Black•Vault HSM

Java Jarsigner

Integration Guide

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

1. Disclaimer and Warranty

Engage Black is a business unit of Engage Communication.

Engage Communication makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability of fitness for any particular purpose. Information in this document is subject to change without notice and does not represent a commitment on the part of Engage Communication, Inc. Product specifications are subject to change without notice. Engage Communication assumes no responsibility for any inaccuracies in this document or for any obligation to update the information in this document.

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

Phone +1 (831) 688-1021

http://www.engageblack.com/

http://www.engageinc.com/

2. Table of Contents

3. Introduction

The BlackVault Hardware Security Module (HSM) integrates with Java Jarsigner to enable you to identify the publisher of a software component before running it on your computer and to verify that no one has altered the code after it has been signed. Java Jarsigner relies on proven cryptographic techniques and the use of one or more private keys to sign and time-stamp the published software. It is important to maintain the confidentiality of these keys.

The benefits of using an HSM with Java Jarsigner include:

- Protection for the organizational credentials of the software publisher.
- Secure storage of the private key.
    - Signing code within a cryptographically secure environment
- FIPS 140-2 level 3 validated hardware.

3.1. Supported Operating Systems And Java Versions

Supported operating systems

OS Name	Version	32 bit	64 bit
Windows	7	X	X
	Server 2008 R2 x64		X
	8.1	X	X
	Server 2012 x64		X
	Server 2012 R2 x64		X
	10	X	X
	Server 2016 x64		X
Ubuntu	12.04	X	X
	14.04	X	X
	16.04	X	X
Centos	6	X	X
	7 x64		X
Red Hat	6 x64		X
	7 x64		X
Open Suse	13.2	X	X
	42.1 x64		X
Debian	7.9	X	X
	8.4		X

The Java version used must support the Sun PKCS#11 provider. This includes Java 8 and later versions as well as the 32 bit version of Java 7.

4. Procedure

To proceed the following is needed:

- BlackVault HSM
- BlackVault Card Set
- BlackVault HSM Setup CD
- A client computer that has a supported Operating System installed.

Additionally, the BlackVault must be Initialized and Configured properly (see section 6.3 and 6.4 of the BlackVault HSM User Guide for more details)

To setup Java with the BlackVault HSM:

- Install the appropriate version of Java (if not already installed)
- Install the BlackVault HSM Libraries onto the Client Machine (included on the Setup CD)
- Run the Setup Wizard
- Validate operation (i.e. create test key …. Etc.)

The following assumes you already initialized the BlackVault HSM and are installing this software on a machine that does not already have java working with the BlackVault.

4.1. Install Java

To Install Java please consult the proper Java Documentation typically found here:

https://www.java.com/en/download/help/download_options.xml

4.2. Install BlackVault Library and integrate with Java

The BlackVault communicates over the network using PKCS#11. For a host to communicate with the BlackVault, the BlackVault's PKCS#11 library, TLS certificates, and pkcs.dat must be installed first. Additionally, the Java files nss.cfg and java.security must be correctly setup.

If installing the HSM client application on a Windows client, proceed to Section 4.2.1; if on a Linux Client, proceed to Section 4.2.2.

4.2.1. Windows

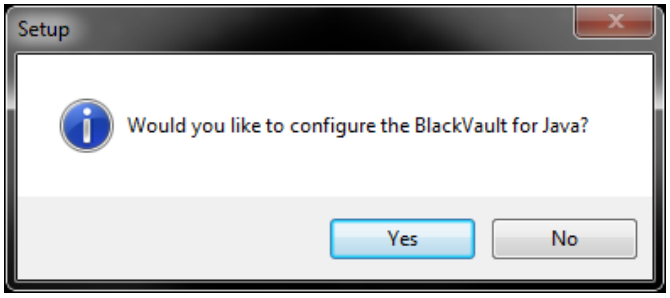
1. Insert the BlackVault HSM Setup CD into your computer, on it you will find a file called bv-setup.exe

2. Run bv-setup.exe

3. Windows asks if you want the installer to make changes to computer, select yes.

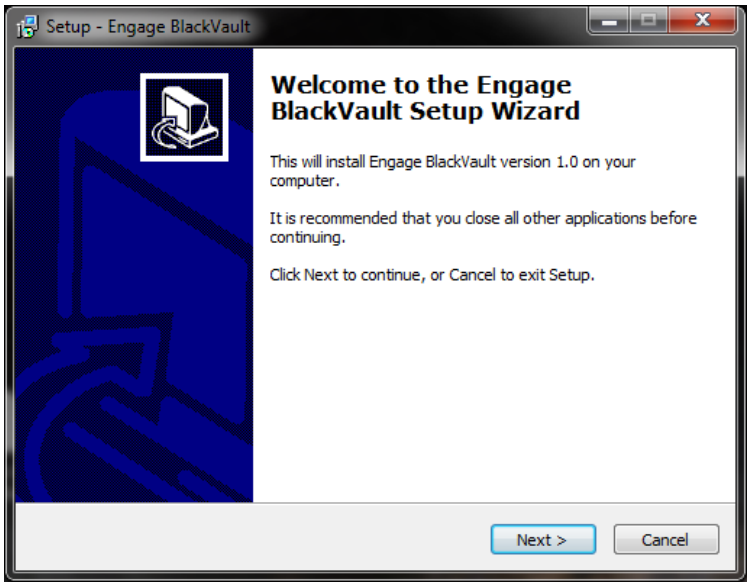
4. The installer asks, “Would you like to configure the BlackVault for Java?” select yes or no.

a. if no is selected, skip step 6

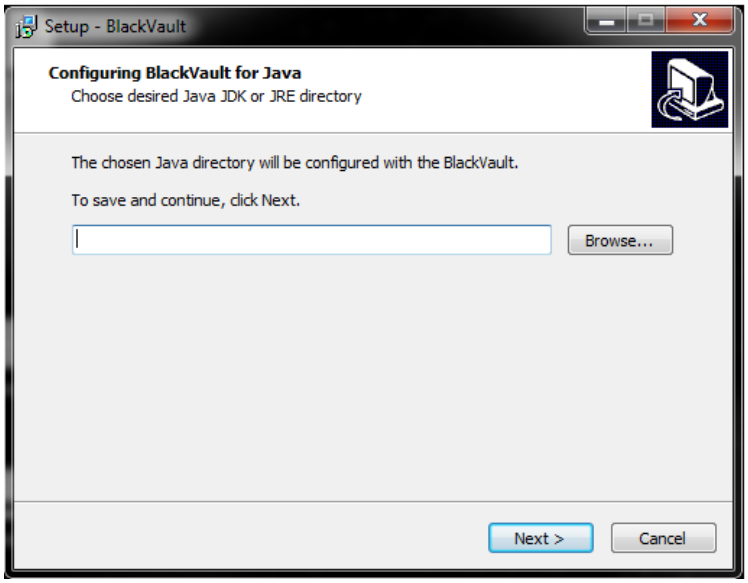


b. If yes is selected, perform step 6

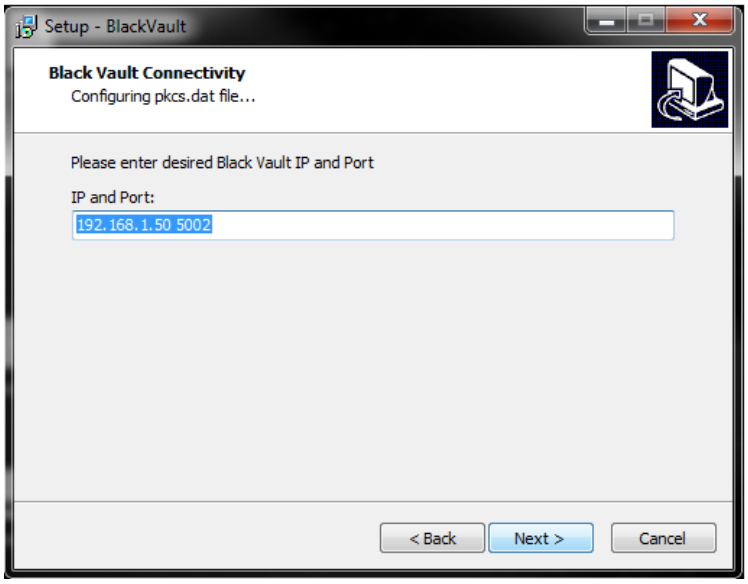
5. The first window you will see is the overview window. It goes over what will be installed. Select Next to continue.



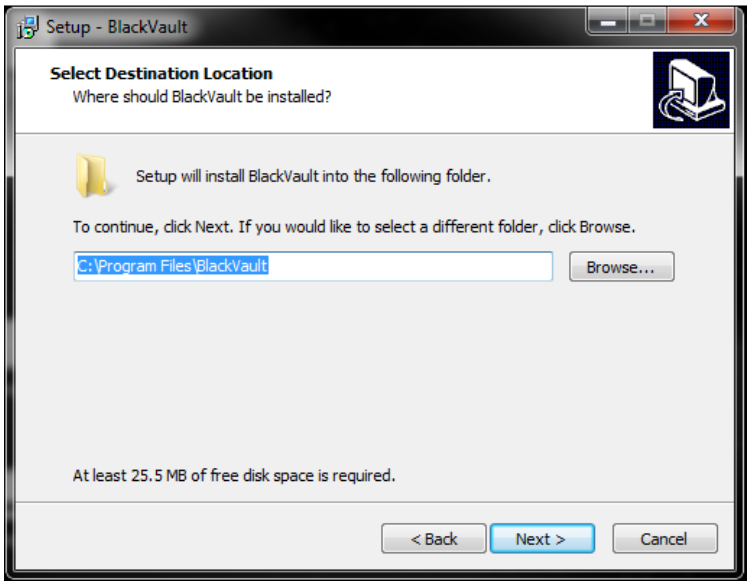
6. Next the installer asks for the Java directory currently being used on Windows. Browse for the desired top-level Java directory (usually in C:/Program Files/Java/jre8 or C:/Program Files(x86)/Java/jre8) then press next



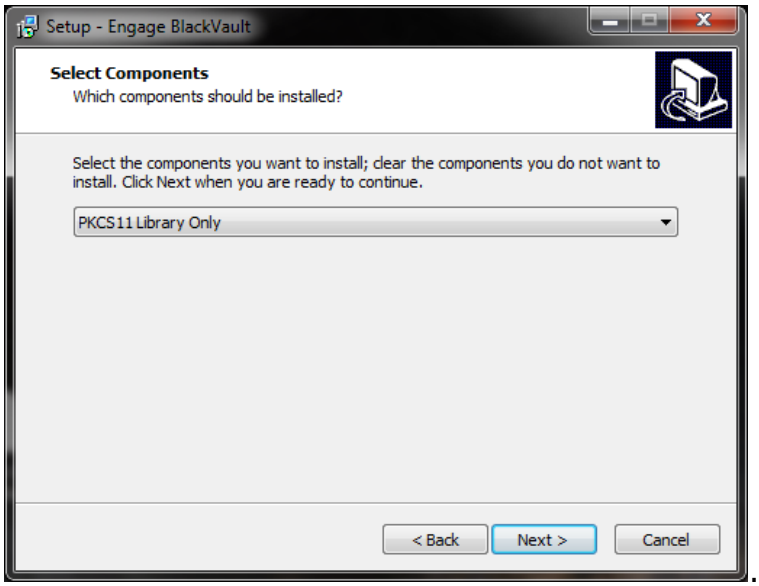
7. Next the installer asks for the BlackVault IP address and TLS Port. Enter those there and press next.



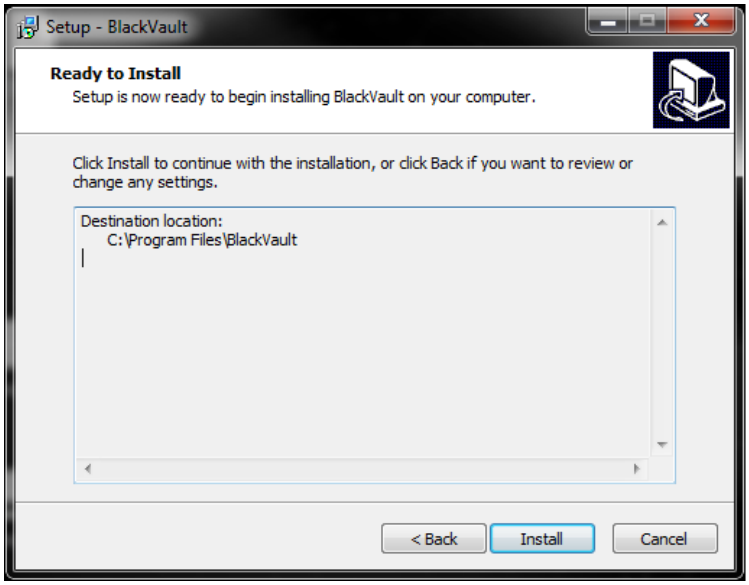
8. Next, the installer asks for the location of the directory to install the necessary files to. Either use the default location, or to select a new location, click browse and specify the new location.



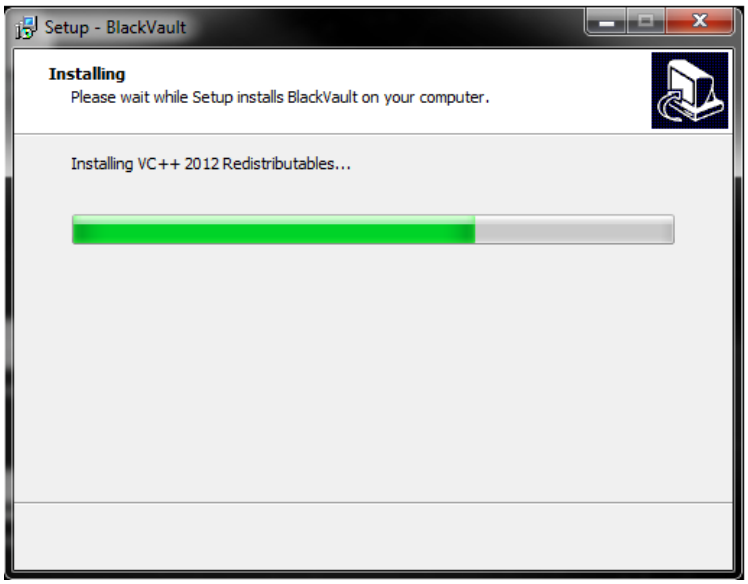
9. Next, the installer asks for which components to install, the Engage BlackVault Cryptography Provider (CNG/Authenticode and PKCS#11 libraries), or just the PKCS#11 Library

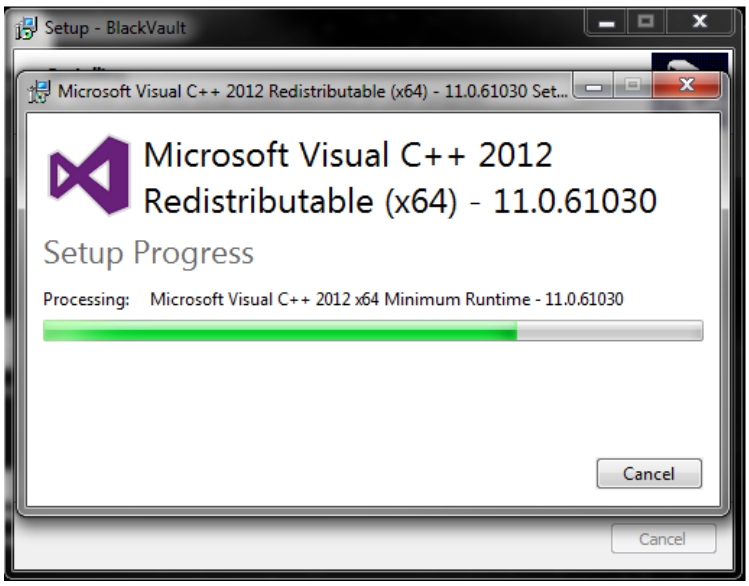


10. The installer then gives a summary of items to be installed press install to continue.

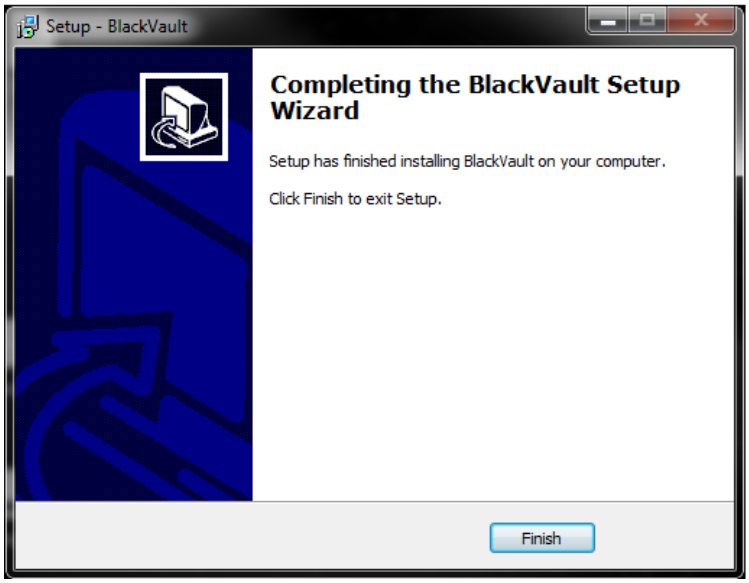


11. The installer then installs all the components, including a Microsoft Visual C++ Redistributable.





12. Once the installer finishes press finish.



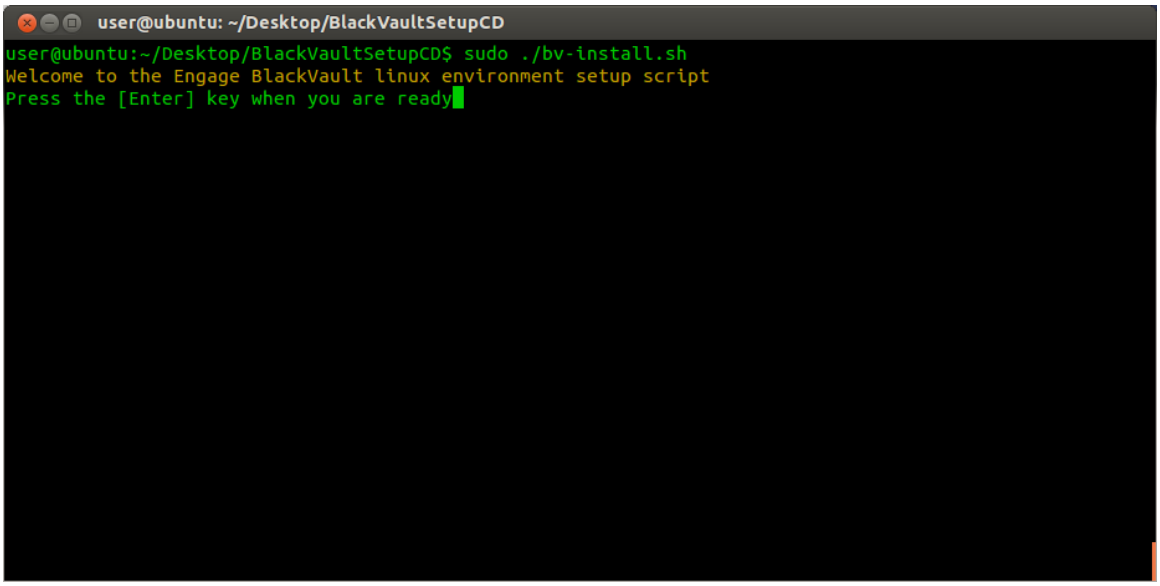
4.2.2. Linux

1. Copy the BlackVault Setup CD to a directory on your system. In that directory do the following:

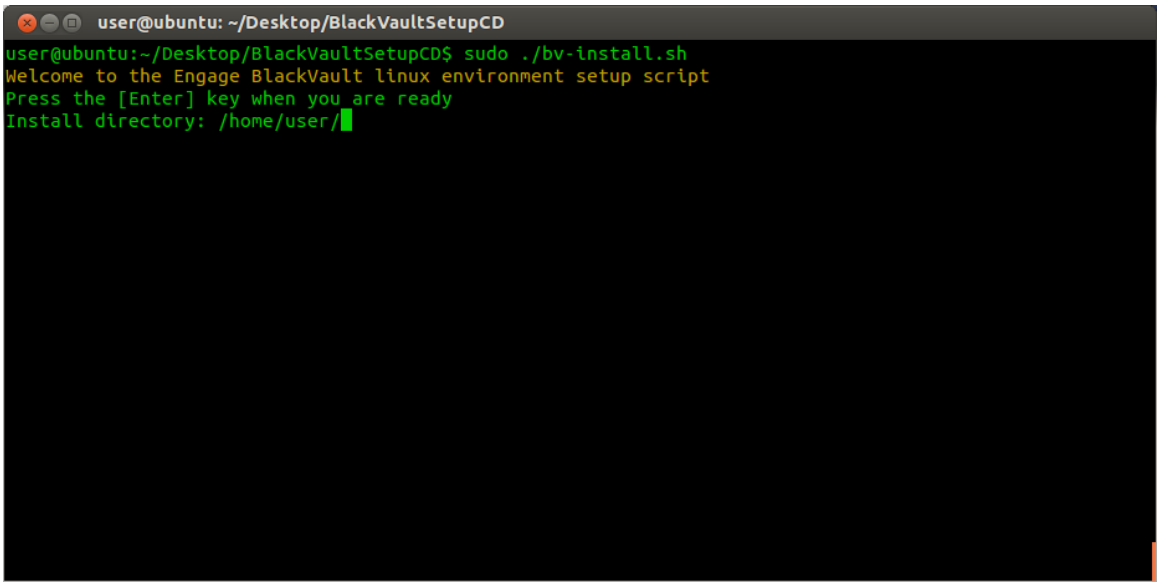
2. chmod +x bv-install.sh

3. sudo ./bv-install.sh

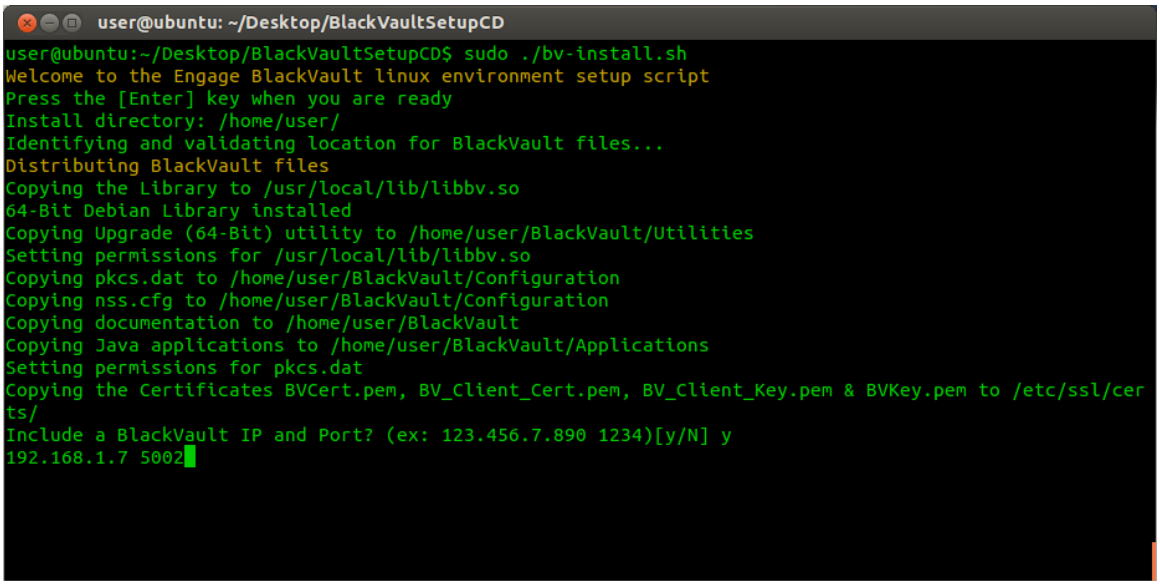
4. press enter



5. The appropriate PKCS11 library libbv is installed in /usr/local/lib. Then the script asks for an install directory (default to your home folder). Enter the install directory (or leave blank for default and press enter.

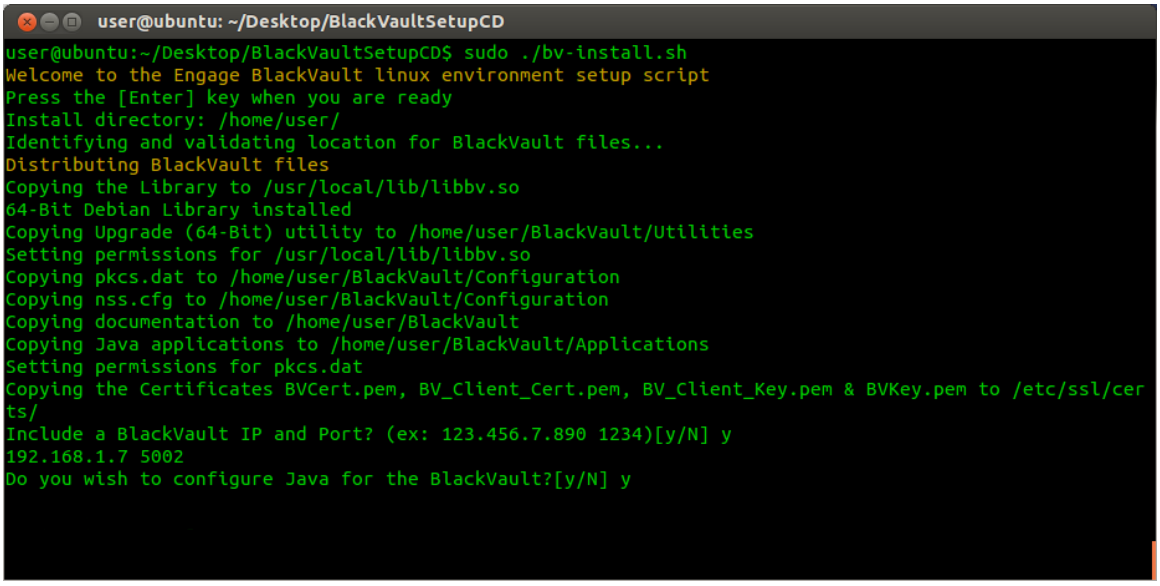


6. The script then asks for if you would like to "Include a BlackVault IP and Port? (ex: 123.456.789 1234) {y/N]" for the pkcs.dat file. To continue, press "y", then the enter key. Next type the ip address and port number of the BlackVault and press the enter key again.

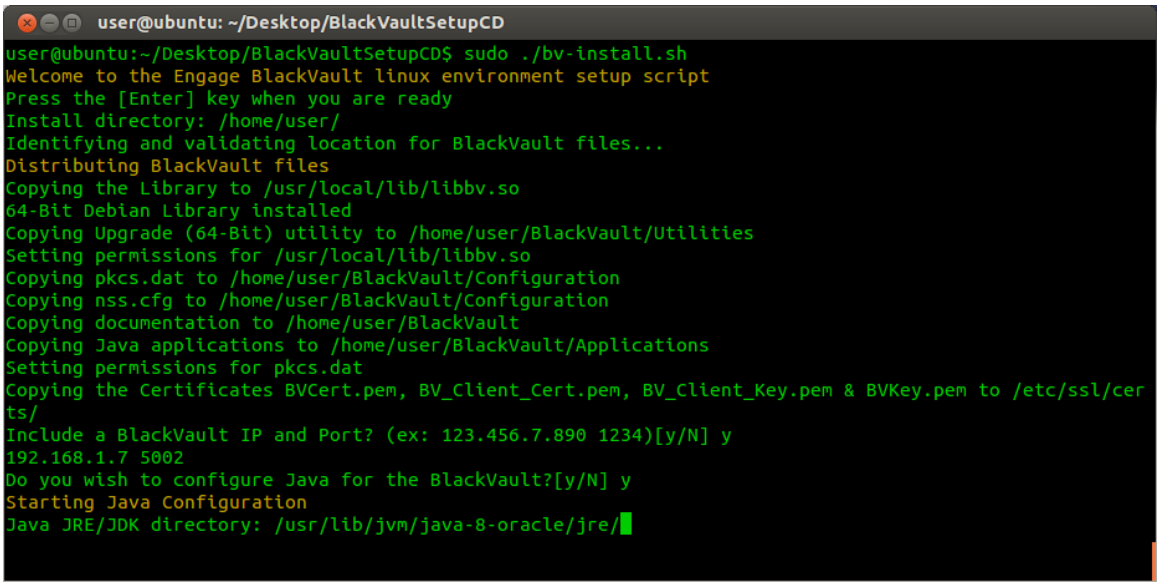


7. The TLS certificates are installed in /etc/ssl/certs.

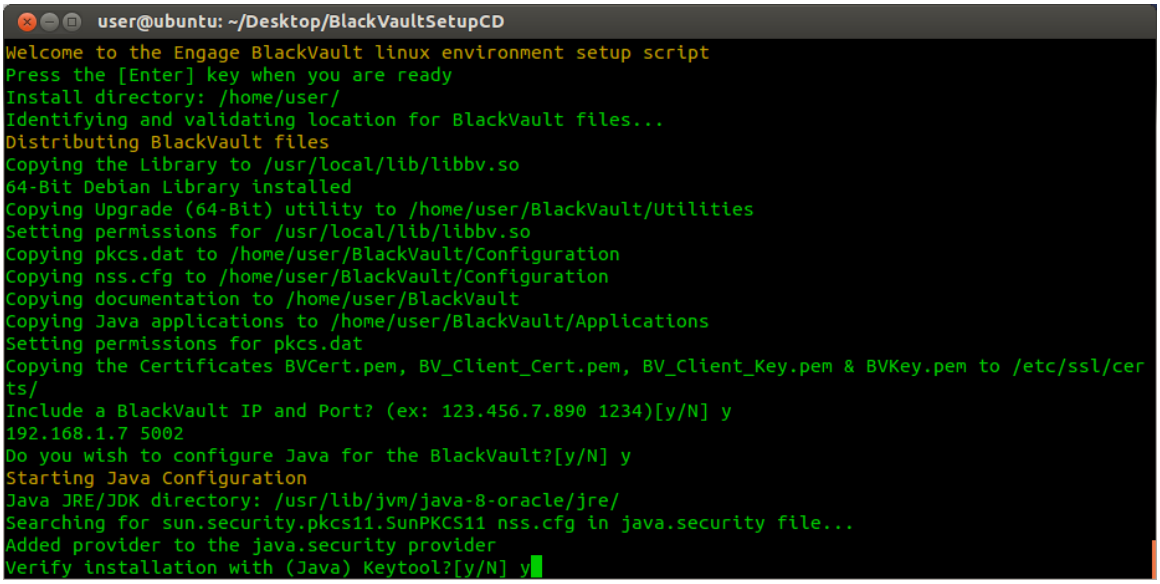
8. At this point the BlackVault is setup to work with any non-Java application written to the PKCS11 API such as applications written in C or C++. Select y and press enter to continue setting up java.



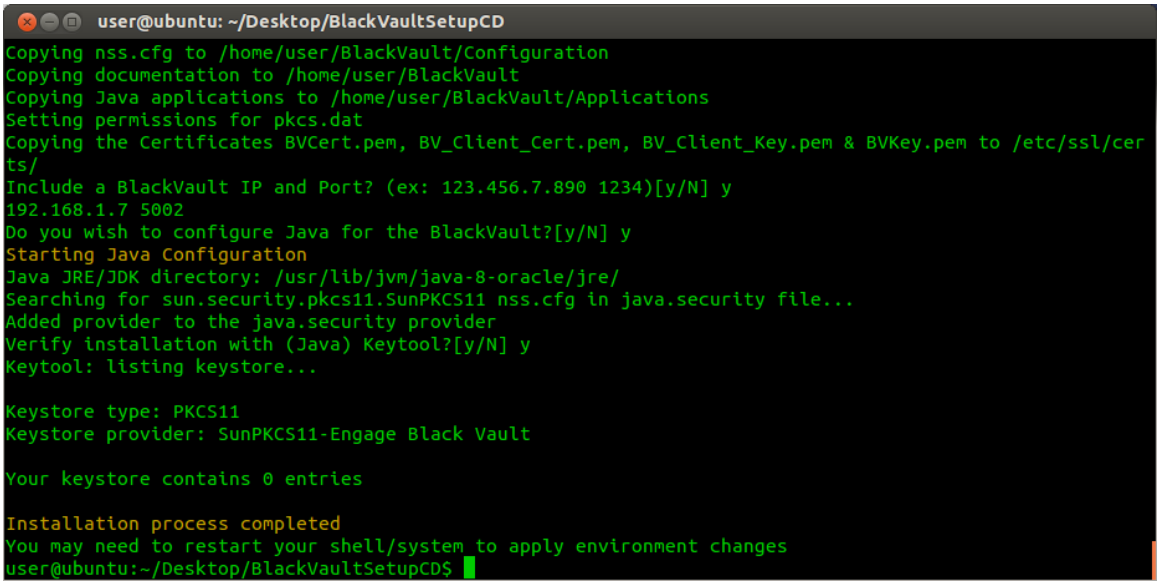
9. The Script asks for the "Java JRE/JDK directory" then displays the environment variable $JAVA_HOME. Press enter if this is the correct directory. If it is not change it now then press enter.



10. The script now asks, “Verify installation? [y/N]”. Choose “y” if you wish to have the script run Java keytool to list the keys stored on the BlackVault.



11. Log in as User on the BlackVault. Provided no keys have been created, the expected result would look something like the following:



12. In order for full settings to take restart the system, as environment variables were changed.

4.3. Signing with Java Jarsigner and the BlackVault HSM

To do code signing per industry best practices, along with creating and storing the key inside a secure HSM, a code signing certificate associated with the key is required. This section first describes how to create a key, and then how to create the certificate.

If a key already has been created by Java, you can skip the first section and move onto Jarsinger operation

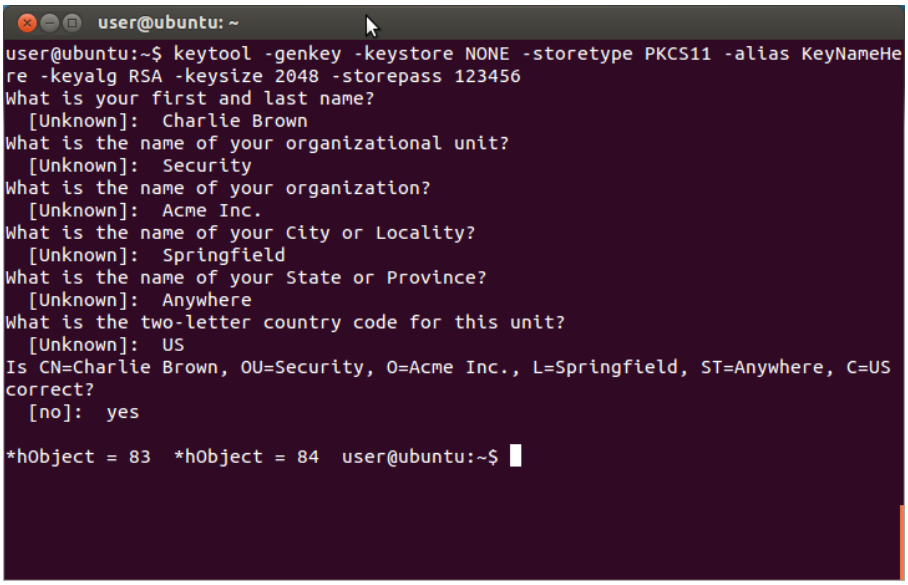
4.3.1. Key and Certificate Creation

Before signing code, you need a key and certificate for that key. Keytool is used for this purpose as described below.

Keytool is a key and certificate management utility that allows users to administer their own public/private key pairs and associated certificates. It is used in self-authentication or data integrity and authentication services, using digital signatures. It also allows users to cache public keys (in the form of certificates) of their communicating peers.

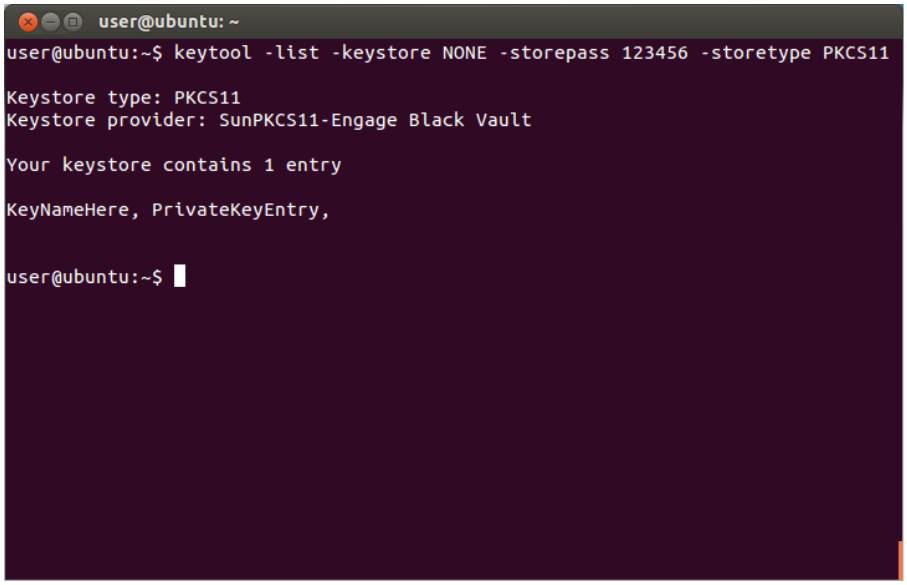
If you have completed the previous steps in this guide (Java configured correctly, and PKCS#11 library installed), perform the following using a command terminal:

1. To create a key:
    a. keytool -genkey -keystore NONE -storetype PKCS11 -alias KeyNameHere -keyalg RSA -keysize 2048 -storepass 123456
        i. keystore and storetype is telling Java to use the PKCS#11 functions of Java
        ii. alias is the name of the key
        iii. keyalg is what key algorithm you want to use
        iv. keysize is the size of the algorithm
        v. Although Java requires a storepass entry (typically used for authenticating the keystore), the BlackVault HSM authenticates itself and does not use this password.



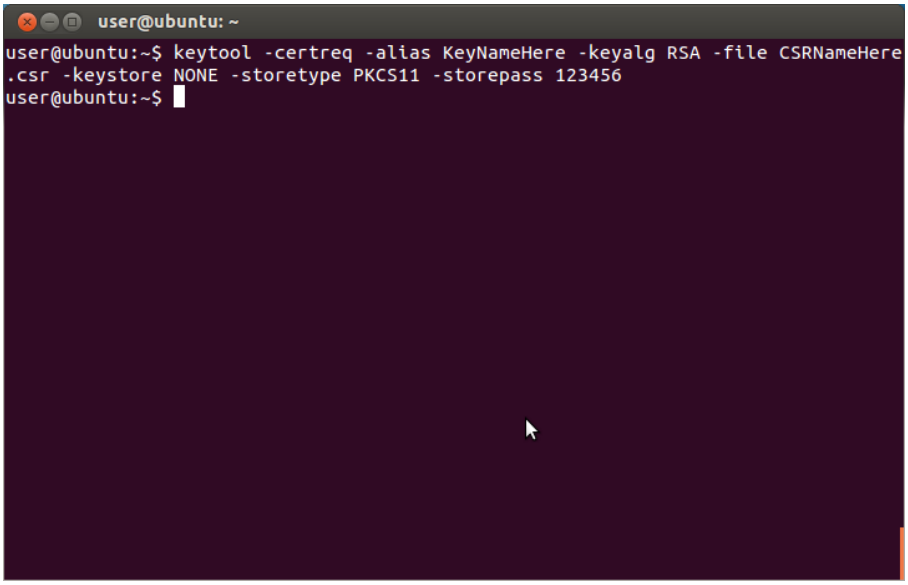
b. After entering the keytool command Java will prompt for Identifying information. This is used for the self-signed certificate that is created by the BlackVault HSM at the same time the key is created.
c. It will then prompt if all entries are correct. When you are sure the information is correct, type yes and then press the enter key.

2. After creating the key, verify the key is stored in the BlackVault by listing the keys from the BlackVault with the following command.
    a. For java keytool has a list key function
        i. keytool -list -keystore NONE -storepass 123456 -storetype PKCS11



If a self-signed certificate is sufficient skip to the next section, if you would like a certificate signed by an independent certificate Authority perform the following:

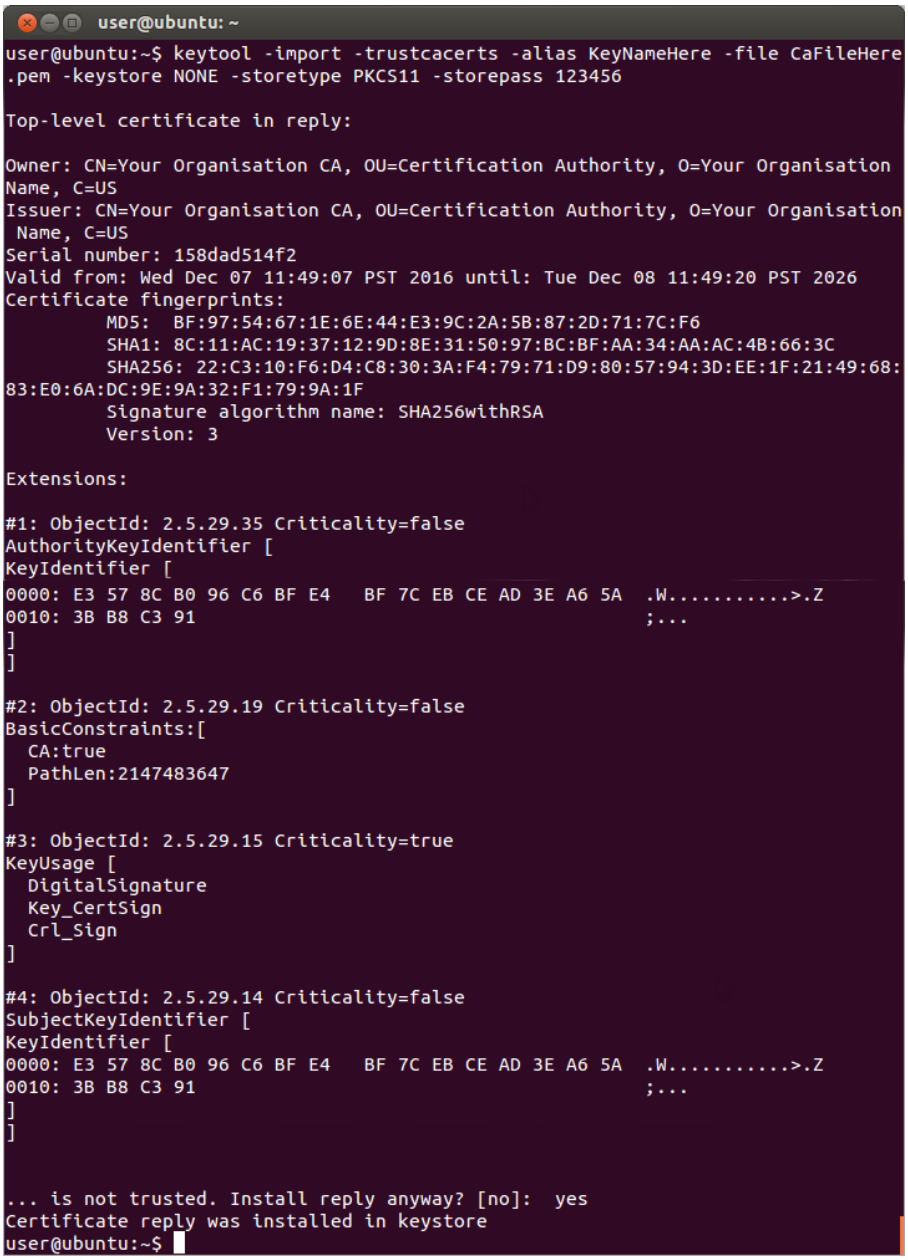
1. Next a certificate signing request (CSR) is generated from the key just created by performing the following steps.
    a. For Java enter the following command:
    keytool -certreq -alias KeyNameHere -keyalg RSA -file CSRNameHere.csr -keystore NONE -storetype PKCS11
        i. keystore and storetype is telling Java to use the PKCS#11 functions of Java
        ii. alias is the name of the key
        iii. keyalg is what key algorithm you want to use
        iv. keysize is the size of the algorithm
        v. storepass is normally used for authenticating the keystore, but the BlackVault HSM authenticates itself and does not use this password. Java requires it anyways
        vi. file is the filename that the CSR will output to.



2. Get the CSR signed by a Certificate Authority (i.e. Digicert, Verisign, etc)

3. 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
    a. For Java enter the following command:
    keytool -import -trustcacerts -alias KeyNameHere -file CaFileHere.pem -keystore NONE -storetype PKCS11

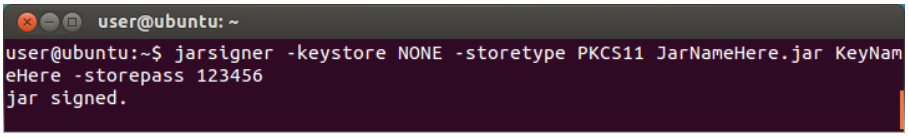
i. keystore and storetype is telling Java to use the PKCS#11 functions of Java
ii. alias is the name of the key
iii. file is the location and file name of the certificate that was created by a certificate authority



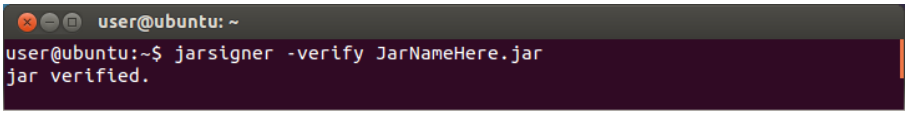
4.3.2. Jarsigner Operation

Everything is setup now to start code signing. To do a sample code signing to verify everything is working do the following:

1. sign code
    a. In Java, use the following command
    jarsigner -keystore NONE -storetype PKCS11 /Path/To/Your/Jar KeyNameHere
        i. keystore and storetype is telling Java to use the PKCS#11 functions of Java
        ii. KeyNameHere is the name of the key on the BlackVault
        iii. /Path/To/Your/Jar is the location and file name of the jar that is needing to be signed.



2. Verify code
    a. In Java, use the following command
    jarsigner -verify /Path/To/Your/Jar
        i. /Path/To/Your/Jar is the path to the jar that needs to be verified
    b. Provided everything was set up correctly, the output will show the information of your certificate as well as the information of the CA that signed the certificate.



    c. If the output is something different than the Verify failed and the code that you try to verify was changed from what was originally signed.