EJBCA

Integration Guide

gemalto
security to be free

Document Number: 007-013323-001, Rev. J

Release Date: May 2018

Contents

Preface

This document is intended to guide administrators through the steps for EJBCA and SafeNet Luna HSM integration. This guide provides the necessary information to install, configure, and integrate EJBCA with SafeNet Luna Hardware Security Modules (HSM).

Scope

This guide provides instructions for setting up a small test lab with EJBCA running with SafeNet Luna HSM for securing the private keys of CA. It explains how to install and configure the software that is required for setting up EJBCA while storing private key on SafeNet Luna HSM.

Document Conventions

This section provides information on the conventions used in this template.

Notes

Notes are used to alert you to important or helpful information. These elements use the following format:

NOTE: Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. These elements use the following format:

CAUTION: Exercise caution. Caution alerts contain important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. These elements use the following format:

WARNING: Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.

Command Syntax and Typeface Conventions

Convention	Description
bold	The bold attribute is used to indicate the following: • Command-line commands and options (Type **dir /p**.) • Button names (Click **Save As**.) • Check box and radio button names (Select the **Print Duplex** check box.) • Window titles (On the **Protect Document** window, click **Yes**.) • Field names (**User Name:** Enter the name of the user.) • Menu names (On the **File** menu, click **Save**.) (Click **Menu** > **Go To** > **Folders**.) • User input (In the **Date** box, type **April 1**.)
italic	The italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)
`Consolas`	Denotes syntax, prompts, and code examples.

Support Contacts

Contact Method	Contact Information	
Address	Gemalto 4690 Millennium Drive Belcamp, Maryland 21017, USA	
Phone	US	1-800-545-6608
	International	1-410-931-7520
Technical Support Customer Portal	https://supportportal.gemalto.com Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the Gemalto Knowledge Base.	

1
Introduction

Overview

SafeNet Luna HSM integrates with EJBCA to provide significant performance improvements by off-loading cryptographic operations from the server to HSM. In addition, SafeNet Luna HSM provides extra security by protecting the CA private keys within a FIPS 140-2 certified hardware security module.

This integration between SafeNet Luna HSM and EJBCA uses the industry standard PKCS#11 interface. EJBCA generates the 2048 bit RSA keys on SafeNet Luna HSM and it is used by the CA for Certificate and CRL signing.

The installation is performed in several steps:

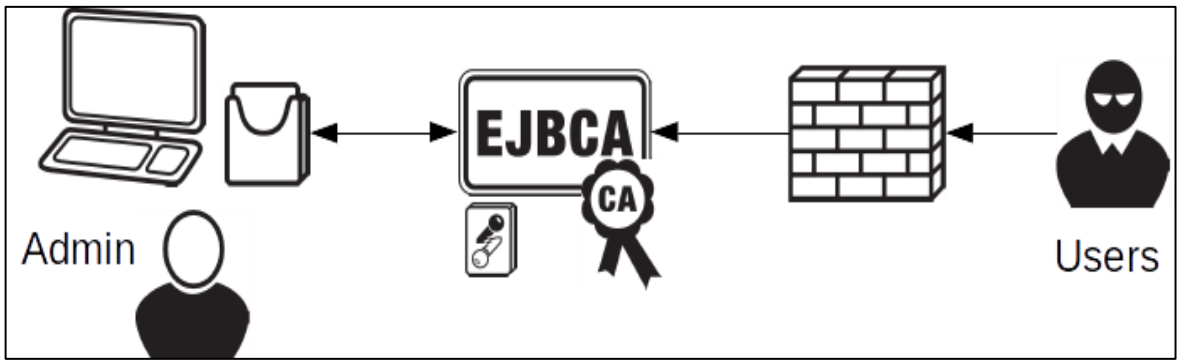
- Install and configure SafeNet Luna HSM.
- Install and configure EJBCA using SafeNet Luna HSM.

Understanding the EJBCA

EJBCA is an enterprise class PKI Certificate Authority software, built using Java (JEE) technology. It is a robust, high performance, platform independent, flexible, and component based CA to be used stand-alone or integrated with other applications.

EJBCA is today one of the most widely deployed PKI software in the world. It is used in mission critical production environments by governments, corporations, and individuals all around the world. There are EJBCA users on all inhabited continents, and EJBCA is one of the most flexible, performant and scalable PKIs used by many organizations.

The following diagram shows a good example setup of a secured CA that receives certificate requests.



3rd Party Application Details

- EJBCA 6.4.0
- EJBCA 6.3.1.1
- JBOSS AS 7.1.1

EJBCA can be downloaded from PrimeKey Support site:

http://www.ejbca.org/download.html

And JBOSS from the official site of JBOSS Application Server:

http://www.jboss.org/jbossas/downloads/

Supported Platforms

SafeNet Luna HSM (v7.x)

The following platforms are tested with SafeNet Luna HSM:

Operating Systems	SafeNet Luna HSM	EJBCA	JBOSS AS	Java Version
Red Hat Enterprise Linux 7.0(64-bit)	Luna Network HSM Appliance Software v7.2.0 Firmware v7.2.0 Luna HSM Client 7.2.0	6.4.0	7.1.1	Open JDK 7
Red Hat Enterprise Linux 7.0(64-bit)	Luna Network HSM Appliance Software v7.1.0 Firmware v7.1.0 Luna HSM Client 7.1.0	6.4.0	7.1.1	Open JDK 7
Red Hat Enterprise Linux 6.5(64-bit)	Luna Network HSM Appliance Software v7.0.0 Firmware v7.0.1 Luna HSM Client 7.0.0	6.4.0	7.1.1	Open JDK 7

SafeNet Luna HSM (v5.x/6.x)

The following platforms are tested with SafeNet Luna HSM:

Operating Systems	SafeNet HSM	EJBCA	JBOSS AS	Java Version
Red Hat Enterprise Linux 6.5 (64 bit)	Luna SA Appliance Software v6.3.0 Firmware 6.10.9 & 6.27.0 Luna Client 6.3.0	6.4.0	7.1.1	Open JDK 7
Red Hat Enterprise Linux 6.5 (64 bit)	Luna SA Appliance Software v6.2.2 Firmware 6.10.9 & 6.24.3 Luna Client 6.2.2	6.4.0	7.1.1	Open JDK 7
Red Hat Enterprise Linux 6.5 (64 bit)	Luna SA Appliance Software v6.2.1 Firmware 6.10.9 Luna Client 6.2.1	6.4.0	7.1.1	Open JDK 7
Red Hat Enterprise Linux 6.5 (64 bit)	Luna SA Appliance Software v6.2.1 Firmware 6.10.9 & 6.24.2 Luna Client 6.2.1	6.3.1.1	7.1.1	Open JDK 7
Red Hat Enterprise Linux 6.5 (64 bit)	Luna SA Appliance Software v6.1 Firmware 6.10.9 Luna Client 6.1	6.3.1.1	7.1.1	Open JDK 7
Red Hat Enterprise Linux 6.5 (64 bit)	Luna SA Appliance Software v5.4.7 Firmware 6.10.9 Luna Client 5.4.1	6.3.1.1	7.1.1	Open JDK 6

NOTE: EJBCA is tested with Luna Clients in HA & FIPS Mode also.

Prerequisites

Configuring PED Auth SafeNet Luna HSM (v6.1/v7.0)

You need to obtain the following patch to work with PED based SafeNet Luna HSM when using the version 6.1 and 7.0:

DOC ID: DOW4166

Part No: 630-010467-001 Alpha3

TITLE: Luna 5 Compatibility Shim for Luna 6

> **NOTE:** The below configuration is only applicable for Version 6.1/7.0 of PED based SafeNet Luna HSM.

CONFIGURATION:

1. Copy the libshim.so to <lunaclient installation>/lib directory. It is advised to first rename the previous shim.
2. Point the application to the libshim.so instead of the Cryptoki shared object library. To point it, open the `/etc/Chrystoki.conf` file and make the following changes:

   For Linux,

```
Chrystoki2 = {
   LibUNIX64 = /usr/safenet/lunaclient/lib/libshim.so;
}
Shim2 = {
   LibUNIX64 = /usr/safenet/lunaclient/lib/libCryptoki2_64.so;
}
 Misc = {
  :
ApplicationInstance=SA5_COMPATIBILITY;
FunctionBindLevel=2;
 :
}
```

Contact Customer Support if you need assistance regarding the above configuration.

Configuring PED Auth SafeNet Luna HSM (v6.2.x/v6.3.0/v7.x)

For Ped based SafeNet Luna HSM make sure ProtectedAuthenticationPathFlagStatus is set to '1' in Misc Section of Chrystoki.conf (Linux) file.

For Linux,

```
Misc = {
ProtectedAuthenticationPathFlagStatus = 1;
}
```

Configuring SafeNet Luna Network HSM 7.x

SafeNet Luna Network HSM allows to create Per-Partition Security Officer (PPSO) partition. HSM Administrator is not Security Officer (SO) for PPSO partitions. The HSM SO/Administrator elects to create a partition as PPSO-type, which creates an empty structure that is handed to the new owner, who initializes the partition to create the Partition Security Officer (PSO) role or identity for management functions. The PSO in turn creates the partition Crypto Officer (CO) to control client cryptographic operations on the partition.

Refer to the SafeNet Luna HSM documentation for installation steps and details regarding the configuration and setup of the box on UNIX/Windows systems. Before you get started ensure the following:

- SafeNet Luna Network HSM appliance and a secure admin password.
- SafeNet Luna Network HSM, and a hostname, suitable for your network.
- SafeNet Luna Network HSM network parameters are set to work with your network.
- Initialize the HSM on the SafeNet Luna Network HSM appliance.
- Create and exchange certificates between the SafeNet Luna Network HSM and your Client system.
- Create a partition on the HSM that will be later used by EJBCA.
- Register the Client with the partition. And run the "vtl verify" command on the client system to display a partition from SafeNet Luna HSM. The general form of command is "`C:\Program Files\SafeNet\LunaClient> vtl verify`" for Windows and "`/usr/safenet/lunaclient/bin/vtl verify`" for Unix.
- Initialize the Partition as mentioned in steps below for Password/PED based respectively
- Enabled Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to SafeNet Luna Network HSM with Trusted Path Authentication [which is FIPS 140-2 level 3] only).

Initialize the Partition SO and Crypto Officer Roles on a PW-Auth Partition

These instructions assume a password-authenticated SafeNet Luna Network HSM that has been initialized, and an application partition has been created, capable of having its own Security Officer.

- **Initialize the Partition SO role**
  Set the active slot to the created, uninitialized, application partition.
  Type **slot set -slot <slot number>**

```
lunacm:> slot set -slot 0
Current Slot Id:    0     (Luna User Slot 7.0.0 (Password) Signing With Cloning Mode)
Command Result : No Error
```

  Initialize the application partition, to create the partition's Security Officer (SO).
  Type **partition init -label <part_label>**

```
lunacm:> par init -label <part_label> –password <part_password>
You are about to initialize the partition.
All partition objects will be destroyed.
Are you sure you wish to continue?
Type 'proceed' to continue, or 'quit' to quit now -> proceed
Command Result: No Error
```

- **Initialize the Crypto Officer role**

  a. The SO of the application partition can now assign the first operational role within the new partition.
  Type **role login -name Partition SO**.

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

  b. Type **role init -name Crypto Officer**.

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

  c. The application partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. Therefore, the SO must log out to allow the Crypto Officer to log in.
  Type **role logout**.

```
lunacm:> role logout
```

Initialize the Partition SO and Crypto Officer Roles on a PED-Auth Partition

These instructions assume a PED-authenticated SafeNet Luna Network HSM that has been initialized, and an application partition has been created, capable of having its own Security Officer.

Take the following steps to initialize the PSO and CO roles:

- **Initialize the Partition SO role**

  Set the active slot to the created, uninitialized, application partition.
  Type **slot set -slot <slot number>**

  ```
  lunacm:> slot set -slot 0
  Current Slot Id:    0     (Luna User Slot 7.0.0 (PED) Signing With Cloning Mode)
   Command Result : No Error
  ```

  Initialize the application partition, to create the partition's Security Officer (SO).
  Type **partition init -label <part_label>**

  ```
  lunacm:> par init -label <part_label>
  You are about to initialize the partition.
  All partition objects will be destroyed.
  Are you sure you wish to continue?
  Type 'proceed' to continue, or 'quit' to quit now -> proceed
  Please attend to the PED.
   Respond to SafeNet PED prompts...
   Command Result : No Error
  ```

- **Initialize the Crypto Officer role**

  The SO of the application partition can now assign the first operational role within the new partition.

  Type **role login -name Partition SO**.

  Type **role init -name Crypto Officer**.

  ```
  lunacm:> role init -name Crypto Officer
  Please attend to the PED.
  Respond to SafeNet PED prompts...
  Command Result: No Error
  ```

The application partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. Therefore, the SO must log out to allow the Crypto Officer to log in.
Type **role logout**.

Now, the Crypto Officer, or an application using the CO's challenge secret/password can perform cryptographic operations in the partition, as soon as the Crypto Officer logs in with **role login -name Crypto Officer**. However, the Crypto Officer can create, modify and delete crypto objects within the partition, in addition to merely using existing crypto objects (sign/verify). You can also create a limited-capability role called Crypto User that can use the objects created by the Crypto Officer, but cannot modify them.

NOTE: The black Crypto Officer PED key/Crypto Officer Password (in case of PW-Auth) is valid for the initial login only. You must change the initial credential on the key using the command **role changepw** during the initial login session, or a subsequent login. Failing to change the credential will result in a CKR_PIN_EXPIRED error while performing role-dependent actions.

Controlling User Access to the HSM

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 your client software while retaining your hsmusers group configuration.

Adding users 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. Users you add to the hsmusers group are able to access the HSM. Users who are not part of the hsmusers group are not able to access the HSM.

- **Adding a user to hsmusers group**
  a. Ensure that you have **sudo** privileges on the client workstation.
  b. 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 users from hsmusers group

To revoke a user's access to the HSM, you can remove them from the hsmusers group.

- **Removing a user from hsmusers group**
  a. Ensure that you have **sudo** privileges on the client workstation.
  b. 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.

Configuring SafeNet Luna Network HSM (v5.x/6.x)

Refer to the SafeNet Luna HSM documentation for installation steps and details regarding the configuration and setup of the box on UNIX systems. Before you get started ensure the following:

- SafeNet Luna Network HSM appliance and a secure admin password
- SafeNet Luna Network HSM, and a hostname, suitable for your network
- SafeNet Luna Network HSM network parameters are set to work with your network
- Initialize the HSM on the SafeNet Luna Network HSM appliance.
- Create and exchange certificates between the SafeNet Luna Network HSM and your Client system.
- Create a partition on the HSM, remember the partition password that will be later used by EJBCA.
- Register the Client with the partition. And run the "vtl verify" command on the client system to display a partition from SafeNet Luna Network HSM. The general form of command is `"C:\Program`

`Files\SafeNet\LunaClient> vtl verify`" for Windows and "`/usr/safenet/lunaclient/bin/vtl verify`" for Unix.

- Enabled Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to SafeNet Luna Network HSM with Trusted Path Authentication [which is FIPS 140-2 level 3] only).

Using Luna 6.x/7.x 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 operation in a FIPS-compliant HSM. If you are using the SafeNet Luna HSM in FIPS mode, you have to make the following change in configuration file:

```
Misc = {
   RSAKeyGenMechRemap = 1;
}
```

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

NOTE: The above configuration is valid for Luna 7.x and Luna 6.x (F/W Version 6.22.0 and above only).

Also please make the following changes in `Chrystoki.conf` for Luna Client 6.x/7.x to list the registered partition with Slot ID 1 instead of 0:

```
Presentation = {
   OneBaseSlotId =1;
}
```

EJBCA Setup

Before proceeding, it is recommended to familiarize yourself with EJBCA. Refer to the EJBCA documentation for more information to install and pre-installation requirements.

https://www.ejbca.org/docs/installation.html

EJBCA must be installed on the target machine to carry on with the integration process. Details of the target machine are as follows:

- An RHEL machine required to setup an EJBCA Certification Authority.

The machine utilized is denoted in the setup is as follows:

- **ca.example.com:** EJBCA Certificate Authority
- Set the ca.example.com at the first line in `/etc/hosts` file.

2

Integrating SafeNet Luna HSM with EJBCA

SafeNet Luna HSM with EJBCA

To set up SafeNet Luna HSM for EJBCA, perform the following steps:

Configuring, Installing, and Deploying the EJBCA

For running the EJBCA as Certificate Authority, several software requirements are needed to be met. Below is the list of required software:

- Open JDK 6 or Open JDK 7
- Apache Ant Build Tool
- JBoss Server
- My SQL
- My SQL JDBC Driver

Download following software from the URL provided below on ca.example.com server.

- Apache Ant Build Tool: http://archive.apache.org/dist/ant/binaries/
- JBoss Server: http://jbossas.jboss.org/downloads
- EJBCA: https://www.ejbca.org/download.html

  After downloading all the software, you need to unzip EJBCA, JBOSS, and ANT in the /opt/ directory. To do this, you need to execute the following commands on the terminal:

  ```
  # unzip /home/apache-ant-1.9.6-bin.zip -d /opt/
  # unzip /home/jboss-as-7.1.1.Final.zip -d /opt/
  # unzip /home/ejbca_ce_6_3_1_1.zip -d /opt/
  ```

  After unzip rename the directories as the following for convenience:

  ```
  # mv /opt/apache-ant-1.9.6 /opt/apache-ant
  # mv /opt/jboss-as-7.1.1 /opt/jboss
  # mv /opt/ejbca_ce_6_3_1_1 -d /opt/ejbca
  ```

  When the unpacking of software completed, following environment variables must be set up on ca.example.com use JAVA 7 with SafeNet Luna HSM version 6 and above:

  ```
  # export JAVA_HOME=<Path to Java JDK>
  # export PATH=$JAVA_HOME/bin:$PATH
  # export ANT_HOME=/opt/apache-ant
  # export JBOSS_HOME=/opt/jboss
  # export PATH=$JBOSS_HOME/bin:$PATH
  # export APPSRV_HOME=$JBOSS_HOME
  ```

```
# export PATH=$ANT_HOME/bin:$PATH
# export EJBCA_HOME=/opt/ejbca
# export CLASSPATH=$JAVA_HOME/jre/lib/ext:$CLASSPATH
```

Configuring the PKCS11 Provider on EJBCA

To setup PKCS11 provider on ca.example.com server, perform the following steps on the terminal:

1. Create the Luna configuration file with the contents provided below and refer the configuration file to use with the PKCS11 provider.

   ```
   # vi $JAVA_HOME/jre/lib/security/luna.cfg
   ```

   Enter the following contents:

```
    #SafeNet Luna
    name = Luna
    library = /usr/safenet/lunaclient/lib/libCryptoki2_64.so
    description = Luna config
    slot = 1
    attributes(*,*,*) = {
    CKA_TOKEN = true
    }
    attributes(*,CKO_SECRET_KEY,*) = {
    CKA_CLASS=4
    CKA_PRIVATE= true
    CKA_KEY_TYPE = 21
    CKA_SENSITIVE= true
    CKA_ENCRYPT= true
    CKA_DECRYPT= true
    CKA_WRAP= true
    CKA_UNWRAP= true
    }
    attributes(*,CKO_PRIVATE_KEY,*) = {
    CKA_CLASS=3
    CKA_LABEL=true
    CKA_PRIVATE = true
    CKA_DECRYPT=true
    CKA_SIGN=true
    CKA_UNWRAP=true
    }
    attributes(*,CKO_PUBLIC_KEY,*) = {
    CKA_CLASS=2
    CKA_LABEL=true
    CKA_ENCRYPT = true
    CKA_VERIFY=true
    CKA_WRAP=true
}
```

2. Modify the `java.security` file to include the PKCS11 Provider. Open the `java.security` file and do the following changes:

For Java 6:

```
# vi $JAVA_HOME/jre/lib/security/java.security
```

```
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=sun.security.pkcs11.SunPKCS11 ${java.home}/lib/security/luna.cfg
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
```

For Java 7:

```
# vi $JAVA_HOME/jre/lib/security/java.security
```

```
security.provider.1=sun.security.pkcs11.SunPKCS11 ${java.home}/lib/security/nss.cfg
security.provider.2=sun.security.provider.Sun
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=sun.security.pkcs11.SunPKCS11 ${java.home}/lib/security/luna.cfg
security.provider.7=com.sun.crypto.provider.SunJCE
security.provider.8=sun.security.jgss.SunProvider
security.provider.9=com.sun.security.sasl.Provider
security.provider.10=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.11=sun.security.smartcardio.SunPCSC
```

3. Ensure that `nss.cfg` file has the following entry

   ```
   nssLibraryDirectory = /usr/lib64
   ```

4. Change the other security providers' preference accordingly and save the file.

   Ensure that the PKCS11 provider jar must be present on the "`$JAVA_HOME/jre/lib/ext`" location.

Generate the keys for EJBCA

The tool "`EJBCA_HOME/dist/clientToolBox/ejbcaClientToolBox.sh PKCS11HSMKeyTool`" is used to administrate and generate keys. Perform the following steps to generate the keys on SafeNet Luna HSM using PKCS11 Provider:

```
# cd $EJBCA_HOME

# ant clientToolBox

# dist/clientToolBox/ejbcaClientToolBox.sh PKCS11HSMKeyTool generate
/usr/safenet/lunaclient/lib/libCryptoki2_64.so 2048 signKey 1

# dist/clientToolBox/ejbcaClientToolBox.sh PKCS11HSMKeyTool generate
/usr/safenet/lunaclient/lib/libCryptoki2_64.so 2048 defaultKey 1

# dist/clientToolBox/ejbcaClientToolBox.sh PKCS11HSMKeyTool generate
/usr/safenet/lunaclient/lib/libCryptoki2_64.so 2048 myKey 1
```

When you execute the above commands it will prompt for token password, enter the SafeNet Luna HSM partition password and wait till the keys are generated. These keys are used to create the initial Admin CA, Root CA, and Server CA.

To test the keys on the HSM that will be used by EJBCA, use the following command and enter the partition password if prompted:

```
# dist/clientToolBox/ejbcaClientToolBox.sh PKCS11HSMKeyTool test
/usr/safenet/lunaclient/lib/libCryptoki2_64.so 1
```

NOTE: Slot id is 1 and libCryptoki2_64.so is the SafeNet Luna HSM library.

Installing Required Software Packages

It is required to install the MySQL Server and MySQL JDBC Driver before installing the EJBCA. But if your server is not registered with official RHN repositories then you need to attach your Redhat installation DVD as a local repository. To do so, perform the following steps:

```
# yum repolist
# mount | grep iso9660
# vi /etc/yum.repos.d/RHEL_6.5_Disc.repo
```

1. Enter the following contents at the end of the file:

```
[RHEL_6.5_Disc]
name=RHEL_6.5_x86_64_Disc
baseurl="file:///media/RHEL_6.5 x86_64 Disc 1/"
gpgcheck=0
```

2. Verify that your repolist will show some entry:

```
# yum repolist
```

3. Install the MySQL Server and MySQL JDBC

```
# yum install mysql-server
# yum install mysql-connector-java
```

Setting up MySQL Server for EJBCA

It is a good idea to make MySQL use UTF-8 all the time. This can save a lot of trouble if you end-up adding non-latin characters in subject DN's or anywhere else in the EJBCA front-end. Edit the following configuration file:

```
# vi /etc/my.cnf
```

1. Enter the following contents at the end of the file:

```
[client]
default-character-set=utf8

[mysqld]
default-character-set=utf8
default-collation=utf8_unicode_ci
character-set-server=utf8
init-connect='SET NAMES utf8'
character-set-client = utf8
```

NOTE: This configuration setting is required for RHEL 6.5.

2. After this you need to start the MySQL server to apply the changes:

```
# service mysqld start
```

Once the MySQL server has been installed and set-up to use UTF-8, it is necessary to create the database to store EJBCA data, as well as to create and grant appropriate permissions to the user, used for accessing the database:

```
# mysql -u root -p
  mysql>  create database ejbca;
  mysql> grant all privileges on ejbca.* to 'ejbca'@'localhost' identified by 'ejbca';
  mysql> flush privileges;
  mysql> exit;
```

3. After setting the user ejbca and password 'ejbca' restart the MySQL, you can replace the username and password as per your choice:

```
# service mysqld restart
```

4. Verify that ejbca user is able to log in to mysql user and test their access on the database:

```
# mysql -u ejbca -p
  mysql> use ejbca;
  mysql> show grants for ejbca@localhost;
  mysql> exit;
```

Creating the User Accounts

Create a user account to run JBOSS and EJBCA.

```
# adduser ejbca
```

```
# passwd ejbca
```

When prompted for the password, enter the password for the ejbca user and note down the password as it is required when the user needs to log in to the ca.example.com server.

Installing JBOSS

Install JBOSS. No need to configure every detail (no mail, default logging), but enough details to get the platform running and tweaked the way EJBCA needs for installation.

1. Tweak the JBOSS configuration by enabling certain security functions that EJBCA requires.

```
# cd $JBOSS_HOME/modules/sun/jdk/main
# vi module.xml
```

2. Add the following entries to the system export paths:

```
<path name="sun/security/x509"/>
<path name="sun/security/pkcs11"/>
<path name="sun/security/pkcs11/wrapper"/>
<path name="sun/security/action"/>
```

3. Now, create the directory that will hold JBOSS' link to mysql-connector-java.jar, and the link itself:

```
# mkdir -p $JBOSS_HOME/modules/com/mysql/main
# cd $JBOSS_HOME/modules/com/mysql/main
# ln -s /usr/share/java/mysql-connector-java.jar mysql-connector-java.jar
```

4. Now, build the module.xml file that describes the connector.

```
# vi module.xml
```

5. Add the following in to the module.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
   <resources>
      <resource-root path="mysql-connector-java.jar"/>
   </resources>
      <dependencies>
         <module name="javax.api"/>
         <module name="javax.transaction.api"/>
       </dependencies>
 </module>
```

6. Now first make the ejbca user as the owner of the JBOSS directory tree and then start the JBOSS Server.

```
# chown -R ejbca:ejbca /opt/jboss/
```

7. Open a new terminal and logged in as ejbca user and export the environment variables defined in the Configuring Installing and Deploying the EJBCA section.

```
# cd $JBOSS_HOME/bin

# ./standalone.sh
```

8. When the JBOSS successfully started ensure that something like this at the end:

```
11:12:00,514 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss AS
7.1.1.Final "Brontes" started in 6329ms - Started 133 of 208 services (74 services are
passive or on-demand)
```

Now when the JBOSS service is running, Enable MySQL connector using the JBOSS command line interface, which will update the configuration of the standalone instance. But before making any changes, first backup the configuration file:

```
# cd $JBOSS_HOME/standalone/configuration

# cp standalone.xml standalone.xml.initial
```

9. Run registration command from the jboss CLI (the small text is a single line):

```
# cd $JBOSS_HOME/bin

# sh jboss-cli.sh

connect

/subsystem=datasources/jdbc-driver=com.mysql.jdbc.Driver:add(driver-
name=com.mysql.jdbc.Driver,driver-module-name=com.mysql,driver-xa-datasource-class-
name=com.mysql.jdbc.jdbc.jdbc2.optional.MysqlXADataSource)

exit
```

This cli action defines our MySQL driver in `/opt/jboss-as-7.1.1.Final/standalone/configuration/standalone.xml`, and then reloads JBOSS.

If changes have been successful, following message displays in the console logs of JBOSS:

```
11:16:18,349 INFO  [org.jboss.as.connector.subsystems.datasources] (ServerService Thread
Pool -- 27) JBAS010404: Deploying non-JDBC-compliant driver class com.mysql.jdbc.Driver
(version 5.1)
```

By default, the standalone instance is defined with an h2/hsqldb database connector, and an example database. If left unchanged, EJBCA is preconfigured to use it for example purposes. It is not required and you need to disable it in the standalone.xml configuration file.

```
# vi $JBOSS_HOME/standalone/configuration/standalone.xml
```

10. Remove the following:

```
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS"
enabled="true"                                          use-java-context="true">
   <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
      <driver>h2</driver>
      <security>
         <user-name>sa</user-name>
         <password>sa</password>
      </security>
</datasource>
```

Also remove:

```
<driver name="h2" module="com.h2database.h2">
   <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
</driver>
```

Now, watch the console log after restarting JBOSS, you should no longer see:

```
11:16:18,156 INFO  [org.jboss.as.connector.subsystems.datasources] (ServerService Thread
Pool -- 27) JBAS010403: Deploying JDBC-compliant driver class org.h2.Driver (version 1.3)
```

But you should continue to see:

```
11:19:25,098 INFO  [org.jboss.as.connector.subsystems.datasources] (ServerService Thread
Pool -- 27) JBAS010404: Deploying non-JDBC-compliant driver class com.mysql.jdbc.Driver
(version 5.1)
```

Preparing EJBCA Installation Files

You need to set up the configuration files for the EJBCA. The configuration file samples, with plenty of comments describing various options, can be found within the "`$EJBCA_HOME/conf/`" directory.

1. First copy the sample files which required for EJBCA installation and make the changes as described:

   ```
   # cd $EJBCA_HOME/conf
   ```

2. Setup the ejbca configuration file:

   ```
   # cp ejbca.properties.sample ejbca.properties
   ```

   Make the following changes in `ejbca.properties` and save the file:

   ```
   # Application server home directory used during development.
   appserver.home=/opt/jboss
   # Which application server is used?
   appserver.type=jboss
   # EJBCA instance.
   ejbca.productionmode=ca
   ```

3. Setup the database configuration file:

   ```
   # cp database.properties.sample database.properties
   ```

Make the following changes in `database.properties` and save the file:

```
# JNDI name of the DataSource used for EJBCA's database access.
datasource.jndi-name=EjbcaDS
# The database name selected for deployment, used to copy XDoclet merge files.
database.name=mysql
# Database connection URL.
database.url=jdbc:mysql://127.0.0.1:3306/ejbca?characterEncoding=UTF-8
# JDBC driver classname.
database.driver=com.mysql.jdbc.Driver
# Database username.
database.username=ejbca
# Database password.
database.password=ejbca
```

NOTE: For RHEL 7, edit the parameter database.url with below entry database.url=jdbc:mysql://127.0.0.1:3306/ejbca? And remove characterEncoding=UTF-8

4. Setup the install configuration file:

```
# cp install.properties.sample install.properties
```

Make the following changes in `install.properties` and save the file:

```
# Enter a short name for the administrative CA.
ca.name=AdminCA1
# The Distinguished Name of the administrative CA.
ca.dn=CN=AdminCA1,O=EJBCA Sample,C=SE
# The token type the administrative CA will use.
ca.tokentype=org.cesecore.keys.token.PKCS11CryptoToken
# Password for the administrative CA token.
ca.tokenpassword=<Partition_password>
# Configuration file were you define key name, password and key alias for the HSM
ca.tokenproperties=/opt/ejbca/conf/catoken.properties
# The keyspec for the administrative CAs key.
ca.keyspec=2048
# The keytype for the administrative CA, can be RSA, ECDSA or DSA
ca.keytype=RSA
# Default signing algorithm for the administrative CA.
ca.signaturealgorithm=SHA256WithRSA
```

```
# The validity in days for the administrative CA, only digits.
ca.validity=3650
# The policy id of the administrative CA. Policy id determines which PKI policy the CA
uses.
ca.policy=null
```

5. Setup the catoken configuration file:

```
# cp catoken.properties.sample catoken.properties
```

Make the following changes in `catoken.properties` and save the file:

```
# Configuration file were you define key name, password and key alias for the HSM.
sharedLibrary=/usr/safenet/lunaclient/lib/libCryptoki2_64.so
slotLabelType=SLOT_NUMBER
slotLabelValue=1
pin=userpin1
certSignKey=signKey
crlSignKey=signKey
defaultKey=signKey
```

6. Setup the web configuration file:

```
# cp web.properties.sample web.properties
```

Make the following changes in `web.properties` and save the file:

```
# Password for java trust keystore (p12/truststore.jks).
java.trustpassword=changeit
# The CN and DN of the super administrator.
superadmin.cn=SuperAdmin
superadmin.dn=CN=${superadmin.cn},O=EJBCA Sample,C=SE
# The password used to protect the generated super administrator P12 keystore.
superadmin.password=ejbca
# Set this to false if you want to fetch the certificate from the EJBCA public web
# pages, instead of importing the P12-keystore. This can be used to put the initial
# superadmin-certificate on a smart card.
superadmin.batch=false
# The password used to protect the web servers SSL keystore.
httpsserver.password=serverpwd
# The CA servers DNS host name, must exist on client using the admin GUI.
httpsserver.hostname=ca.example.com
# The Distinguished Name of the SSL server certificate used by the administrative web gui.
httpsserver.dn=CN=${httpsserver.hostname},O=EJBCA Sample,C=SE
```

NOTE: The configuration setting is done here for the objective of this guide, change these settings according to your environment.

Installing EJBCA

The installation itself is more or less a straightforward process. The ejbca user must be the owner of both the JBOSS directory tree and the EJBCA directory tree. Before running our initial deployment, it's a requirement to ensure that this is the true.

```
# chown -R ejbca:ejbca /opt/jboss
# chown -R ejbca:ejbca /opt/ejbca
```

It is necessary to run the JBoss AS instance in order to get on with the next step. To perform this task, open a new terminal on ca.example.com and export the environment variables defined in the Configuring Installing and Deploying the EJBCA section.

```
# cd $JBOSS_HOME/bin
# ./standalone.sh
```

Once the server has started up, the following line displays at the end:

```
14:20:49,326 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss AS 7.1.1.Final
"Brontes" started in 5907ms - Started 130 of 204 services (74 services are passive or on-demand)
```

Ensure that the server starts without any error. See the server logs of JBOSS Server at the following location: `"$JBOSS_HOME/server/default/log/server.log"`

Once the instance is up and running, proceeds with the installation of EJBCA. Open a new terminal and log in as ejbca user on ca.example.com server and export the environment variables defined in the Configuring Installing and Deploying the EJBCA section.

```
# cd $EJBCA_HOME
# ant deploy
```

BUILD SUCCESSFUL message displays when deployment completed successfully. The deployment command might take a while, after deployment has finished wait for the JBOSS to complete deployment.

Once the server has started up, the following line displays:

```
14:33:26,946 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed
"ejbca.ear"
```

Finalize the deployment with following command on the EJBCA terminal:

```
# ant install
```

BUILD SUCCESSFUL message displayed when installation completed successfully. Once the installation has finished, start the JBOSS again with the following command on the JBOSS terminal:

```
# ./standalone.sh
```

Importing Super-Administrator Token

Since EJBCA is protecting the access to administration segment through client certificates, it is necessary to import the super-administrator token into web browser. The certificate should be installed from the EJBCA web server to workstation which will be used for further configuration on CA server.

1. Open the Mozilla Firefox web browser to access the EJBCA web page and enter the following URL:

   `http://<hostname/IP address>:8080/ejbca`

2. When the EJBCA public web page opened, click on the **Create Browser Certificate** under **Enroll** section.

3. Enter the Username and Password in Authentication section and click **OK**.

> **NOTE:** Username and Password would be superadmin and ejbca respectively, if you have set it according to this guide.

4. Click the **Enroll** button under Options section on the EJBCA Certificate Enrollment page. It imports the certificate and this certificate will be used for communicating the EJBCA for Administrative settings.
5. Once this has been completed, it should be possible to access the administration interface of the EJBCA by clicking the **Administration**.

Enable Key Recovery

An important aspect when generating private keys is their secrecy and safekeeping. In case of private keys which should be used for non-repudiation, these keys should not be backed-up.

On the other hand, in case of private keys which are used for encryption, it is essential to maintain copies of those keys. If the keys get lost, the data encrypted with them is rendered useless.

In order to allow key recovery, go to the **Administration** > **System Configuration** page and activate the **Enable Key Recovery** option by selecting the **Activate** check box. Save the setting by clicking the **Save** button.

Create Root CA

Ensure that the PKCS#11 token using SafeNet Luna HSM has been created successfully. Click the **Crypto Tokens** and verify that the PKCS#11 token is listed under the **Manage Crypto Tokens.** Also ensure that it displays the SafeNet PKCS#11 library along with the Slot ID. You need to verify that it is in activated and used state.

> **NOTE:** AdminCA1 is the name of the Crypto Token as per the settings mentioned in this guide.

With basic installation done, it is time to set up the certification authority hierarchy. The starting point is the Root CA. Click **Certification Authorities** and enter **ExampleRootCA** as the name of new certification authority, then click the **Create** button. Make the following changes:

```
Signing Algorithm: SHA256WithRSA
Crypto Token: AdminCA1.
defaultKey=defaultKey
certSignKey=signKey
Description: Root CA for Example Inc
Subject DN: CN=ExampleRootCA,O=Example Inc,C=RS
Validity: 20y
Issuing Distribution Point on CRLs: On
Default CRL Dist. Point: Click on Generate button.
CRL Expire Period: 1y
CRL Overlap Time: 2d
```

Once information is filled, click the **Create** button. Wait until the operation finishes. Once it is completed, a new certification authority will be available in the list of Certification Authorities.

Create Sub-CA's

Open the **Certificate Profiles** page, from **List of Certificate Profiles**, click the **Clone** button against the **SUBCA** profile, enter **Example Sub-CA** in the **Name of new certificate profile** box, and click the **Create from Template** button. It creates a new certificate profile with properties copied from the SUBCA profile.

1. Select the newly created **Example Sub-CA** and click the **Edit** button. The following options for this profile should be changed:

```
Available bit lengths: 2048 bits
Validity: 15y
Allow validity override: Off
CRL Distribution Points: On
Use CA defined CRL Dist. Point: On
Available CAs: ExampleRootCA
```

2. Click **Save** to commit the changes.
3. The next CA in line is the certification authority which will issue certificates for servers. Open the **Certification Authorities** page, and enter **ExampleServerCA** in the **Add CA** box, then click the **Create** button. Make the following changes on the page:

```
Signing Algorithm: SHA256WithRSA
Crypto Token: AdminCA1.
defaultKey=myKey
Description: Example's CA in charge of issuing certificates for servers within the organization.
Subject DN: CN=ExampleServerCA,O=Example Inc,C=RS
Signed By: ExampleRootCA
Certificate Profile: Sub-CA
Validity (*y *mo *d) or end date of the certificate: 15y
Use Issuing Distribution Point on CRLs: On
Default CRL Dist. Point: Click on Generate button
CRL Expire Period (*y *mo *d *h *m): 14d
CRL Overlap Time (*y *mo *d *h *m): 12h
```

4. Click the **Create** button to finalize the creation of basic CA hierarchy.

Create Certificate Profiles for End Entities

The same way certificate profiles were created for Sub-CA's, it is also necessary to create certificate profiles for end entities. These profiles will be based on the default profiles provided by the EJBCA itself.

Go to the **Certificate Profiles** page and click the **Clone** button of the **SERVER** profile, enter **ExampleServer** in the **Name of new certificate profile** box, and click on the **Create from template**. Select the newly created certificate profile and click on the **Edit** button. Make the following changes to the profile:

```
Available bit lengths: 1024, 2048
CRL Distribution Points: On
Use CA defined CRL Dist. Point: On
Available CAs: ExampleServerCA
```

Click **Save**. This concludes the creation of basic certificate profiles.

Create End Entity Profiles

Click the **End Entity Profiles** page and enter **Server** in the **Add Profile** text box. Click the **Add** button. Select the newly created Server profile and click the **Edit End Entity Profile** button. Add the following Subject DN Attributes and mark them all as **Required** and **Modifiable**:

```
O, Organization
```

```
C, Country (ISO 3166)
```

Change the fields of Server profile as follows:

```
Username: Server
Password: Server
Batch generation (clear text pwd storage) use: On
CN, Common name: Server
O, Organization: Example Inc
C, Country (ISO 3166): RS
Default Certificate Profile: ExampleServer
Available Certificate Profiles: ExampleServer
Default CA: ExampleServerCA
Available CAs: ExampleServerCA
Default Token: User Generated
Available Tokens: User Generated
```

Click **Save**. All the basic necessary end entity profiles are now available.

Configure Publish Queue Process Service

Once you start publishing the certificates and CRL's to remote locations it is necessary to ensure that in case of failures the certificates and CRL's do get published once the technical issues are resolved.

Go to the A**dministration** -> **Services** page. Enter **Publish Queue Process Service** in the **Add Service** box, and click **Add**. Select the newly created service, and click the **Edit Service** button. Enter the following information:

```
Select Worker: Publish Queue Process Service
Select Interval: Periodical Interval
Period: 1 minutes
Select Action: No Action
Active: On
Pin to Specific Node(s): ca.example.com
Description: Publish certificates and CRL's from the publisher queue.
```

Apply the changes by clicking on the **Save** button.

Configure CRL Updater

As a final step to installing the EJBCA, it is necessary to set-up the CRL updater. CRL updater is in charge of generating the CRL's and making sure that once they expire they get regenerated.

Go to the **Administration** -> **Services** page. Enter **CRL Updater** in the **Add Service** box, and click **Add**. Select the newly-created service, and click the **Edit Service** button. Enter the following information:

```
Select Worker: CRL Updater
CAs to Check: ExampleRootCA, ExampleServerCA
Select Interval: Periodical Interval
Period: 5 minutes
Select Action: No Action
Active: On
Pin to Specific Node(s): ca.example.com
Description: Updates the CRL's if necessary. Checks are made every 5 minutes.
```

Once the information is filled-in, click **Save**. This concludes the initial deployment, installation, and configuration of the EJBCA as certification authority using the SafeNet Luna HSM to secure the CA signing keys.