales Customer Support will provide you a patch containing Keycloak Luna plugin that will enable the Keyclock to use Luna Keystore and signing keys generated on Luna HSM.

1. Extract the patch and copy the Keycloak Luna Plugin zip file (extracted from the patch) to the Keycloak modules directory.

```
# tar -xvf 630-000621-001_SW_Patch_keycloak_integration_Custom_Release.tar
```

```
# cp 630-000621-001_SW_Patch_keycloak_integration_Custom_Release/keycloak-spi-luna-keystore-1.0-assemblyModule.zip /opt/keycloak/modules/
```

**For KeyClock 15 onwards:**

```
# cp 630-000621-001_SW_Patch_keycloak_integration_Custom_Release/keycloak-spi-luna-keystore-1.1-assemblyModule.zip /opt/keycloak/modules/
```

2. Traverse to the Keycloak modules directory and extract the plugin zip file.

```
# cd /opt/keycloak/modules/
```

```
# unzip keycloak-spi-luna-keystore-1.0-assemblyModule.zip
```

**For KeyClock 15 onwards:**

```
# unzip keycloak-spi-luna-keystore-1.1-assemblyModule.zip
```

```
[root@keycloak modules]# unzip keycloak-spi-luna-keystore-1.0-assemblyModule.zip
Archive:  keycloak-spi-luna-keystore-1.0-assemblyModule.zip
   creating: com/
   creating: com/safenetinc/
   creating: com/safenetinc/luna/
   creating: com/safenetinc/luna/keycloak/
   creating: com/safenetinc/luna/keycloak/provider/
  inflating: com/safenetinc/luna/keycloak/provider/module.xml
  inflating: com/safenetinc/luna/keycloak/provider/keycloak-spi-luna-keystore-1.0.jar
[root@keycloak modules]#
```

3. Traverse to the com/safenetinc/luna directory and create a main/lib/linux-x86_64 folder under the com\safenetinc\luna directory.

```
# cd /opt/keycloak/modules/com/safenetinc/luna/
```

```
# mkdir -p main/lib/linux-x86_64
```

4. Copy the LunaProvider.jar and libLunaAPI.so file to the main folder created under the com\safenetinc\luna directory.

```
# cp <Luna_installation_directory>/jsp/lib/LunaProvider.jar main/
```

5. Copy libLunaAPI.so file to the main/lib/linux-x86_64 directory created under the com\safenetinc\luna directory.

```
# cp <Luna_installation_directory>/jsp/lib/libLunaAPI.so main/lib/linux-x86_64
```

6. Create a file module.xml in the main folder and add the following information in it.

```
# cd main
```

```
# vi module.xml
```

```
<module name="com.safenetinc.luna" xmlns="urn:jboss:module:1.9">
  <resources>
    <resource-root path="LunaProvider.jar"/>
  </resources>
  <dependencies>
   <module name="java.logging"/>
  </dependencies>
</module>
```

7. Ensure that the LunaProvider.jar, module.xml, and lib/linux-x86_64/libLunaAPI.so files are present in the /opt/keycloak/modules/com/safenetinc/luna/main directory that you've created.

```
[root@keycloak main]# ls -ltr /opt/keycloak/modules/com/safenetinc/luna/main/
total 628
drwxr-xr-x. 3 root root     26 Jun 11 20:57 lib
-rw-r--r--. 1 root root 638381 Jun 11 20:58 LunaProvider.jar
-rw-r--r--. 1 root root    218 Jun 11 21:58 module.xml
[root@keycloak main]#
```

8. Traverse to the com/safenetinc/luna/keycloak/provider directory created in the Keycloak's modules directory.

```
# cd /opt/keycloak/modules/com/safenetinc/luna/keycloak/provider/
```

9. Create a main folder under the com/safenetinc/luna/keycloak/provider directory.

```
# mkdir main
```

10. Move the keycloak-spi-luna-keystore-1.0.jar and module.xml to the main directory.

```
# mv keycloak-spi-luna-keystore-1.0.jar module.xml main/
```

**For KeyClock 15 onwards:**

```
# mv keycloak-spi-luna-keystore-1.1.jar module.xml main/
```

11. Ensure that the keycloak-spi-luna-keystore-x.x.jar and module.xml are present in the /opt/keycloak/modules/com/safenetinc/luna/keycloak/provider/main directory that you've created.

    Where x.x is the version depending upon the Keycloak version used.

```
[root@keycloak main]# ls -ltr /opt/keycloak/modules/com/safenetinc/luna/keycloak/provider/main
total 16
-rw-rw-r--. 1 root root 9242 Jun  8 14:49 keycloak-spi-luna-keystore-1.0.jar
-rw-rw-r--. 1 root root  752 Jun 11 21:06 module.xml
[root@keycloak main]#
```

12. Ensure that the module.xml file has the following content, including the name of the Plugin jar file.

> **NOTE:** For Keycloak v15.x, ensure that "keycloak-spi-luna-keystore-1.1.jar" is used.

```
<?xml version='1.0' encoding='UTF-8'?>
<module xmlns="urn:jboss:module:1.3" name="com.safenetinc.luna.keycloak.provider" slot="main">
    <resources>
        <resource-root path="keycloak-spi-luna-keystore-1.0.jar"/>
    </resources>
    <dependencies>
        <module name="org.keycloak.keycloak-server-spi" services="import"/>
        <module name="org.keycloak.keycloak-server-spi-private" services="import"/>
        <module name="org.keycloak.keycloak-core"/>
        <module name="org.keycloak.keycloak-common"/>
        <module name="org.keycloak.keycloak-services"/>
        <module name="org.jboss.resteasy.resteasy-jaxrs"/>
        <module name="org.jboss.logging"/>
        <module name="com.safenetinc.luna"/>
    </dependencies>
</module>
~
```

**13.** Edit the Keycloak configuration file standalone.xml to add the Provider and SPI details as follows:

```
# vi /opt/keycloak/standalone/configuration/standalone.xml
```

```
    <provider>
        module:com.safenetinc.luna.keycloak.provider
    </provider>
     .
     .
     <spi name="keys">
          <provider name="luna-keystore" enabled="true"/>
     </spi>
```

You need to mention the provider and SPI details above in the Provider and SPI sections already present in the XML file. Ensure that luna-keystore spi is specified as the first SPI in the list.

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.1">
    <web-context>auth</web-context>
    <providers>
        <provider>classpath:${jboss.home.dir}/providers/*</provider>
        <provider>module:com.safenetinc.luna.keycloak.provider</provider>
    </providers>
    <master-realm-name>master</master-realm-name>
    <scheduled-task-interval>900</scheduled-task-interval>
    <theme>
        <staticMaxAge>2592000</staticMaxAge>
        <cacheThemes>true</cacheThemes>
        <cacheTemplates>true</cacheTemplates>
        <dir>${jboss.home.dir}/themes</dir>
    </theme>
    <spi name="keys">
        <provider name="luna-keystore" enabled="true"/>
    </spi>
    <spi name="eventsStore">
        <provider name="jpa" enabled="true">
            <properties>
                <property name="exclude-events" value="[&quot;REFRESH_TOKEN&quot;]"/>
            </properties>
        </provider>
    </spi>
```

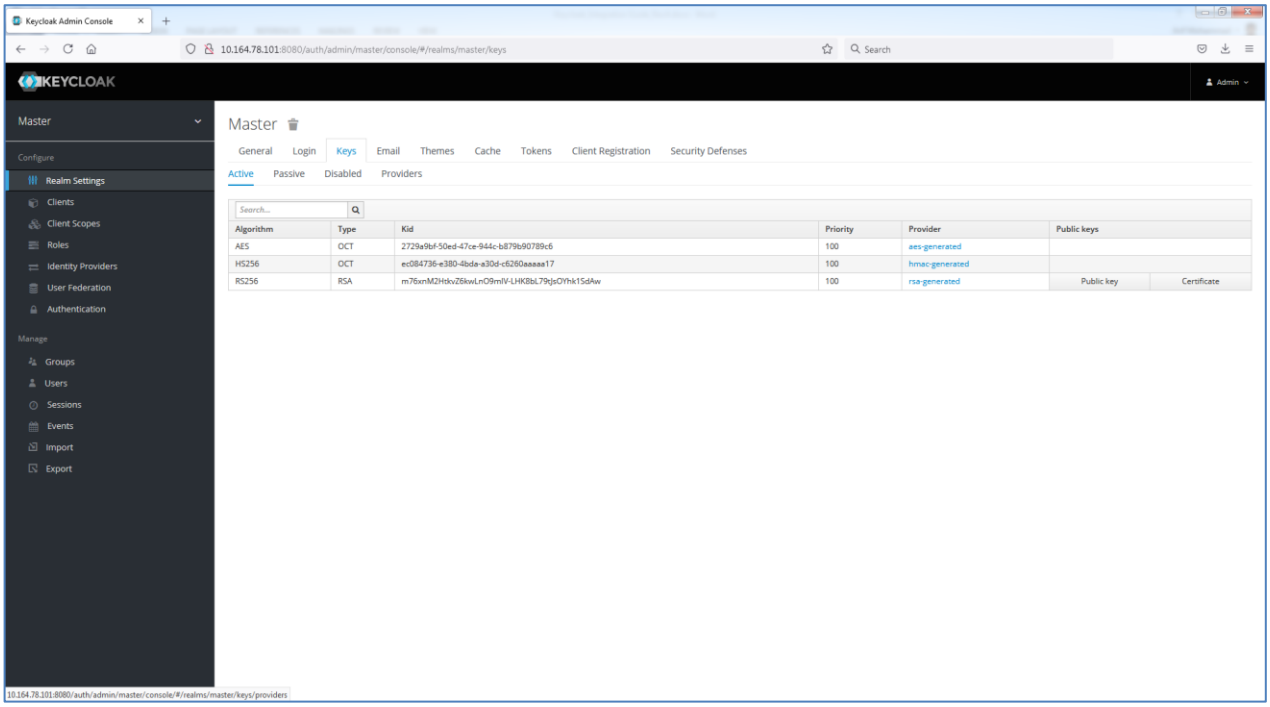
**14.** Save and close the standalone.xml file. Now stop the Keycloak server if it is already running, or start it again.

```
# /opt/keycloak/bin/standalone.sh
```

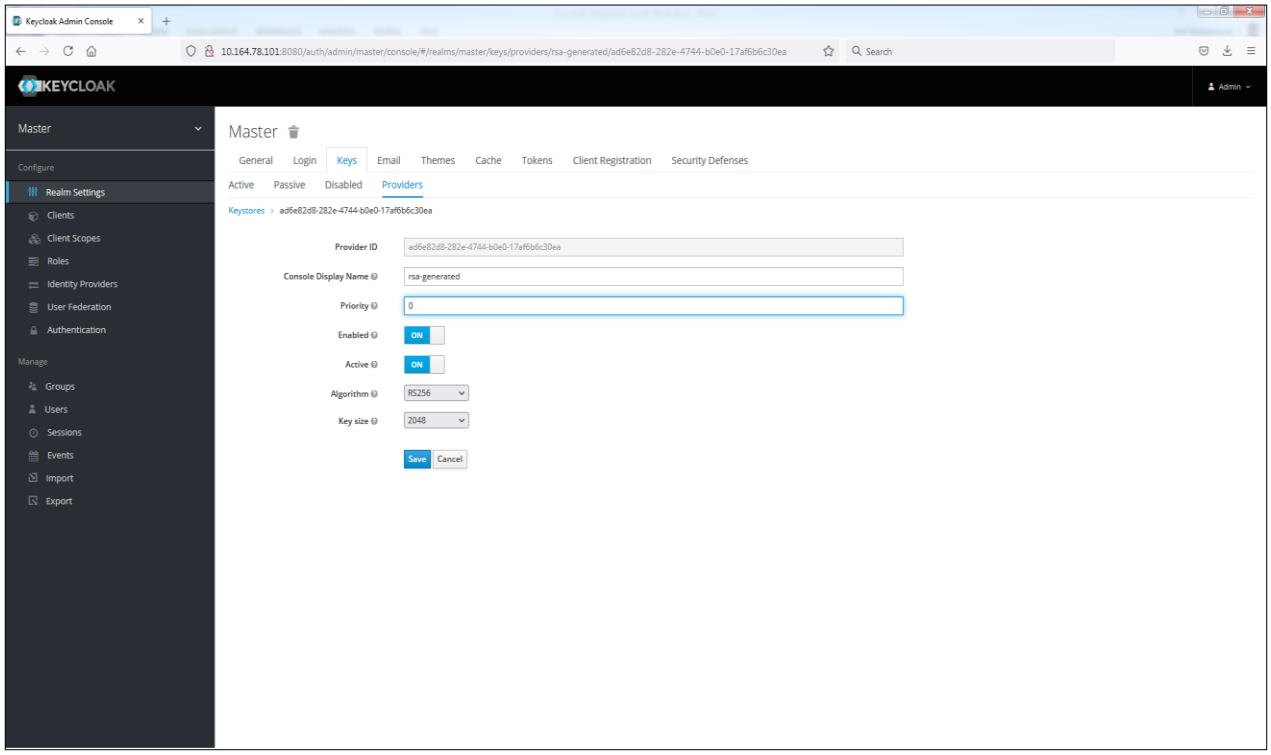
You will see the following information when the server starts.

```
22:27:29,211 INFO  [org.keycloak.services] (ServerService Thread Pool --
   62) KC-SERVICES0001: Loading config from standalone.xml or domain.xml

22:27:29,928 WARN  [org.keycloak.services] (ServerService Thread Pool --
   62) KC-SERVICES0047: luna-keystore
   (com.safenetinc.luna.keycloak.provider.LunaKeystoreProviderFactory) is
   implementing the internal SPI keys. This SPI is internal and may
   change without notice

22:27:30,326 INFO  [org.keycloak.url.DefaultHostnameProviderFactory]
(ServerService Thread Pool -- 62) Frontend: <request>, Admin: <frontend>,
Backend: <request>
```

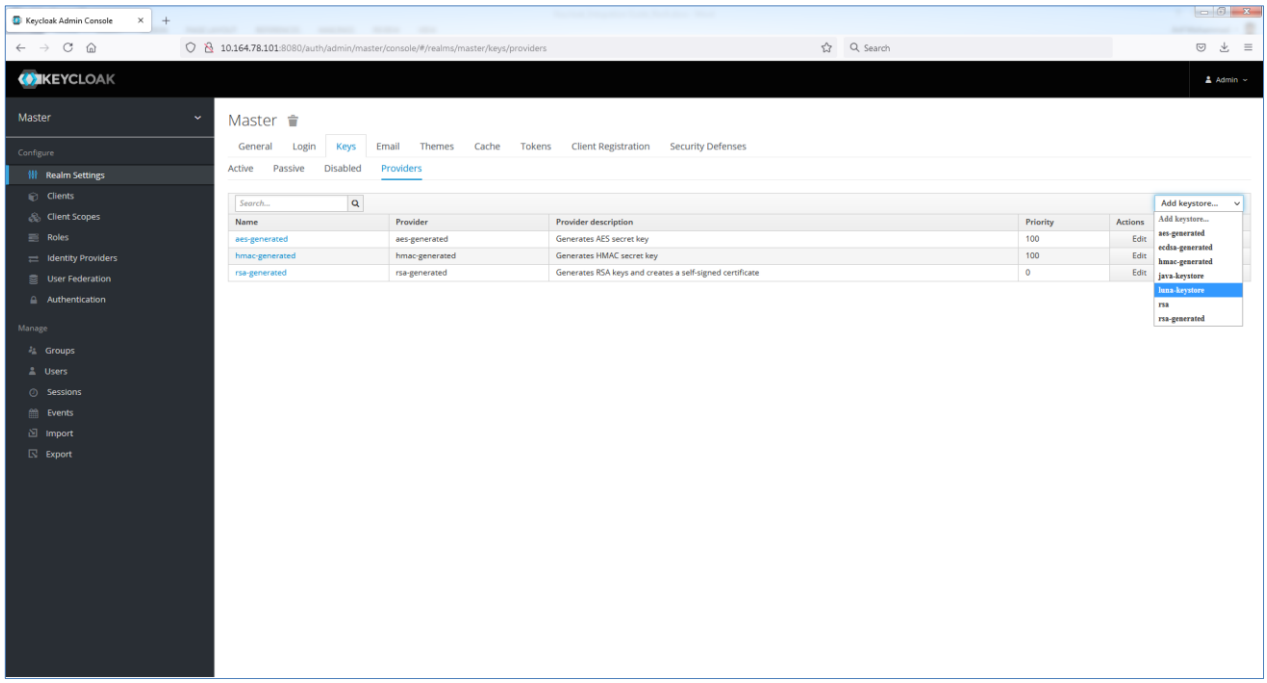
**15.** Log in to the Keycloak Admin Console and navigate to **Realm Settings** > **Keys**.




**16.** Click **rsa-generated** key in **Provider** column and change the **Priority** from 100 to 0. Click **Save**.



**17.** Navigate to **Realm Settings** > **Keys** > **Providers**. Click **Add keystore…** and select **luna-keystore**.




**18.** Enter the values for **Priority**, **Keystore**, **Keystore Password**, **Key Alias** and **Key Password**. Click **Save**.

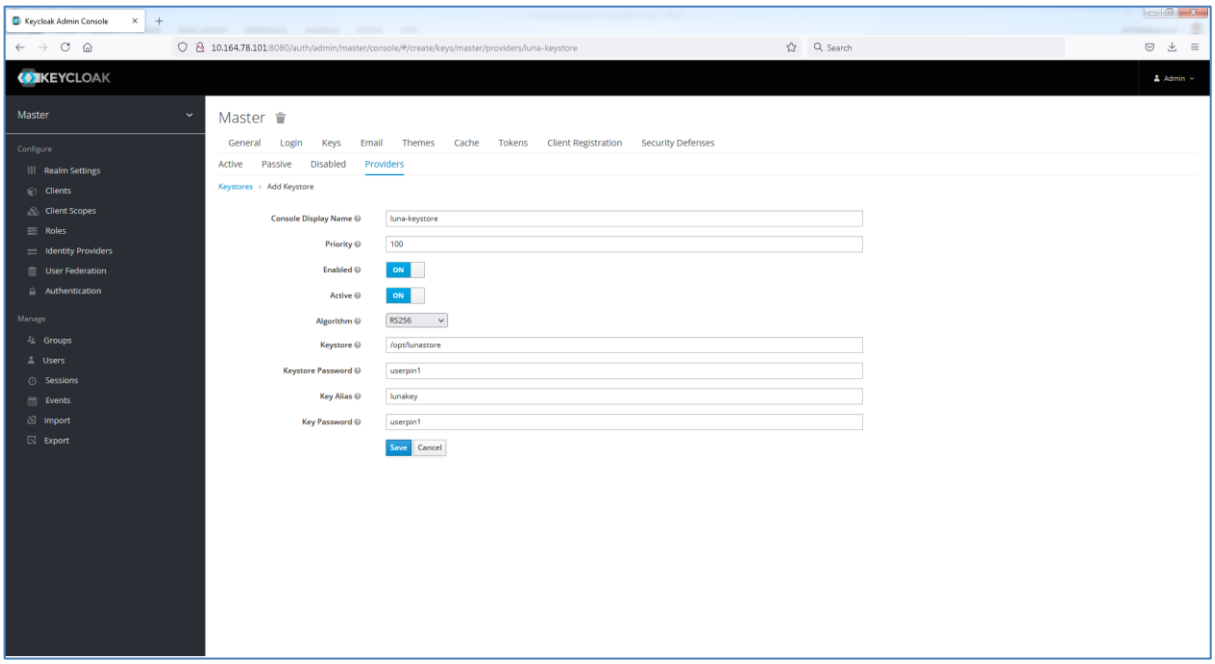
Priority = 100

Keystore = Path to lunastore file

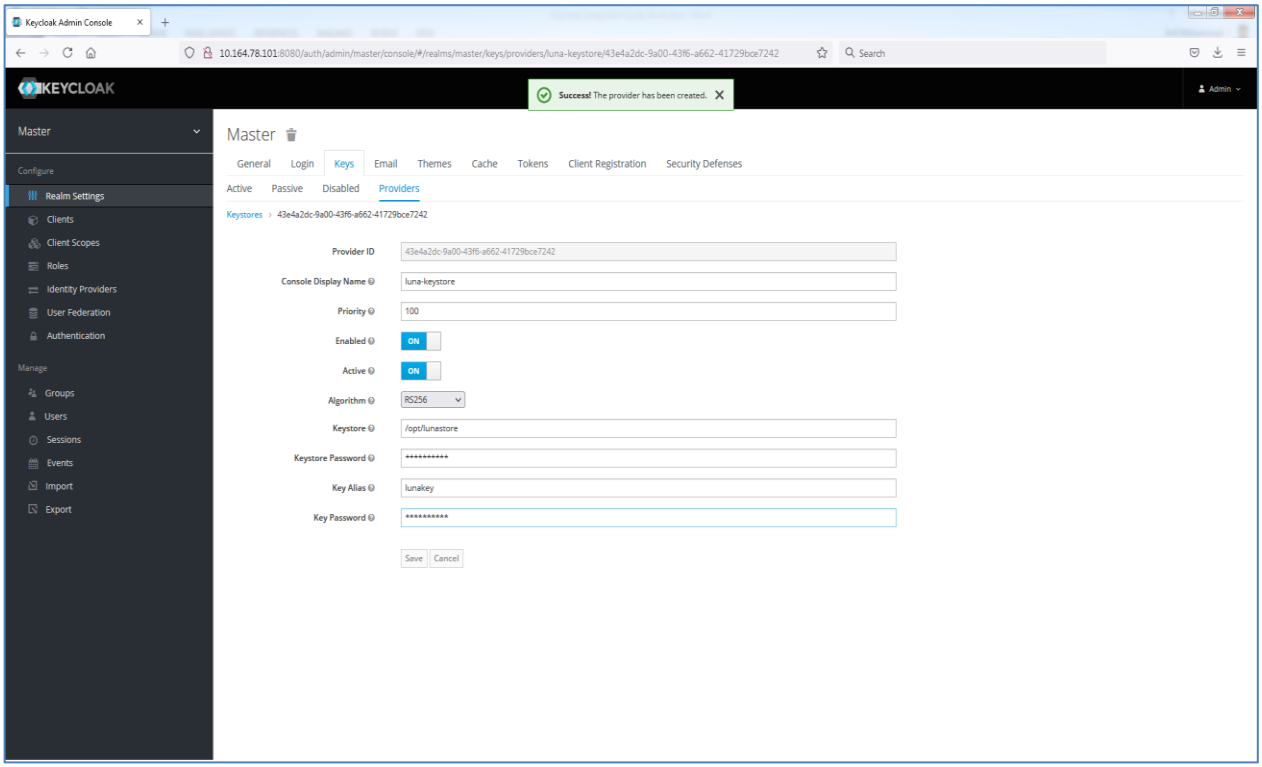
KeyStore Password = Partition CO password

Key Alias = Label of the key generated on Luna HSM

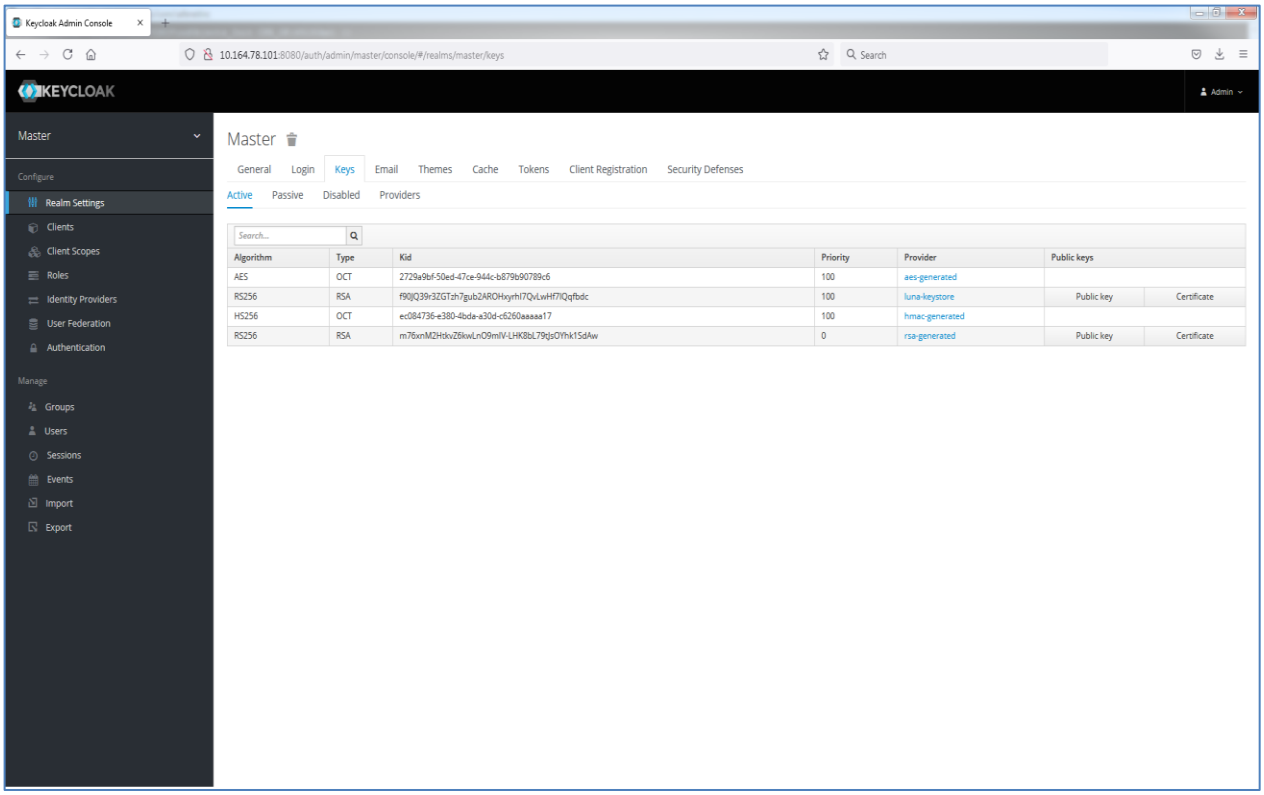
Key Password = key password set while generating the key pair



**19.** You will see the following message when the provider has been created:




**20.** Navigate to **Realm Settings** > **Keys** and verify that **luna-keystore** Provider key is listed with the highest priority (100) in the list.



**21.** Stop and start the Keycloak server again and log in to the Keycloak admin console.

```
# /opt/keycloak/bin/standalone.sh
```

For every login session, token will be signed by Luna HSM generated key. You can now create User, Roles, and Client and when you generate the Authentication Token, it will be signed by Realm Signing Key available on HSM.

If your Luna HSM is not available or if the NTLS is not running, you will not able to login to admin console because Keycloak will not get the signing keys from Luna HSM.

This completes the integration of Keycloak with Luna HSM, enabling you to securely store the signing key and certificate on the Luna HSM.

# Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, 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.