Black●Vault HSM Java Configuration Guide

© Engage Black
9565 Soquel Drive
Aptos, CA 95003
Phone +1 831.688.1021
1 877.ENGAGE4 (364.2434)
sales@engageblack.com

Version 1.1.0 June 28, 2021

Disclaimer and Warranty

Engage Black is a business unit of Engage Communication.

Engage Communications, Inc.9565 Soquel Drive Aptos, CA 95003

Phone +1 (831) 688-1021

http://www.engageblack.com/

http://www.engageinc.com/

USA, FCC

This device complies with Part 15 of the FCC rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference, and

(2) This device must accept any interference received, including interference that may cause undesired operation.

NOTE: This equipment has been tested and found to comply with the limits for a “Class B” digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in an installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient or relocate the receiving antenna

Increase the separation between the equipment and receiver

Connect the equipment into an outlet on a circuit different from that to which the receiver is connected

Consult the dealer or an experienced radio/TV technician for help

Table of Contents

1. Introduction

The BlackVault Hardware Security Module (HSM), through PKCS#11 can interact with all Java applications. The SunPKCS11 cryptographic service provider enables the HSM user to use the standard Java Cryptography Architecture (JCA)/Java Cryptography Extension(JCE) APIs with the BlackVault PKCS#11 library. This allows access to Java's security tools such as keytool, jar and jarsigner. The BlackVault HSM also integrates with Android Studio/Eclipse to enable code developers to perform code builds that are signed each time a build is successful.

2. Installation

To proceed, the following is needed:

- BlackVault HSM, initialized and configured properly (see the BlackVault HSM User Guide for more information)
- BlackVault Card Set
- BlackVault HSM Setup CD
- Java 8, 11, or 14

To use the BlackVault with Java, it is recommended to install Java 8, 11 or 14. Other versions of Java may work but can result in configuration issues. Versions 8 and 11 are the only older releases that are still maintained. For all versions, both the JRE and JDK are compatible for configuration with the BlackVault.

2.1. Defining User Slot with nss.cfg

The following installation steps, for both Linux and Windows operating systems, require use of the nss.cfg file found on the BlackVaultSetupCD's Configuration folder. The default provided nss.cfg file is set up for the BlackVault HSM User on slot 1. Only one user at a time can use Java utilities and keytool. To give access to a user that is not on slot 1, the nss.cfg file will have to be edited before completing the steps below.

The default Windows nss.cfg file should have the following contents:

```
name = "EngageBlack"

library = C:\Windows\System32\bvpkcs.dll


slot = 1


attributes(generate, *, *) = {

  CKA_TOKEN = true
  CKA_PRIVATE = true
}
```

The default Linux nss.cfg file should have the following contents:

```
name = "EngageBlack"
library = /usr/lib/libbvpkcs.so


slot = 1


attributes(generate, *, *) = {
  CKA_TOKEN = true
  CKA_PRIVATE = true
}
```

To use Java utilities and keytool with a user that is not on slot 1, edit the slot number in the nss.cfg file and complete the following installation steps.

2.2. Linux

If you are using machine with a Linux operating system, follow the instructions in this section to properly set up Java with a BlackVault HSM.

2.2.1. Java 8

1. Install Java 8
2. Set the JAVA_HOME environmental variable and add Java executables to Path
    a. Copy the path of the jre or jdk directory. It should look like: /usr/lib/jvm/java-8-(jre or jdk)/jre
    b. Edit /etc/environment as root
    c. Append JAVA_HOME=<path to jre or jdk from above>
    d. Append <same path as JAVA_HOME>/bin to the end of your Path
3. Navigate to the BlackVaultSetupCD’s Configuration folder and find the nss.cfg file
4. Update the library path in the nss.cfg file. This should be /usr/lib/libbvpkcs.so
5. Copy the nss.cfg file to the Java security folder (found here: /usr/lib/jvm/java-8-(jre or jdk)/jre/lib/security)
6. Open the java.security file as root (found here: /usr/lib/jvm/java-8-(jre or jdk)/jre/lib/security/java.security)
7. Copy and paste this statement to the end of the security provider list:

security.provider.10=sun.security.pkcs11.SunPKCS11 path to nss.cfg file

a. The path to the nss.cfg file will be /usr/lib/jvm/java-8-(jre or jdk)/(jre or jdk)/lib/security/nss.cfg

2.2.2. Java 11/14

1. Follow Java 8 install steps 1-7 with the desired version (11 or 14).
2. Copy the nss.cfg file to the Java security folder (found here: /usr/lib/jvm/(java directory for desired version)/conf/security).
3. Open the java.security file as root (found here: /usr/lib/jvm/(java directory for desired version)/conf/security/java.security)
4. Find: security.provider.12=SunPKCS11
5. On the same line as above, enter a space and then the path to the nss.cfg file.
    a. The path to the nss.cfg file will be /usr/lib/jvm/(java directory for desired version)/conf/security/nss.cfg

2.3. Windows

If you are using machine with a Windows operating system, follow the instructions in this section to properly set up Java with a BlackVault HSM.

2.3.1. Java 8

1. Install Java 8.
2. Set the Java path and environmental variable
    a. From the Desktop, right click This PC→Advanced System Setting→Environmental Variables.
    b. Select Path→New.
    c. Set variable name to the following: (the name of the JRE or JDK folder may differ. Use the browse option for correct path.)
        i. C:\Program Files\Java\(jre or jdk)\bin
3. Navigate to the BlackVaultSetupCD's Configuration folder and find the nss.cfg file.
4. Update the library path in the nss.cfg file. This should be C:\Windows\System32\bvpkcs2.dll
5. Copy the nss.cfg file to the Java security folder (found here: C:\Program Files\Java\(JRE or JDK folder)\lib\security).
6. Update the java.security file.
    a. Open Notepad as administrator.
    b. Change file type to All Files.
    c. Open the java.security file (found in C:\Program Files\Java\(JRE or JDK folder)\lib\security\java.security
    d. Copy and paste this statement to the end of the security provider list:

security.provider.11=sun.security.pkcs11.SunPKCS11 path to nss.cfg file

i. The path to the nss.cfg file will be:C:\\Program Files\\Java\\(JRE or JDK folder)\\lib\\security\\nss.cfg

2.3.2. Java 11/14

1. Follow steps 1-4 of Java 8 install for desired Java version (11 or 14).
2. Once you have located the nss.cfg file, copy it to C:\Program Files\Java\(JRE or JDK folder)\conf\security
3. Edit the java.security file
   a. Open java.security file in notepad as administrator
   b. Find: security.provider.12=SunPKCS11
   c. On the same line as above, enter a space and then the path to the nss.cfg file
   d. The nss.cfg file path should be: C:\\Program Files\\Java\\(JRE or JDK folder)\\conf\\security\\nss.cfg
4. Reboot the host for changes to take effect.

2.4. Verifying Install

To verify that the Java installation and configuration is correct, perform the following:

1. Open a command prompt on the host machine.
2. Log in to the BlackVault as user.
3. In the command prompt on the host machine, run the following command:
   keytool -keystore NONE -storetype PKCS11 -storepass 2222 -list
   a. Provided no keys have been generated this far, the output should look like:
      Keystore type: PKCS11
      Keystore provider: SunPKCS11-Engage Black Vault
      Your keystore contains 0 entries
   b. If keys have been created, they should be displayed.
   c. If this command fails, verify the host machine can communicate with the HSM via a ping and/or bvtool command. If the pkcs.dat file is not set up correctly, this command will fail.

3. Application Integration

3.1. Java Keytool and Jarsigner

To do code signing per industry best practices, along with storing the key inside a secure BlackVault HSM, a code signing certificate associated with the key is required. For the Java environment, if all the prerequisites have been competed (Java configured correctly, and PKCS#11 library installed) complete the following steps:

1. Generate a key with keytool:
   ```
   keytool -genkey -keyalg RSA -keysize 1024 -alias keyNameHere -keystore NONE -storetype PKCS11 -storepass 2222
   ```

   a. storetype is telling Java to use PKCS# 11 functions
   b. alias is the name of the key
   c. keyalg is the desired key algorithm
   d. keysize is the desired key size
   e. storepass is normally used for authenticating the keystore, but the BlackVault HSM authenticates itself and does not use this password. Java requires it anyways.
2. After entering the keytool command, Java will prompt for Identification information. This is used for the self-signed certificate that is created at the same time the key is.
3. After creating the key, verify that it is on the BlackVault:
   ```
   keytool -keystore NONE -storetype PKCS11 -storepass 2222 -list
   ```
4. Generate a certificate signing request (CSR) with the generated key:
   ```
   keytool -certreq -alias keyNameHere -keyalg RSA -file CSRNameHere.csr -keystore NONE -storetype PKCS11 -storepass 2222
   ```
5. Get the CSR signed by a Certificate Authority (i.e. BlackVault CA, Digicert, Verisign, etc)
6. After obtaining the certificate from a Certificate Authority it will need to be attached to the key from which it is derived. To do this, it must be imported into the BlackVault. Use the following bvtool command:
   ```
   bvtool importcert -i caFileHere.pem -l certLabel
   ```
7. Verify the certificate is on the BlackVault. Run the command:
   ```
   bvtool list -a
   ```
   This should display keys on the BlackVault as well as the imported certificate

Now, everything is set up to start code signing. Verify everything is configured correctly with a sample run:

1. Sign code with the following command:
   jarsigner -keystore NONE -storetype PKCS11 -storepass 2222 /path/to/jar/file keyNameHere
2. Verify code with the following command:
   jarsigner -verify /path/to/jar/file

3.2. Eclipse IDE

The BlackVault HSM's integration with Eclipse involves a straightforward modification of the Apache ant build.xml build file. In the following section, the highlighted portions of text are specific to your build environment and project. Refer to your eclipse-workspace for correct paths and names.

1. Ensure a key exists on the BlackVault that Java can use. Check with:
   keytool -keystore NONE -storetype PKCS11 -storepass 2222 -list
   Log in to the BlackVault as user and check pkcs.dat file is setup correctly.
2. If an ant build script (build.xml) does not exist for your Java project, create one:
   a. Open Eclipse and select File→Export...
   b. Under the General folder, Select Ant builders→Next
   c. Select the desired project and then Finish.
3. Edit the build.xml file so it only contains the following:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<project name=”Test.makejar” default=”makejar” basedir=”.”>
<target name=”makejar” description=”Your Jar Project”>
<jar destfile=”build/main/Test.jar”>
<manifest>
<attribute name=”Main-Class” value=”Test”/>
<attribute name=”Class-Path” value=”.”/>
</manifest>
<fileset dir=”bin”/>
</jar>
<echo message=”Code Signing started”/>
<signjar=”build/main/Test.jar” alias=”KeyNameHere”  keystore=”NONE”
Storetype=”PKCS11” storepass=”2222”/>
<java jar=”build.main/Test.jar” fork=”true”/>
</target>
</project>
```

4. Right click on your project and select Properties.
5. Select Builders→New...→Ant Builder→Ok
6. An Edit Configuration window should appear. In the Buildfile section, select Browse Workspace...→desired project→build.xml. In the Base Directory section, select Browse Workspace...→desired project Then Ok
7. Build the project. This can be done in the Eclipse IDE or on the command line with the command "ant" or "ant makejar" (must be in the same directory as the build.xml file). In order to perform the buid from the command line, the build machine must have Apache ant installed.
8. If done successfully, the output will display BUILD SUCCESSFUL.
9. To verify the jar is signed, open a terminal and run the following command:
   jarsigner -verify -verbose -certs
   a. If jar verification is successful, output will display jar verified
   b. If jar verification is unsuccessful, output will display jar is unsigned

3.3. Android Studio

The BlackVault HSM's integration with Android Studio involves a straightforward modification of the build.gradle file. In the following section, highlighted portions of text are unique to the build machine and project.

3.3.1. Setting up Gradle to Automatically Sign Code Build

1. Ensure a key exists on the BlackVault that Java can use. Check with:
   keytool -keystore NONE -storetype PKCS11 -storepass 2222 -list
   Log in to the BlackVault as user and check pkcs.dat file is setup correctly
2. In Android Studio, Select the 1:Project tab on the left.
3. Select Project in the Android drop down menu.
4. Select the desired project and then double click build.gradle
5. At the end of the build.gradle file, add the following:
   a. Linux:
   ```
   task sign(type: Exec) {
   executable ”jarsigner”
   args”-keystore”,”NONE”,”-storetype”,”PKCS11”,”-storepass”,”2222”,\
   ”/home/user/AndroidStudioProjects/MyApp/app/build/outputs/apk/release/ap
   p-release-unsigned.apk”,\
   ”2048key”
   }
   ```
   b. Windows:

```
task sign(type: Exec) {
executable ”jarsigner”
args”-keystore”,”NONE”,”-storetype”,”PKCS11”,”-storepass”,”2222”,\
”C:\\User\\user\\AndroidStudioProjects\\MyApp\\app\\build\\outputs\\apk\\re
lease\\app-release-unsigned.apk”,\
”2048key”
}
```

6. From the tool bar, select app then Edit Configurations... in the drop-down menu.
7. Select + (top left) then Gradle.
8. In the Gradle Project: section, enter:
    a. Linux:
       /home/user/AndroidStudioProjects/MyApp/build.gradle
    b. Windows:
       C:\Users\user\AndroidStudioProjects\MyApp\build.gradle
    c. In the Tasks: section, enter:

       assembleRelease sign

3.3.2. Manually Signing a Code Build

To manually sign each code build rather than automatically sign with a build.gradle script, perform the following:

1. Linux:
```
jarsigner -keystore NONE -storetype PKCS11 -storepass 2222 \
/home/user/AndroidStudioProjects/MyApp/app/build/outputs/apk/release/app-
release-unsigned.apk \
keyNameHere
```
2. Windows:
```
jarsigner -keystore NONE -storetype PKCS11 -storepass 2222 \
C:\\User\\user\\AndroidStudioProjects\\MyApp\\app\\build\\outputs\\apk\\release\\a
pp-release-unsigned.apk \
keyNameHere
```