

## Setting up the Offline Stamp SDK for Android

The SnowShoe Offline Stamp SDK enables the user to stamp the screen of an Android device and get a response including: stamp serial number, timestamp, any error data, and a GUID receipt. This functionality is enabled by synchronizing stamp data associated with your SnowShoe account to your mobile app.

Here are the steps to set it up:

1. Copy `stampsdk.aar` to `libs` folder in your module.
2. In the `'fileTree'` line, of the dependencies section, of the `build.gradle` of the module, you need to add `" '*.aar' "` to the `'include'` section: `"[ '*.jar' ]"`. It will then look like so: `"[ '*.jar', '*.aar' ]"`
3. The easiest way to turn a screen into a stampable screen is to add the `"com.snowshoe.stampsdk.SnowShoeView"` view as the last element of the layout you want stampable. You will want to give it an `"id"`, set the `"layout_width"` and `"layout_height"` as `"match_parent"`, and to be the least intrusive also set the `"background"` to `"@android:color/transparent"`.
4. Go to the Activity connected to the layout you added `"SnowShoeView"` to and go to the `"onCreate()"` method and get that view object with `findViewById()`.
  - o With the view object set the `appKey` with the function `"setAppKey(String key)"` and `appSecret` with the function `"setAppSecret(String secret)"` using the `"Application Key"` and `"App Secret"` (found by logging in to <https://snowshoestamp.com>) so that the app will have permission to talk with the cloud.
  - o You will then need to set, on the view object, the `"snowShoeListener"` with `"setSnowShoeListener(SnowShoeListener listener)"`. This interface has two methods:
    - `onGetSerialResult(SnowshoeResult result)` - Runs after a stamp serial number has been found or not found.
    - `onSyncCompleted(boolean result)` - Runs after the local stamp table has been updated from the cloud.
5. You will also want to run the `syncStampInfo()` method before presenting the stamp screen to ensure there is stamp serial number data on the device.
6. For this to function properly there will also need to be a few implementations that you may or may not already have added this same way before or through `.jar` files. If you have already added them you don't need to add them again as it will bring up errors about duplicates.
  - o The first thing you want to do before you start adding these is make sure you are using a version of `"com.android.tools.build:gradle"` of at least 3.1.4, that your `compileSdkVersion` is 27 or later, and that your `com.android.support` libraries are at least version 27.1.1. After these are verified or updated make sure it will build properly then start adding the implementations.

- Add to your module's build.gradle file under the "dependencies{ }" section.  
`implementation 'android.arch.persistence.room:runtime:1.0.0'`  
`annotationProcessor "android.arch.persistence.room:compiler:1.0.0"`
- Add to your module's build.gradle file under the "dependencies{ }" section.  
`implementation 'com.github.scribejava:scribejava-apis:6.8.1'`
- Add inside the "allprojects { repositories { } }" section of the build.gradle file of the project.  
`maven {url "https://s3.amazonaws.com/repo.commonware.com"}`  
Add to your module's build.gradle file under the "dependencies{ }" section.  
`implementation "com.commonware.cwac:saferoom:0.3.2"`  
If you are using proguard on your application you may also need to add:  
`-keep class net.sqlcipher.** { *; }`  
`-keep class net.sqlcipher.database.** { *; }`
- Add to your module's build.gradle file under the "dependencies{ }" section.  
`implementation 'com.google.code.gson:gson:2.8.5'`

## Using the Offline StampSDK for Android

The StampSDK is very lightweight and only performs a couple of operations. When you stamp the screen, it processes the recorded stamp data, and returns to `onGetSerialResult`. Your return parameter is a `SnowShoeResult` object.

The `SnowShoeResult` object includes the following properties which you can access with basic getter methods (i.e. `getCreated()`):

- `"created"` - the date this was created.
- `"error"` - a `SnowShoeError` object which only has information if an error occurred, and contains `"code"` and `"message"` properties. If there is no error, this will be `null`.
- `"receipt"` - a GUID string receipt number that is given to each stamp transaction.
- `"stamp"` - a `SnowShoeStamp` object which includes the stamp serial number in a `"serial"` property. If a stamp is not found, this will be `null`.

`syncStampInfo()` - Updates the local stamp data from the cloud and returns to `onSyncCompleted`. Returns a `bool` which lets you know if it successfully updated the local stamp data. This method will keep track of the last time it was run and only request updated stamp data from the cloud.

`refreshStampInfo()` - Reloads all stamp table on your device from the cloud instead of just updating from the last sync.

**NOTE:** Make sure to run sync/refresh every time or every few times that the app runs to ensure the local stamp data remains up to date.

Reading Logcat Info:

While you are using the stamp in your application, there will be messages printed in your Logcat that will give information about what is happening in the application as you stamp the screen and sync to the cloud data. All the messages run with a tag of `"SnowShoeView"`.

The Logcat messages are most useful for self service help. If you run into problems with getting unwanted results when stamping, go through all steps of syncing and stamping while watching for `"SnowShoeView"` Logcat messages.