Black●Vault

Android Dev Studio

Integration Guide

© Engage Black
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.

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 95003Phone +1(831) 688-1021
http://www.engageblack.com/http://www.engageinc.com/

sales@engageinc.com+1 (831) 688-1021

Global Technical Support+1 (831) 688-1021 **(extension 3)** support@engageinc.com

Product Warranty Seller warrants to the Original Buyer that any unit shipped to the Original Buyer, under normal and proper use, be free from defects in material and workmanship for a period of 12 months from the date of shipment to the Original Buyer. This warranty will not be extended to items repaired by anyone other than the Seller or its authorized agent. The foregoing warranty is exclusive and in lieu of all other warranties of merchantability, fitness for purpose, or any other type, whether express or implied.

Remedies and Limitation of Liability A. All claims for breach of the foregoing warranty shall be deemed waived unless notice of such claim is received by Seller during the applicable warranty period and unless the items to be defective are returned to Seller within thirty (30) days after such claim. Failure of Seller to receive written notice of any such claim within the applicable time period shall be deemed an absolute and unconditional waiver by buyer of such claim irrespective of whether the facts giving rise to such a claim shall have been discovered or whether processing, further manufacturing, other use or resale of such items shall have then taken place.

B. Buyer’s exclusive remedy, and Seller’s total liability, for any and all losses and damages arising out of any cause whatsoever (whether such cause be based in contract, negligence, strict liability, other tort or otherwise) shall in no event exceed the repair price of the work to which such cause arises. In no event shall Seller be liable for incidental, consequential, or punitive damages resulting from any such cause. Seller may, at its sole option, either repair or replace defective goods or work, and shall have no further obligations to Buyer. Return of the defective items to Seller shall be at Buyer’s risk and expense.

C. Seller shall not be liable for failure to perform its obligations under the contract if such failure results directly or indirectly from, or is contributed to by any act of God or of Buyer; riot; fire; explosion; accident; flood; sabotage; epidemics; delays in transportation; lack of or inability to obtain raw materials, components, labor, fuel or supplies; governmental laws, regulations or orders; other circumstances beyond Seller’s reasonable control, whether similar or dissimilar to the foregoing; or labor trouble, strike, lockout or injunction (whether or not such labor event is within the reasonable control of Seller).

European Community Compliance Statement This product conforms with the protection requirements of EU Council Directive 2004/108/EC relating to electromagnetic compatibility.

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 a particular 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

2. Table of Contents

3. Introduction

The BlackVault Hardware Security Module (HSM) integrates with Microsoft Authenticode to enable you to identify the publisher of a software component before it is downloaded from the Internet, and to verify that no one has altered the code after it has been signed. Microsoft Authenticode 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 Microsoft Authenticode include:

- Protection for the organizational credentials of the software publisher.
- Secure storage of the private key.
- FIPS 140-2 level 3 validated hardware.

3.1. Supported Operating Systems

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

4. Procedure

To setup Authenticode with the BlackVault HSM:

- Initialize the HSM
- Install the BlackVault HSM Libraries
- Configure Android Dev Studio to work with HSM

You can find information about how to initialize the BlackVault HSM in the BlackVault User Guide

The following assumes that you are starting with a new project.

4.1. Integrate Android Studio with the BlackVault HSM

Integration of Android Dev studio and the BlackVault HSM requires that Android Dev Studio has been set up; that the BlackVault libraries have been installed; and that java has been installed and configured to use the BlackVault.

For Java integration please consult the java integration guide.

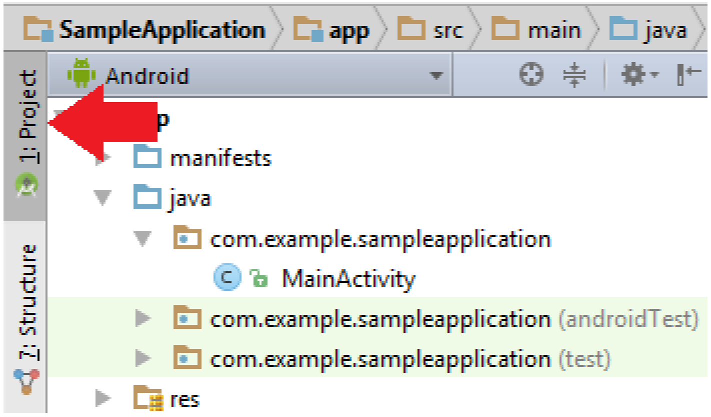
1. Please pick a key that you would like to use; if you have not already generated a key perform the following:

    a. Open a command prompt window on the Host.
    b. Use the following command:

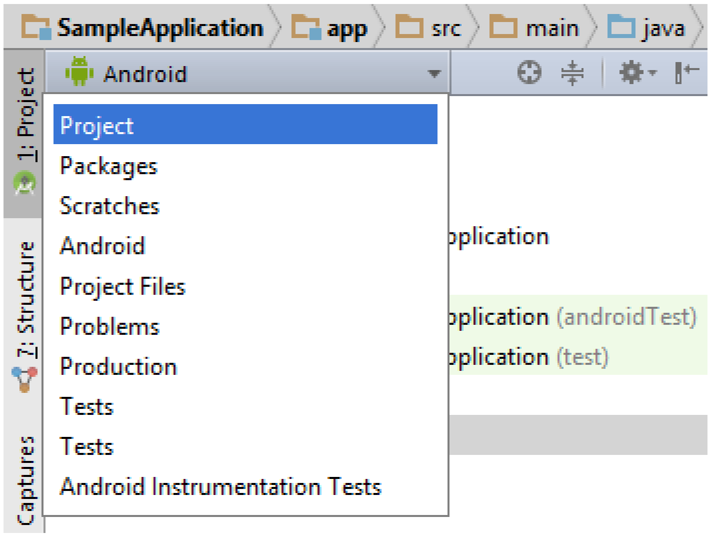
    ```
    keytool -genkeypair -keystore NONE -storepass 2222 -storetype PKCS11 -alias “SampleKey” -keyalg “RSA” -keysize “2048” -dsname “CN=Bob Joe, OU=Development, O=Engage Communication Inc., L=Aptos, S=California, C=US”
    ```

2. Create a project in Android Studio by selecting **File > New > Project**. Once the project has been created, ensure that a copy of updated pkcs.dat file is in the project folder.

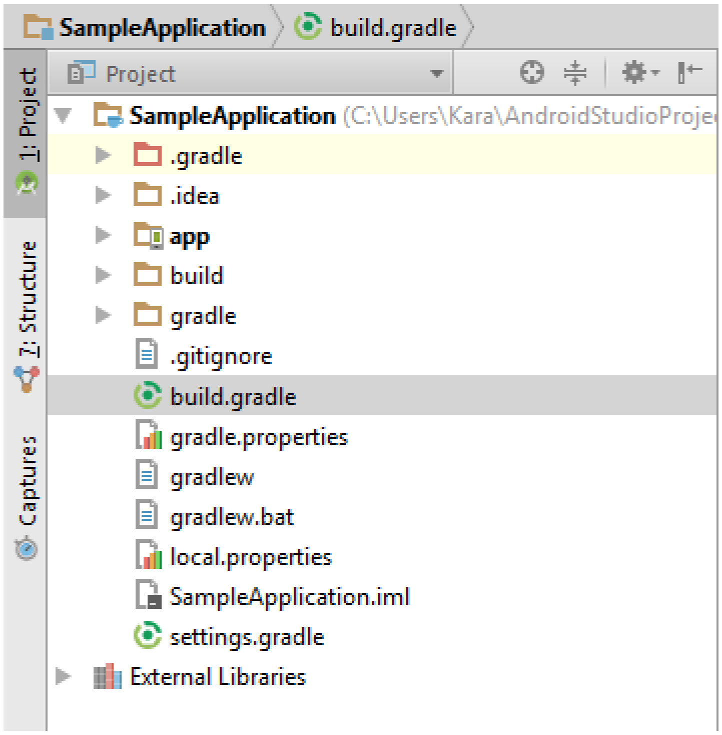
3. Click the Project tab if not already selected.


SampleApplication
app
src
main
java
Android
1: Project
7: Structure
p
manifests
java
com.example.sampleapplication
MainActivity
com.example.sampleapplication (androidTest)
com.example.sampleapplication (test)
res

4. Select the drop-down menu and click Project.

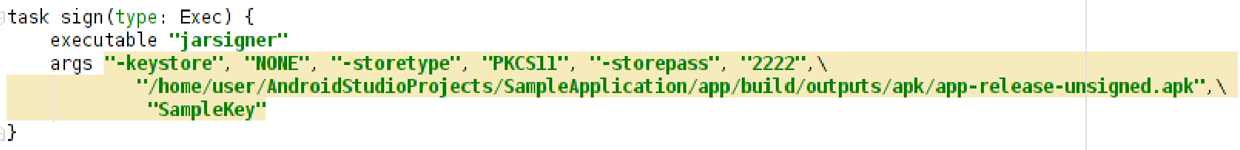


5. Select the project, and then double-click build.gradle.



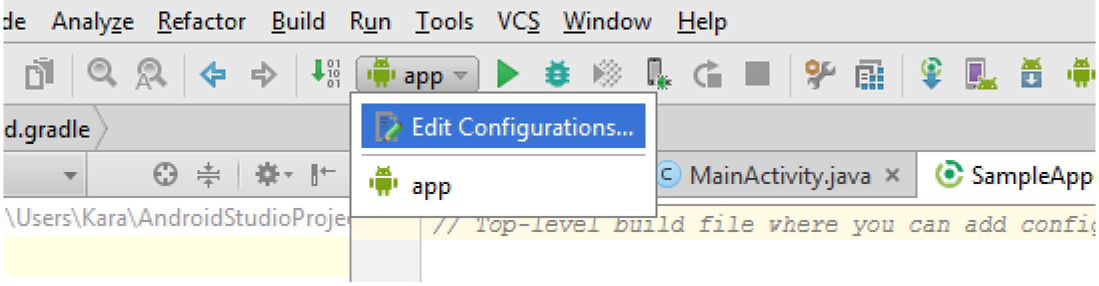
6. Create a signing task in the project folder using the following lines:

   task sign(type: Exec) {
   executable “jarsigner”
   args “=keystore”, “NONE”, “-storetype”, “PKCS11”, “-storepass”, “2222”,\
   “home/user/ANdroidStudioProjects/SampleApplication/app/build/outputs/apk/app-release-unsigned.apk”,\
   “SampleKey”
   }



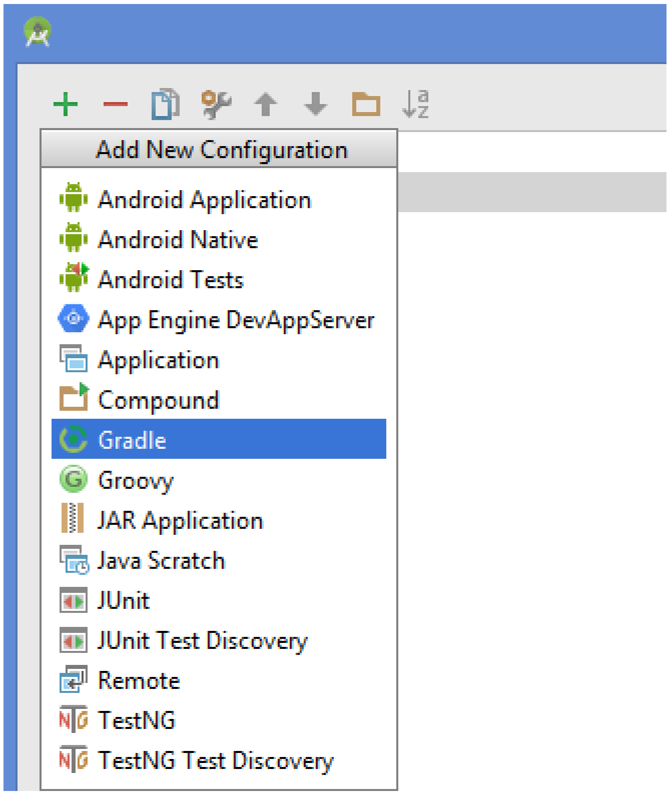
```
task sign(type: Exec) {
    executable "jarsigner"
    args "-keystore", "NONE", "-storetype", "PKCS11", "-storepass", "2222",\
          "/home/user/AndroidStudioProjects/SampleApplication/app/build/outputs/apk/app-release-unsigned.apk",\
           "SampleKey"
}
```

7. From the Tool Bar, click “app”, and then select Edit Configurations.

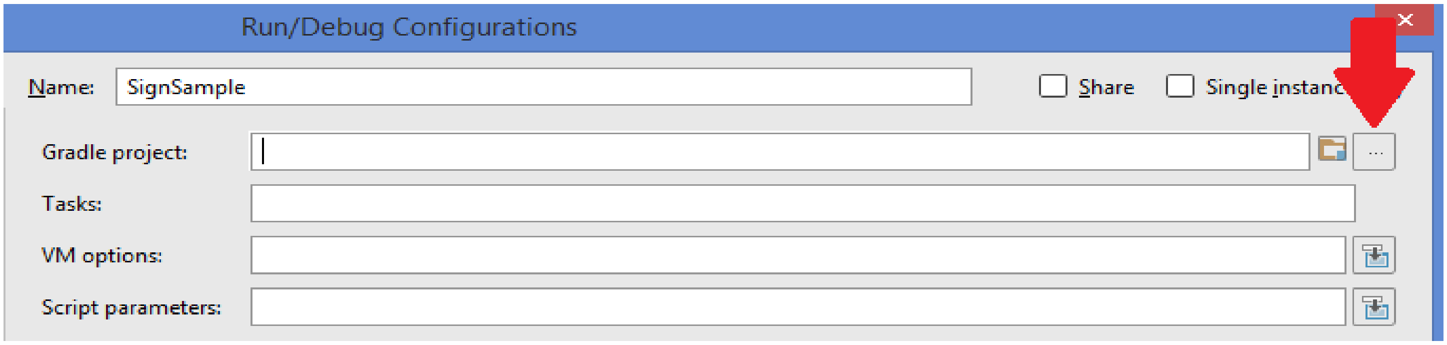

de Analyze Refactor Build Run Tools VCS Window Help
app
Edit Configurations...
app
d.gradle
MainActivity.java
SampleApp
\Users\Kara\AndroidStudioProje

```
// Top-level build file where you can add confi
```

8. Select the plus sign and click Gradle.



9. Name the new gradle, and select the “…” by Gradle Project.



Name: SignSample

☐ Share ☐ Single instanc

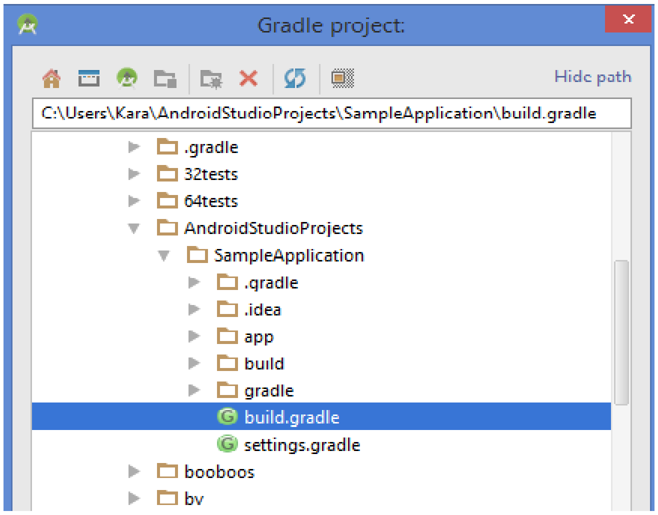
Gradle project:

Tasks:

VM options:

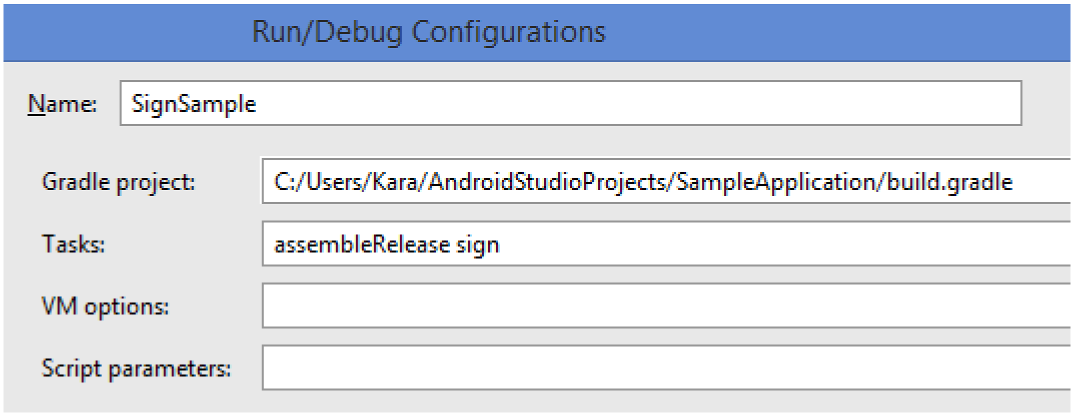
Script parameters:

10. Double-click build.gradle.



C:\Users\Kara\AndroidStudioProjects\SampleApplication\build.gradle

11. Type "assembleRelease sign" in Tasks. When completed, the gradle configurations should look like this:



Name: SignSample

Gradle project: C:/Users/Kara/AndroidStudioProjects/SampleApplication/build.gradle

Tasks: assembleRelease sign

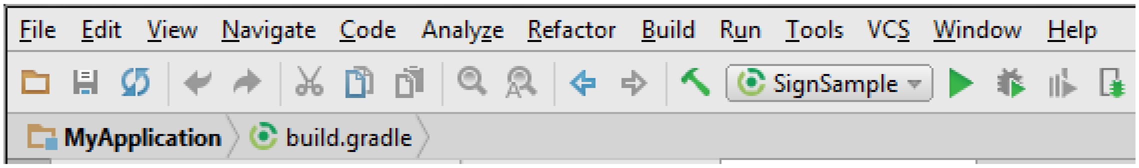
VM options:

Script parameters:

12. Click **Apply** and **Okay**.

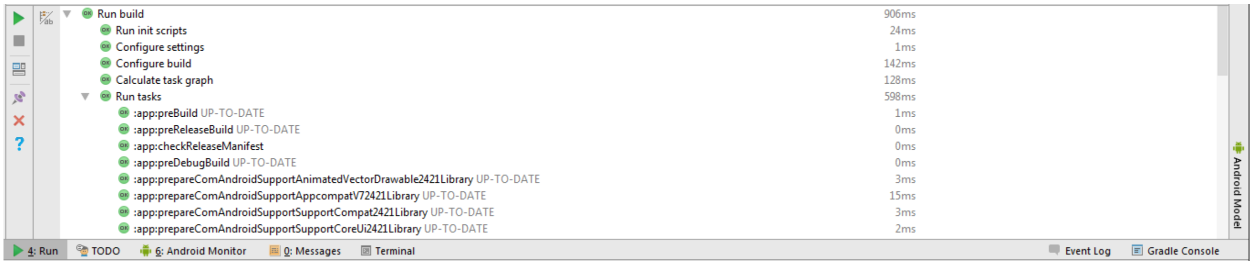
4.2. Signing with Android Studio

1. In the menu bar, adjacent to the Run/Debug configuration you just created press the green play button



This will sign and build your code.

2. Check the output to verify everything built correctly. If everything did you



```
Run build                                                                    906ms
  Run init scripts                                                            24ms
  Configure settings                                                           1ms
  Configure build                                                            142ms
  Calculate task graph                                                       128ms
  Run tasks                                                                  598ms
    :app:preBuild UP-TO-DATE                                                   1ms
    :app:preReleaseBuild UP-TO-DATE                                            0ms
    :app:checkReleaseManifest                                                  0ms
    :app:preDebugBuild UP-TO-DATE                                              0ms
    :app:prepareComAndroidSupportAnimatedVectorDrawable2421Library UP-TO-DATE  3ms
    :app:prepareComAndroidSupportAppcompatV72421Library UP-TO-DATE            15ms
    :app:prepareComAndroidSupportSupportCompat2421Library UP-TO-DATE           3ms
    :app:prepareComAndroidSupportSupportCoreUi2421Library UP-TO-DATE           2ms
```

3. Verify that the signing happened correctly. From a command prompt window, enter:
jarsigner -verify -verbose -certs <jarfilename.jar or androidfilename.apk>

    a. if it is successful it will give the certificates used in signing the jar. It will also at the end state:
    "jar verified"

    b. If it is unsuccessful the command will output:
    "jar is unsigned. (signatures missing or not parsable)"