Mobile Application Builder-Android Guide
Oracle Banking Digital Experience Cloud Service
Release 25.1.0.0.0

Part No. G27932-01

April 2025

ORACLE

Mobile Application Builder-Android Guide

April 2025

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax:+91 22 6718 3001

www.oracle.com/financialservices/

Table of Contents

ORACLE

1. Preface

1.1 Purpose

Welcome to the User Guide for Oracle Banking Digital Experience. This guide explains the operations that the user will follow while using the application.

1.2 Audience

This manual is intended for Customers and Partners who setup and use Oracle Banking Digital Experience.

1.3 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit, http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.4 Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1.5 Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
Italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1.6 Screenshot Disclaimer

The images of screens used in this user manual are for illustrative purpose only, to provide improved understanding of the functionality; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.

1.7 Acronyms and Abbreviations

The list of the acronyms and abbreviations that you are likely to find in the manual are as follows:

Abbreviation	Description
OBDX	Oracle Banking Digital Experience

2. OBDX Servicing Application

2.1 Prerequisites

OBDX Android App is supported only on versions n (current) and n-1 release.

a. Download and Install node Js (will be downloaded to default path)

b. Install node js from https://nodejs.org

c. Download and Install Android Studio

d. Download and install Android Studio from https://developer.android.com/studio/index.html

e. Download and Install Android platforms

f. Update Android SDK to latest API Level.

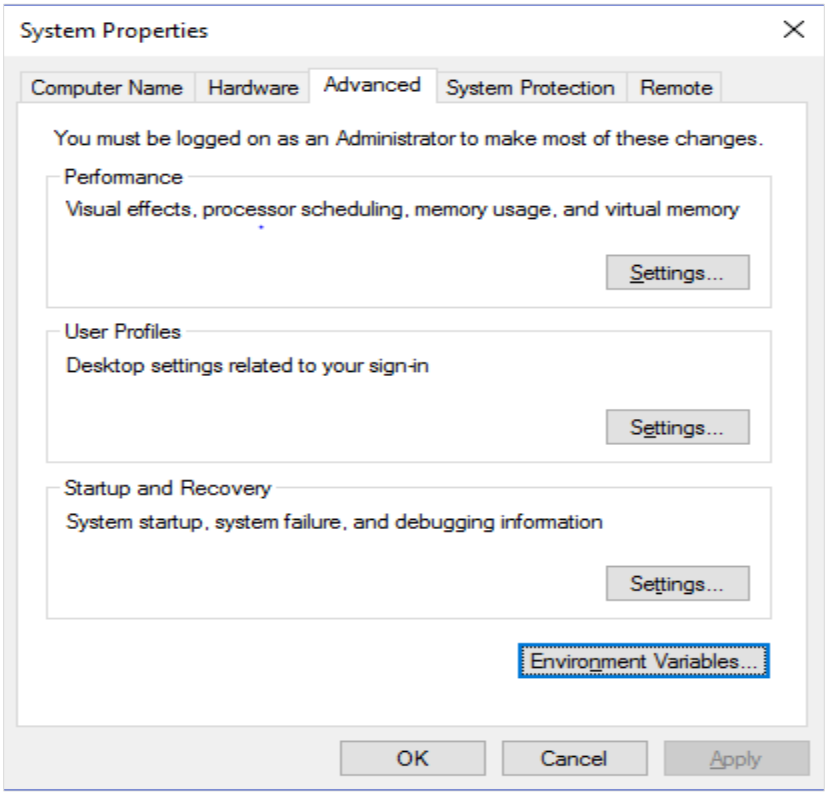
g. Gradle Version: gradle-7.5

h. Android Gradle Plugin Version (7.4.2): 'com.android.tools.build:gradle:7.4.2' or above

i. Set Environment variables

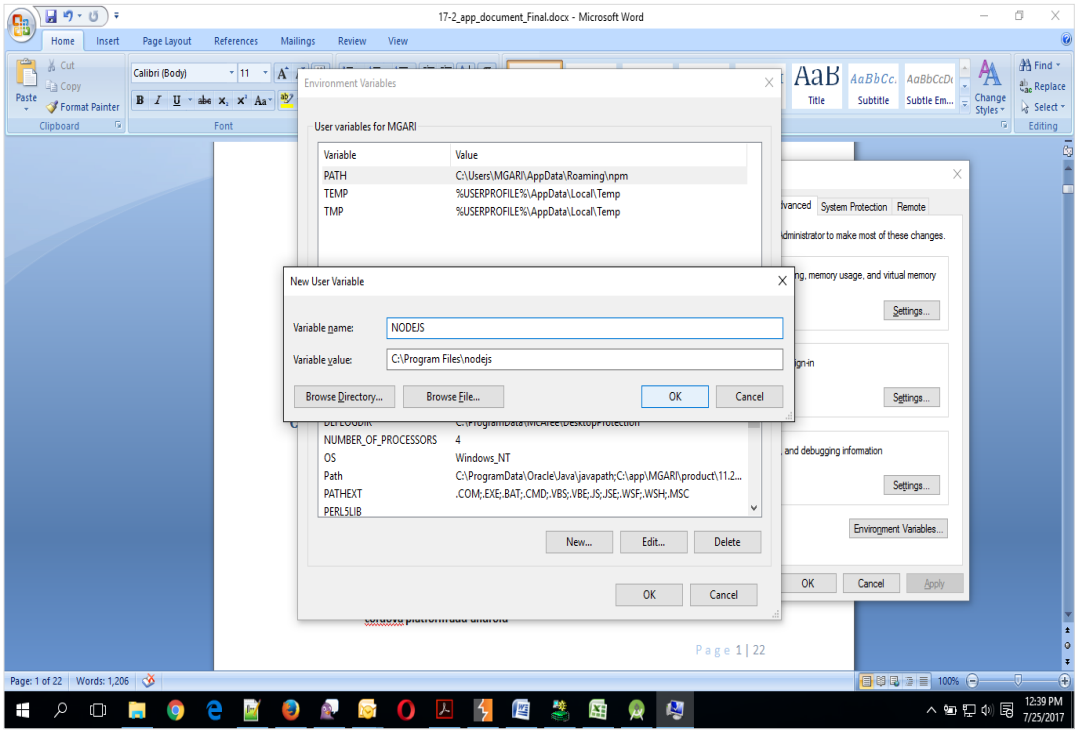
j. Set following system variables:

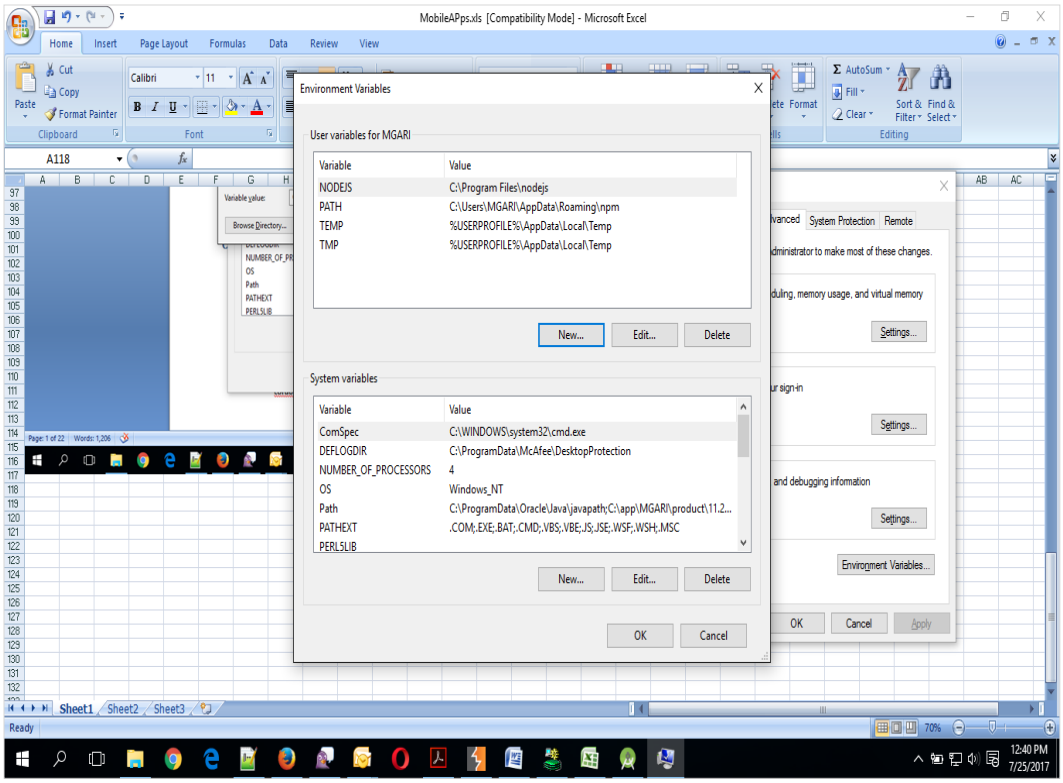
1. Click on Windows key and type Environment Variables.
2. A dialog box will appear. Click on the Environment Variables button as shown below



3. NODEJS <nodejs_path> Example: "*C:\Program Files\nodejs*".

k. Add the above variables in "*PATH*" system variable.





2.2 Create project using Remote UI

a. Index.html changes (use Android Studio or any other editor)

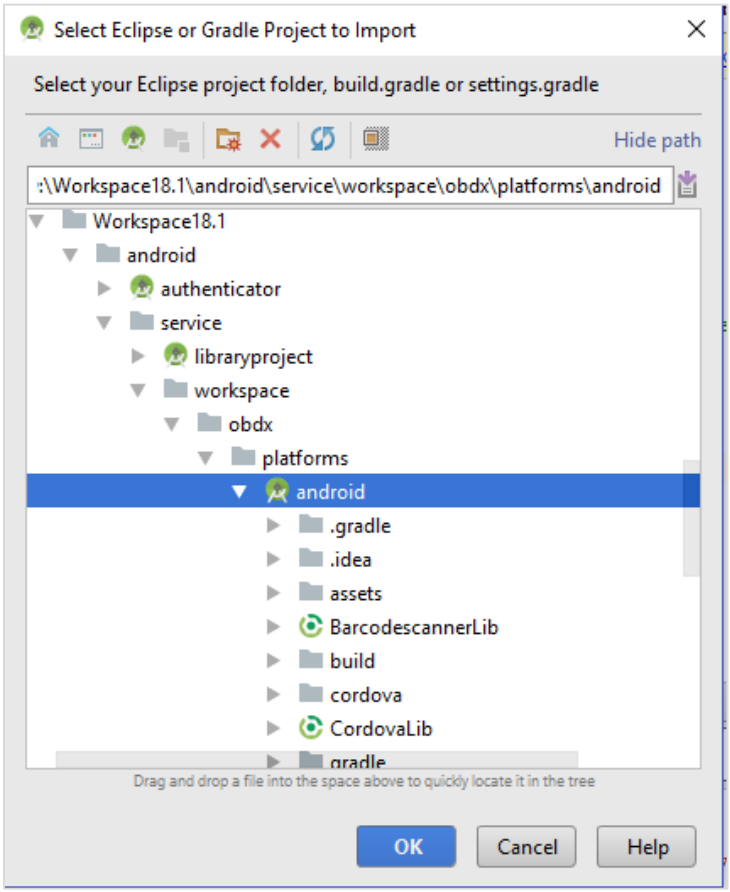
- Update the server URL in app.properties against KEY_SERVER_URL key. This is the URL where the UI is also hosted.

  After this proceed to **2.4 Importing in Android Studio** directly.

2.3 Importing in Android Studio

Open Android Studio

1. Import zigbank**/**platforms**/**android in android studio by clicking on Open an Existing Project.



2.4 Widget Functionality

Widgets are Android native feature. Below widgets are available in the application

1. All Accounts Widgets – Widget, showing all accounts balances & account numbers.

2. Account Details Widget - Widget, showing account balance of default account and last 5 transactions of the same account, can be added to the phone home screen. If default account is not set, then the details of the account fetched first is shown.

3. Multi-Functional Widget – Widget showing default account balance. If default account is not present, it shows details of account fetched first. Additionally, it has option to scan to pay feature

4. Scan to Pay Widget – Widget which allows to scan to pay.

Pre-requisite:

Quick Snapshot feature needs to be enabled in the app application from the login screen. (Refer function doc - User Manual Oracle Banking Digital Experience Quick Snapshot.docx)

Enable below property in app.properties file

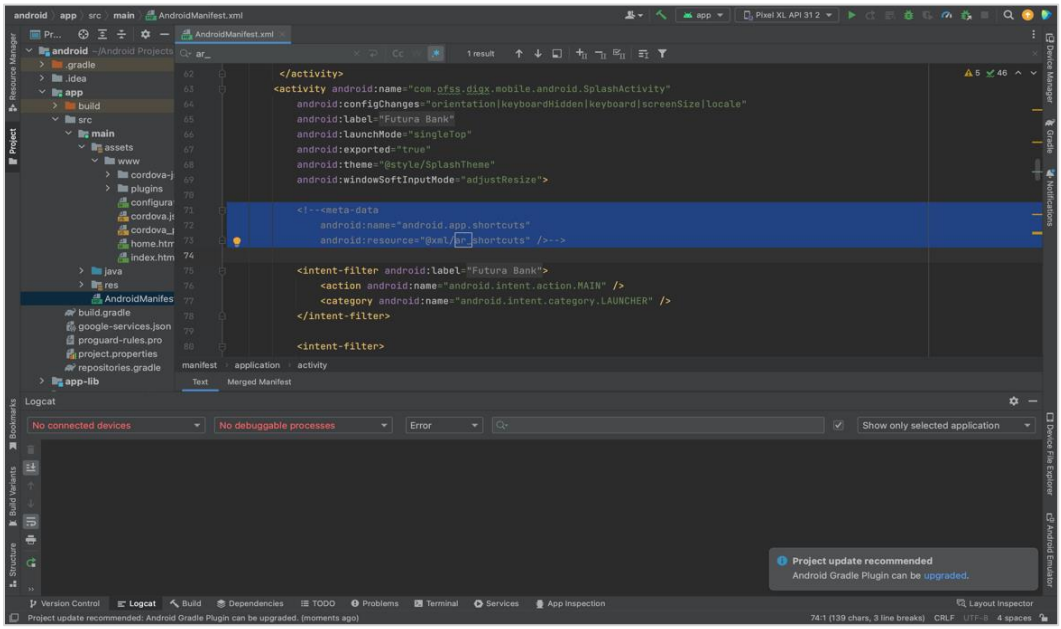
<bool name="ENABLE_WIDGET">true</bool>

If bank does not want this feature, then they can disable this by making above flag to false.

2.6 Scan to Pay from Application Icon –

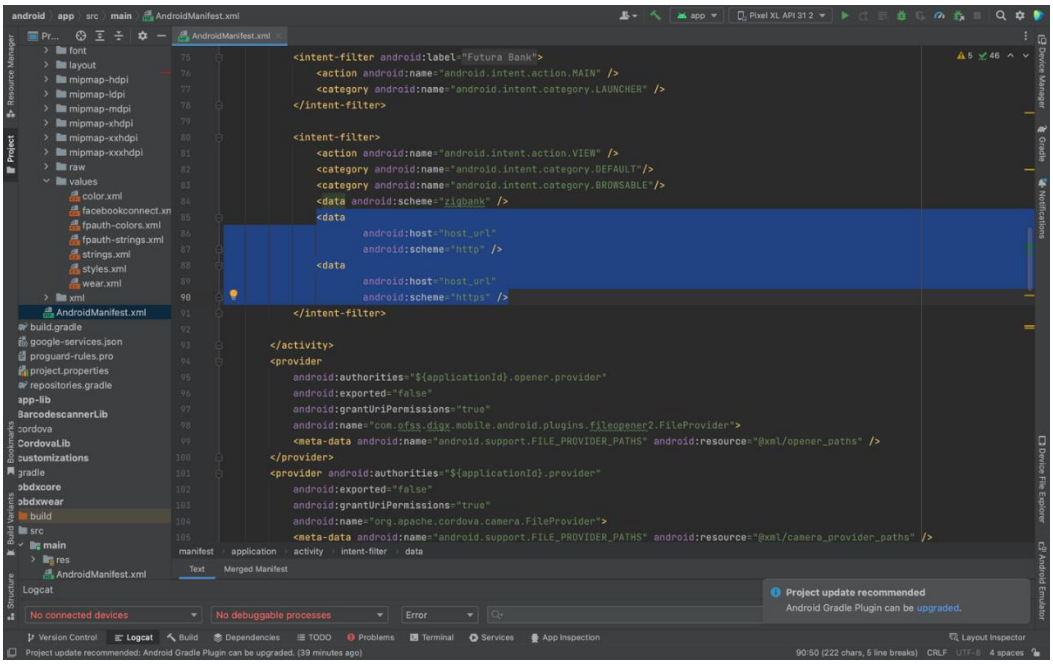
Users can long press on bank's application icon on home screen and click on scan-to-pay option to scan QR and make payments.

To enable this feature uncomment below from app's AndroidManifest.xml



2.5 Deeplinking - To open reset password, claim money links with the application

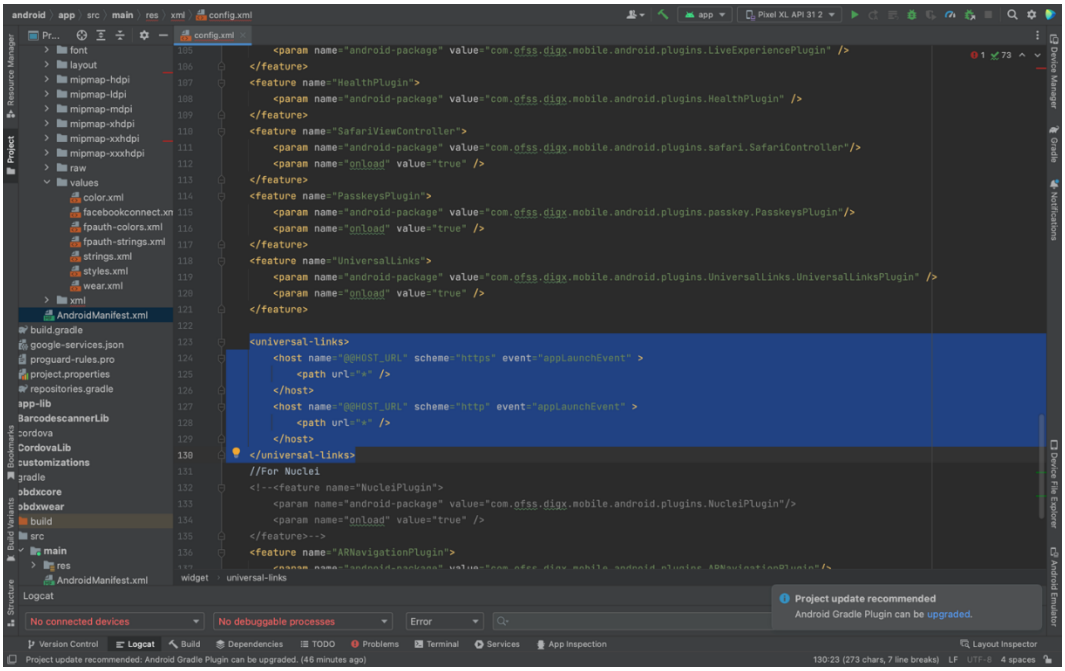
Add host url under data tag in app's AndroidManifest.xml as,



Note: Add host url without https or http.

For example: If your deeplink url is https://exmple.com/test then you can add only example.com in the data tag

Similary you can add the same host url in app's config.xml under universal-links tag as,



2.6 Device Registration and Push Registration Functionality

In this version, only one device is allowed to be registered for alternate login for the same username. If user tries to register another device with same username for alternate login, then the previous registration on other devices will be removed. User will get an error message if he/she tries to use PIN/PATTERN/BIOMETRIC on the de-registered devices.

While user registers his second device or same device again (by re-installing the application), a popup will appear to notify the same.

If user confirms, then the current device will be registered, and all previous registrations will be removed.

If user cancel, the process is exited.

Also, in this version, only one device is allowed to be registered for push.

Bank can allow multiple devices to be registered for same username in their setup by setting below two configurations:

ALLOWED_DEVICE_COUNT to any value between than 1 and 100.

- 1 will allow on one device registration.
- 100 will allow more than one device registration

ALLOWED_PUSH_DEVICE_COUNT any value between 1 and -1

- 1 will only one one device to be registered for push.
- -1 will only multiple devices to be registered for push

2.7 Location Tracking Metrics

This is optional. Bank needs to do if they need location tracking metrics for monitoring location-based data.
ALLOW_LOCATION_SHARE
By default, the value is false. If set to true, user will get location permission prompt to allow location tracking. It can be enabled if user's location needs to be tracked.

2.8 Displaying Rate Option to Redirect to Playstore Page

This is optional. User can have an option ("Rate Us") in settings to display Play Store rating for the application. This option can be enabled/disabled from UI.

Note: App should be listed on playstore before adding this functionality.

2.9 Enabling Force Update

This configuration is optional.

To notify users of a new application version available on the Play Store, consider these options:

1. Within App, when the App detects a new version, prompt users suggesting an update.
2. The flag checks for updates and displays a cancellable popup to the user to update their application.
3. To implement this with the flag `isAppUpdateManagerEnable` to true in `RootCheckFlags`.

Note: Ensure that App update functionality works only when the App is downloaded from the Play Store or via Internal App Sharing.

4. Follow the steps to check force app update: https://developer.android.com/guide/playcore/in-app-updates/test#internal-app-sharing

2.10 Splash Screen Migration

The splash screen implementation is migrated according to latest document from google:

https://developer.android.com/develop/ui/views/launch/splash-screen/migrate

Steps to generate xml file for svg to be used in splash:

1. Right click on /android/app/src/main/res/drawable and select New/Image Asset

2. Select the path to the svg. (note that svg of bank logo is required. PNG and other image extensions won't work)

3. Resize the image from the scroll bar so that the icon is well inside the circle.

4. Keep all the configurations as it is and create the svg.

5. It will directly generate xml files for different resolution.

6. Refer to the foreground xml in styles.xml @drawable/ic_launcher_foreground

2.11 App Update Manager

Note: In App Update functionality will be work only for the apps which will be downloaded from play store/internal app sharing.

Follow below doc to test the in app update functionality.

https://developer.android.com/guide/playcore/in-app-updates/test

2.12 Auto OTP Configuration

Note: User consent otp popup will not be seen from now. OTP will be directly auto populated in the field once the sms is received. For this, 11 digit app hash code is required.

To Generate the 11 digit application hash, follow below doc -

https://developers.google.com/identity/sms-retriever/verify#computing_your_apps_hash_string

f you are signing your app with your own keystore then download below script -

https://github.com/googlearchive/android-credentials/blob/master/sms-verification/bin/sms_retriever_hash_v9.sh

Then run below command -

./sms_retriever_hash_v9.sh --package YOUR-PKG-NAME --keystore YOUR-KEYSTORE-PATH

Add this 11 digit app hashcode at the end of each text msg template

You will get the msg template in below table -

select * from digx_ep_msg_tmpl_b where txt_msg_tmpl like '%OTP%' and destination_type like 'SMS';

Sample SMS will be look like below –

Your OTP is 1111 FA+9qCX9VSu

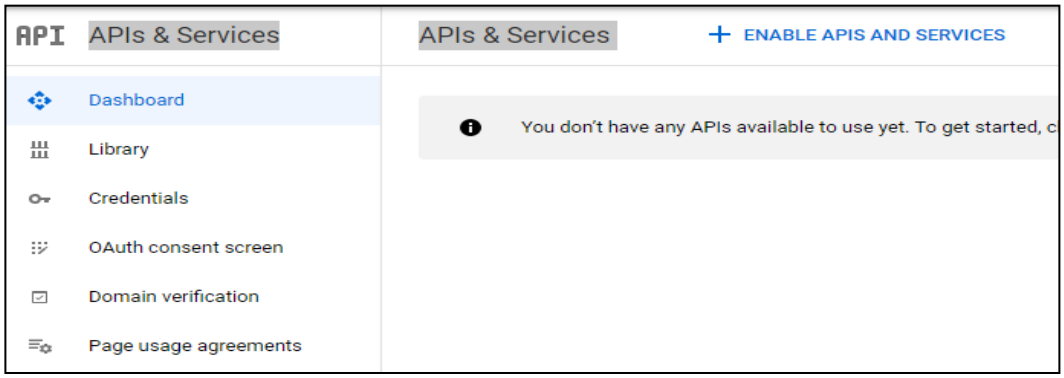
3. Google Play Integrity

a. Go to URL https://console.developers.google.com/

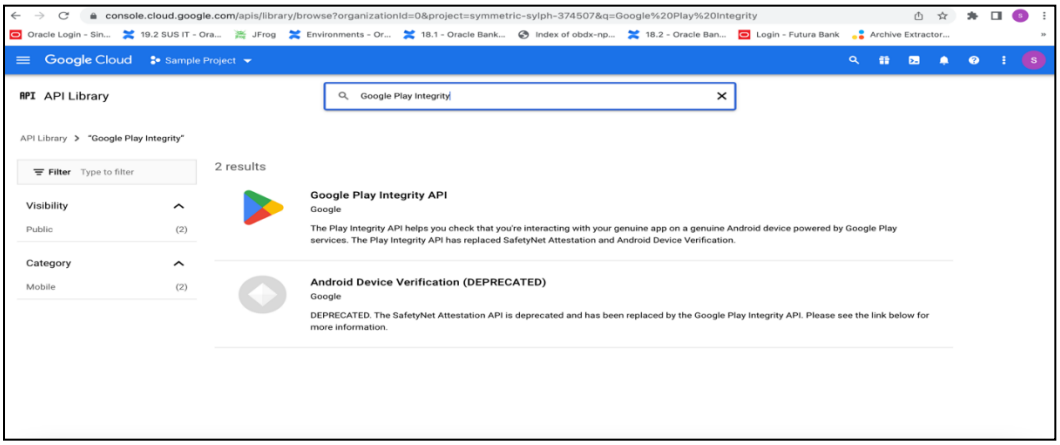
b. Create a new Project and set name of you project

c. Choose **'API's & Services'** option from side bar.

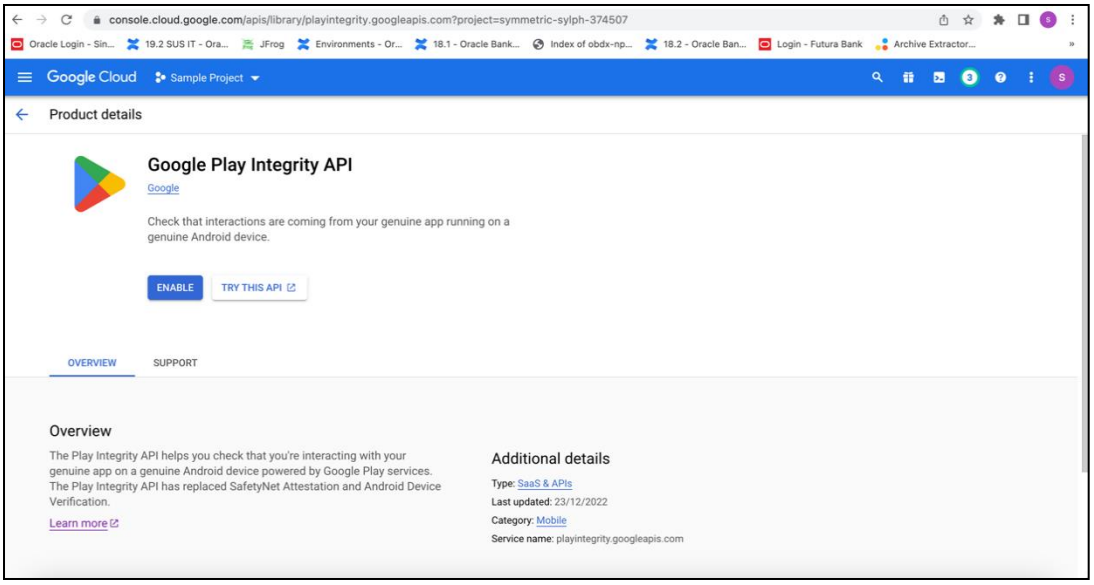
d. In API's & Services > Dashboard > Choose **'Enable APIS AND SERVICES'**.



e. This will redirect to **'Library'** where we need to search **'Google Play Integrity API'.**

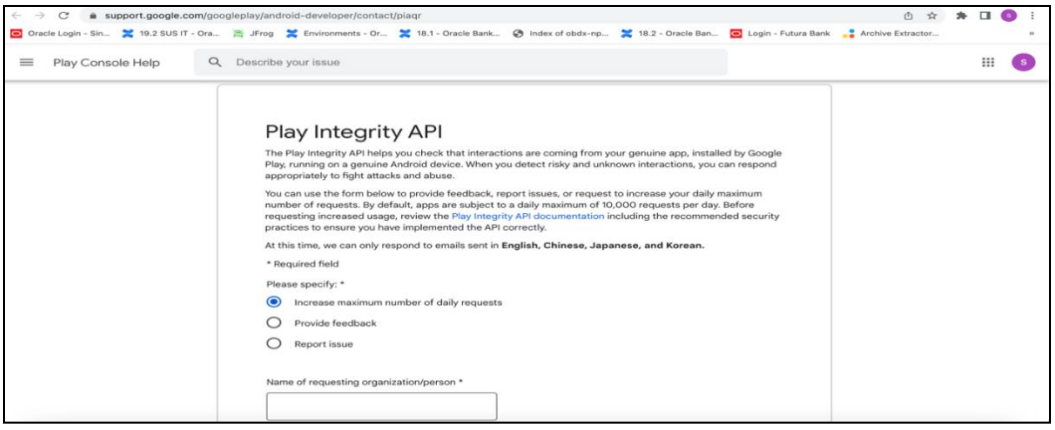


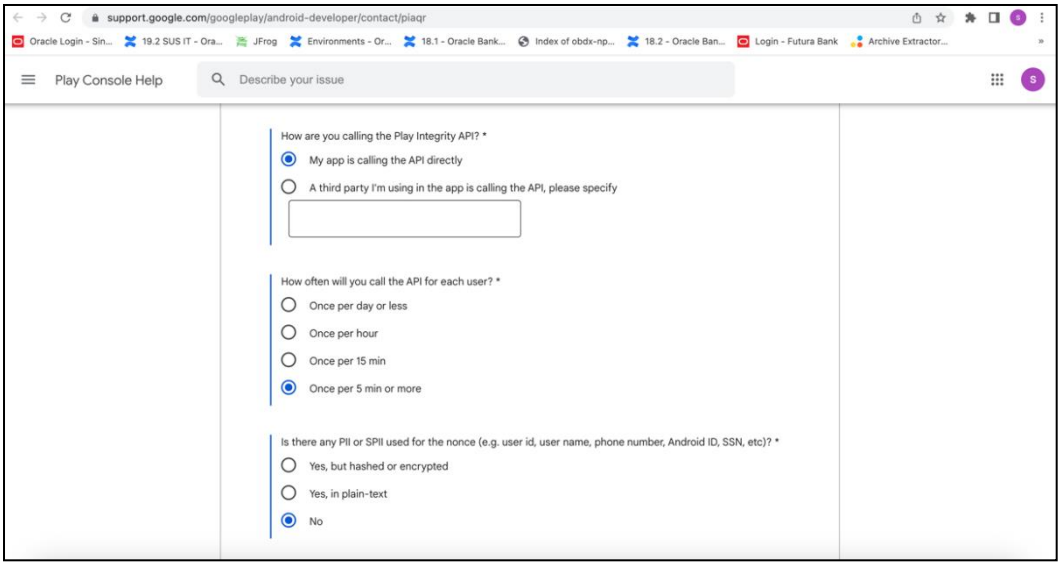
f. Click on Google Play Integrity API and enable it

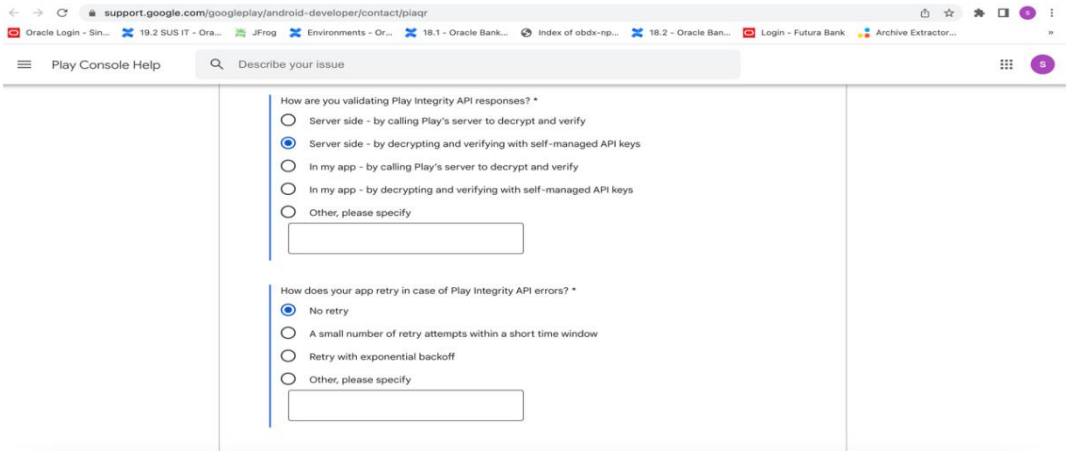


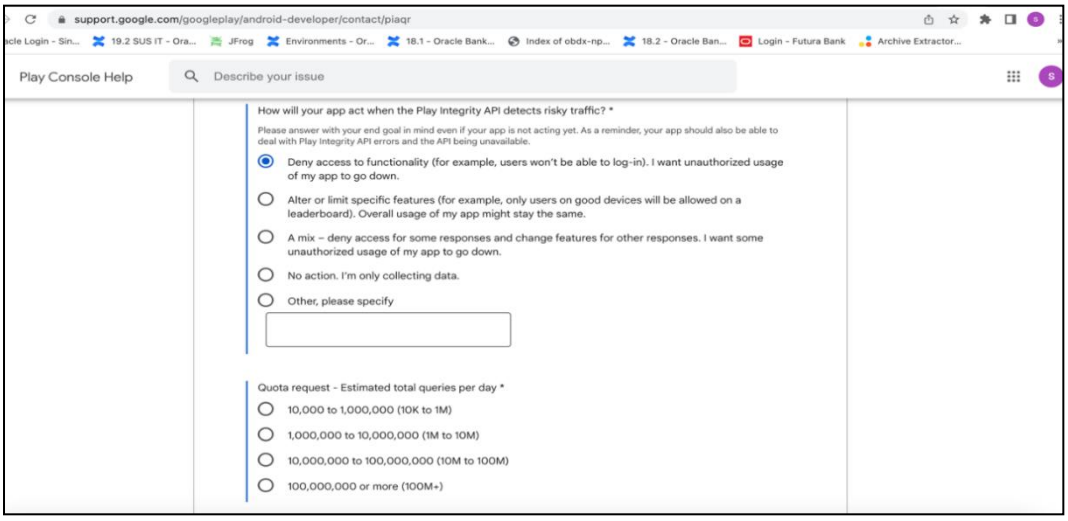
g. If the application usage is high, the quota request form needs to be submitted. Fill quota request form from below site. Also select below options.

https://support.google.com/googleplay/android-developer/contact/piaqr







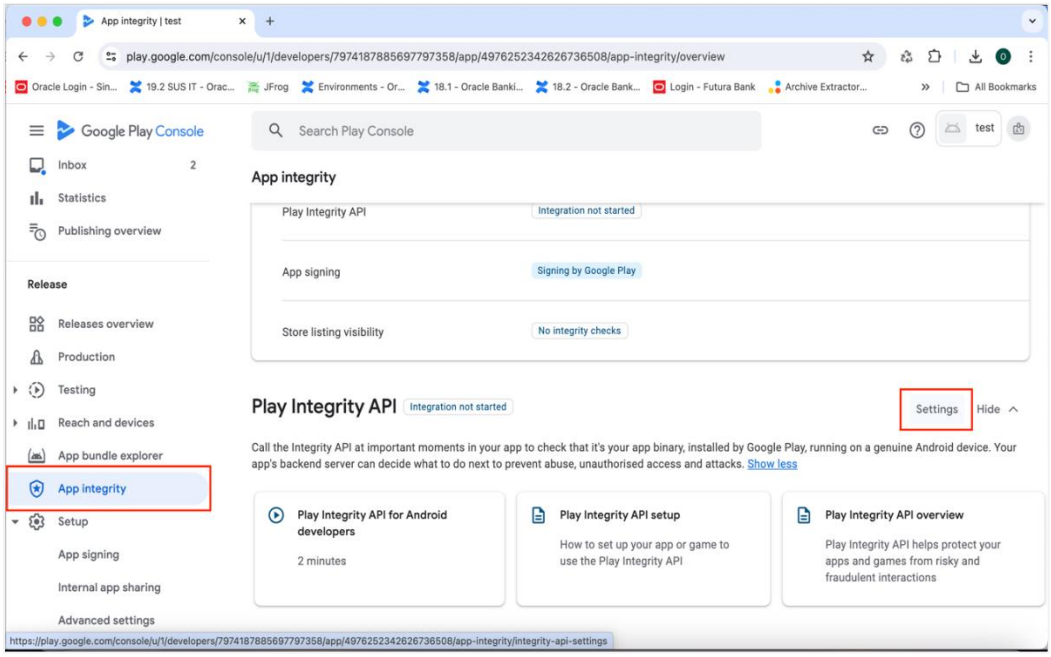


Quota request - Estimated total queries per day * → The approximate load, Play Integrity API is called once each time the app in opened

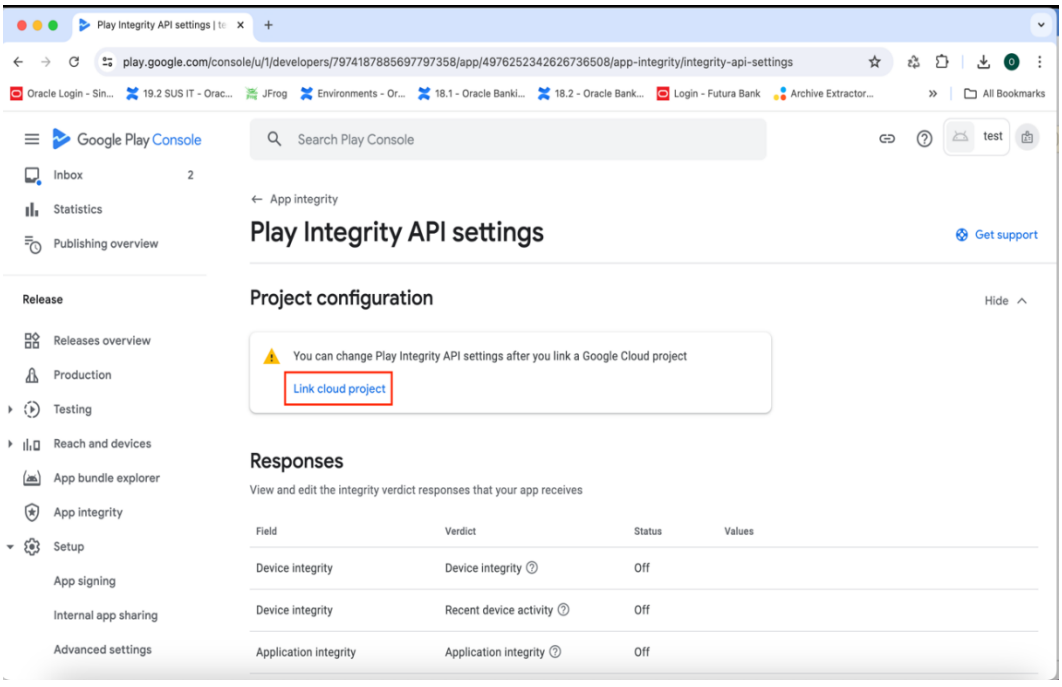
Quota request - Estimated peak queries per second → Leave blank

h. To enable Play Integrity responses please follow below steps-

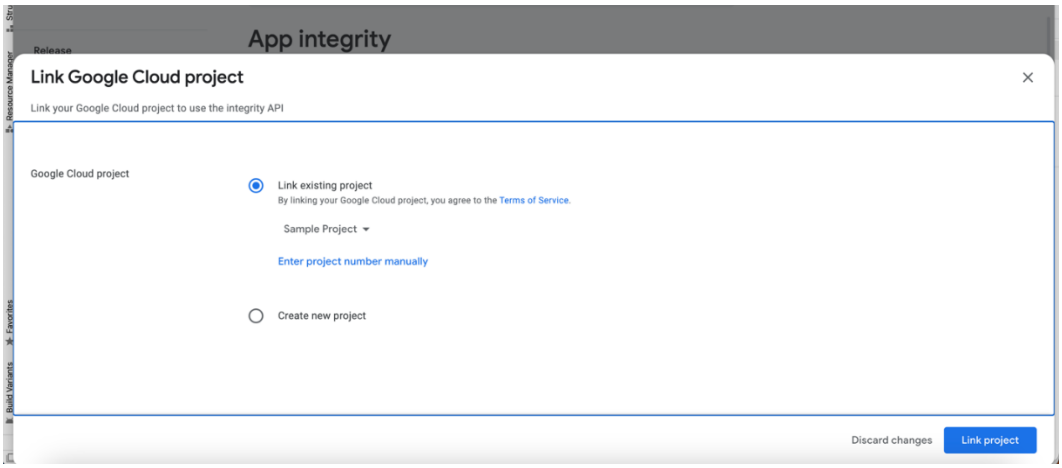
Go to Google Play Console->Side Menu ->App Integrity



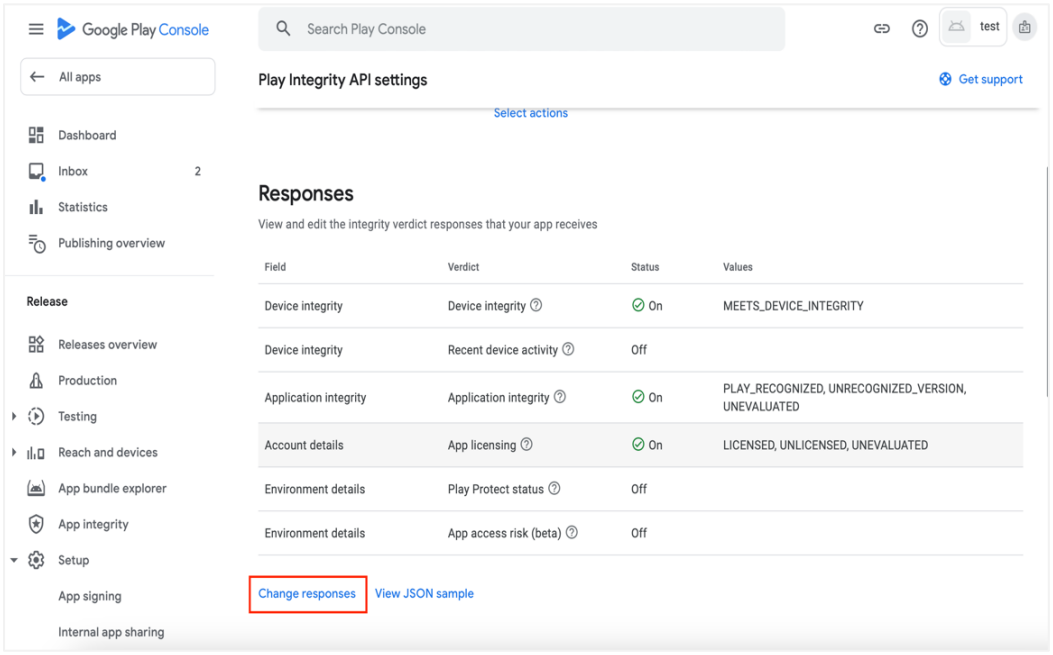
Click on **Settings.**



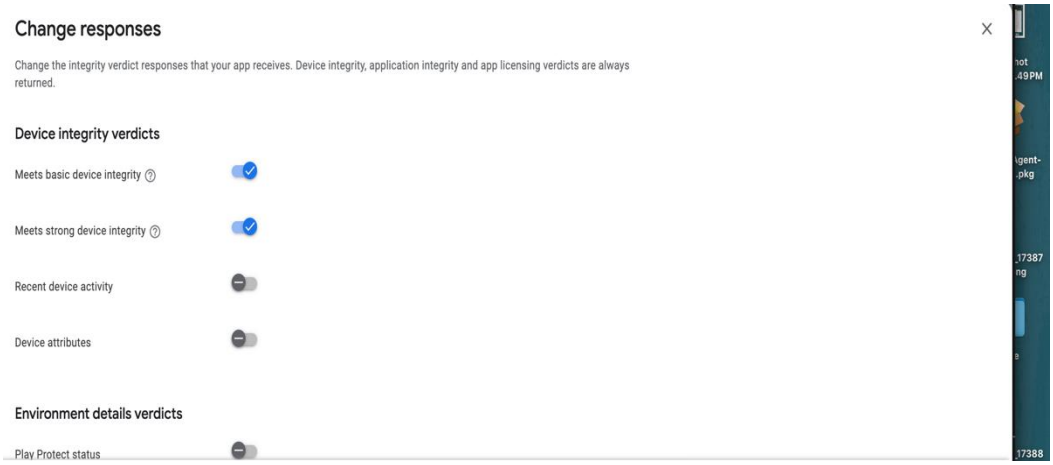
Click on **Link project** and then link your existing google cloud project. If it is not created then create new and link the same.



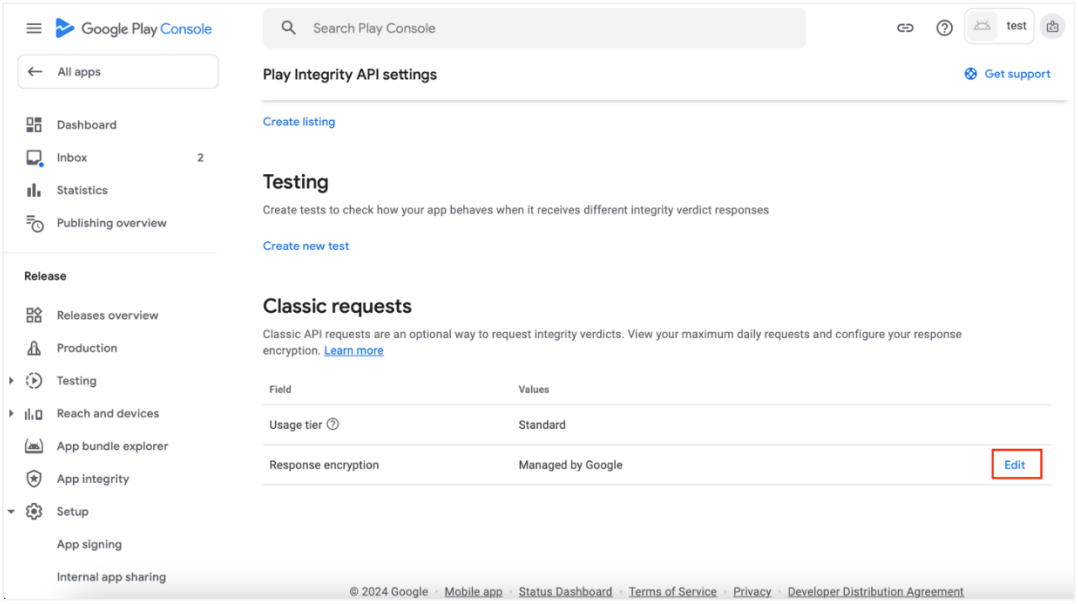
i. Scroll down on the same screen and click on **Change Responses**.



j. Enable the Meet basic Integrity & Meets Strong Integrity option and save the changes.



k. Scroll down on the same screen and click on **Edit** button of classic requests section



l. In the window that appears, select **Manage and download my response encryption keys** and follow below steps to generate response encryption keys-

a. Create a new private-public key pair. RSA key size must be 2048 bits using below command-

openssl genrsa -aes128 -out your_path/private.pem 2048

Then use your password phrase for creating private.pem and also use the same password for verifying the private.pem. Then hit the below command.

openssl rsa -in your_path/private.pem -pubout -out your_path/public.pem

Enter the same password which you have used while creating private.pem. These two files will now appear on your mentioned path. Then upload the public.pem file on the window which was appeared after clicking on Manage and download my response encryption keys option.Once you upload the public.pem file it will automatically download your_app_pkg_name.enc file. Then hit below command as,

openssl pkeyutl -decrypt -inkey your_path/private.pem -pkeyopt rsa_padding_mode:oaep -in your_path/com.demo.xz.enc > your_path/api_keys.txt

Enter the password for private.pem. It will create api_keys.txt file on your path. It must be consist of VERIFICATION_KEY and DECRYPTION_KEY.

b. Maintain this VERIFICATION_KEY and DECRYPTION_KEY in **DIGX**_FW_**CONFIG_ALL_B** table corresponding to the following keys respectively:

PLAY_INTEGRITY_ENCRYPTION_KEY and **PLAY_INTEGRITY_DECRYPTION_KEY**

An example query will be:

update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_DECRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_DECRYPTION_KEY';

update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_ENCRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_ENCRYPTION_KEY';

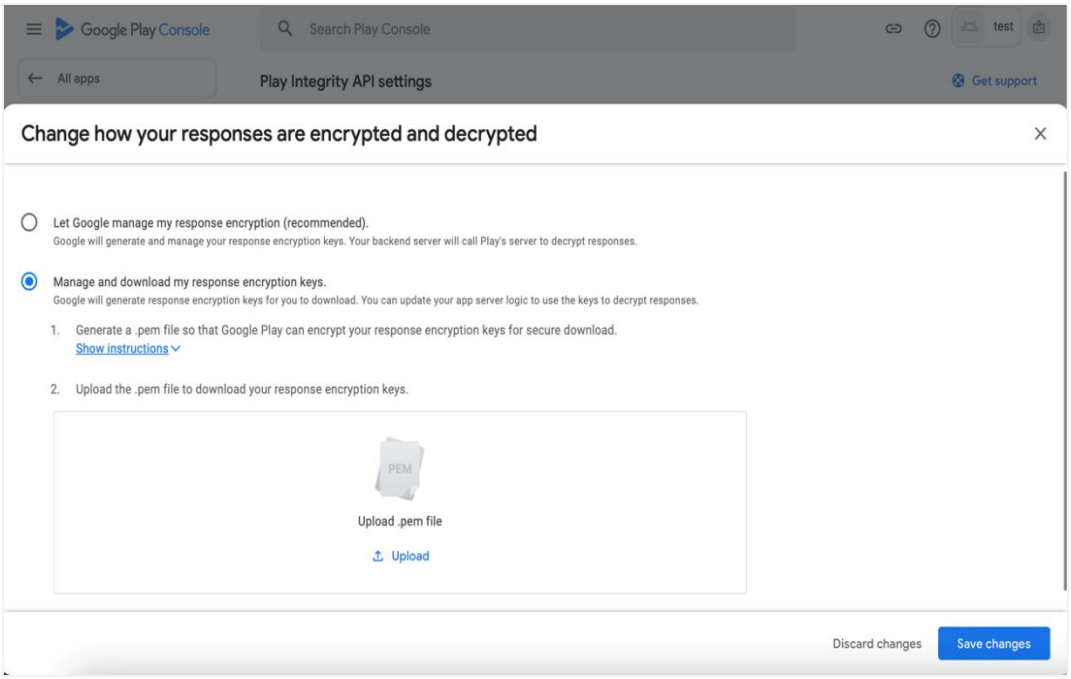
c. Similarly, Obtain the same keys for authenticator app by using above steps and then maintain those in **DIGX_FW_CONFIG_ALL_B** table corresponding to the following keys respectively:

PLAY_INTEGRITY_ENCRYPTION_KEY_AUTHENTICATOR and **PLAY_INTEGRITY_DECRYPTION_KEY_AUTHENTICATOR**

An example query will be:

update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_DECRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_DECRYPTION_KEY_AUTHENTICATOR';

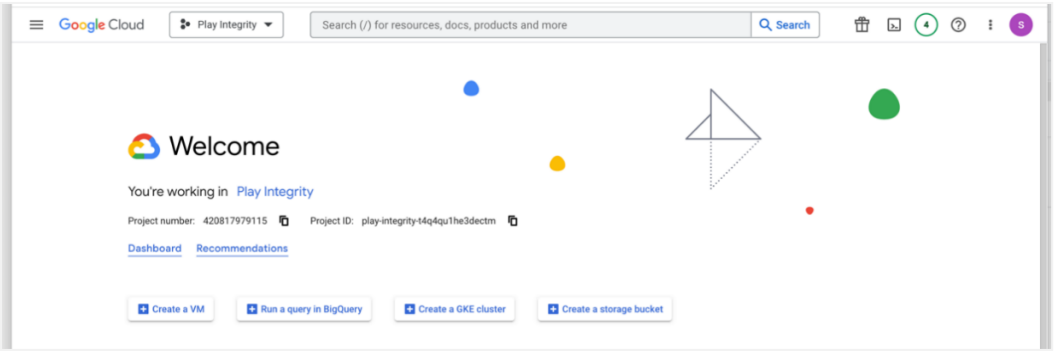
update DIGX_FW_CONFIG_ALL_B set prop_value = 'YOUR_ENCRYPTION_KEY' where prop_id = 'PLAY_INTEGRITY_ENCRYPTION_KEY_AUTHENTICATOR';



m. Add project number in below property of app.properties

<string name="GOOGLE_CLOUD_PROJECT_NO">@@GOOGLE_CLOUD_PROJECT NO</string>

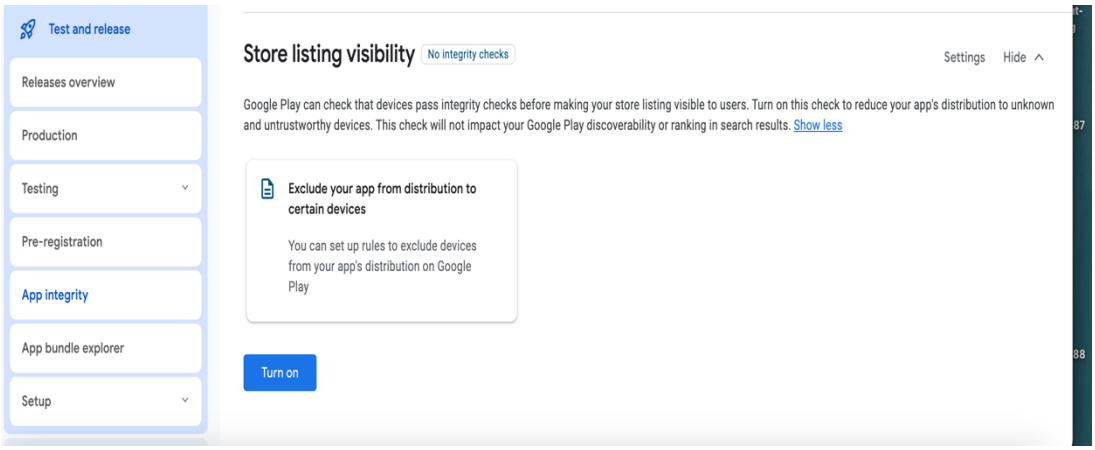
You will get the project number on google cloud console project



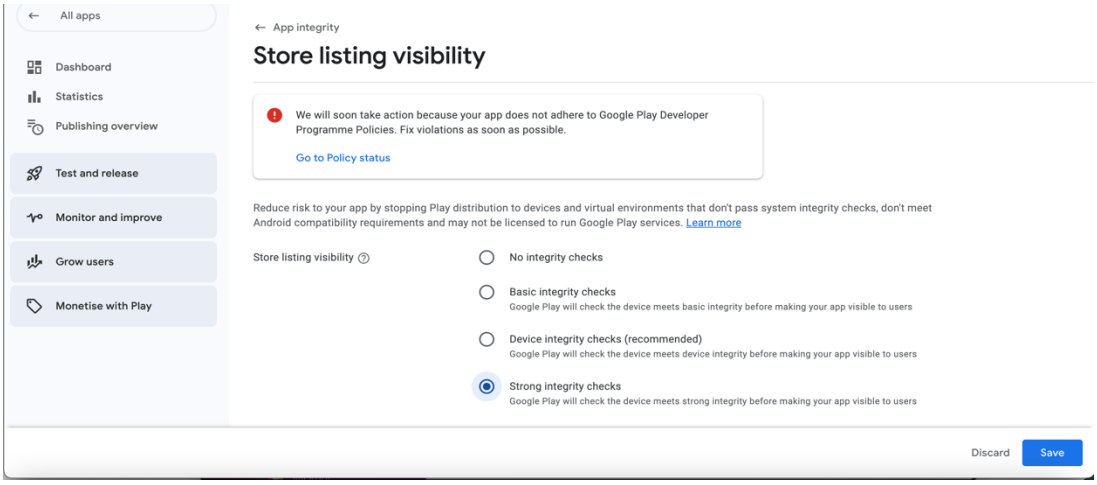
n. Mention the time in seconds to which app can hit the play integrity api. By default it is 300seconds but you can configure as per the requirement. Use below property in RootCheckFlags.java(workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/)

```
long playIntegrityAPICallTime = your_time_in_seconds;
```

l. Scroll down on the App Integrity Page. Navigate to Store listing visibility. Click on Settings button.



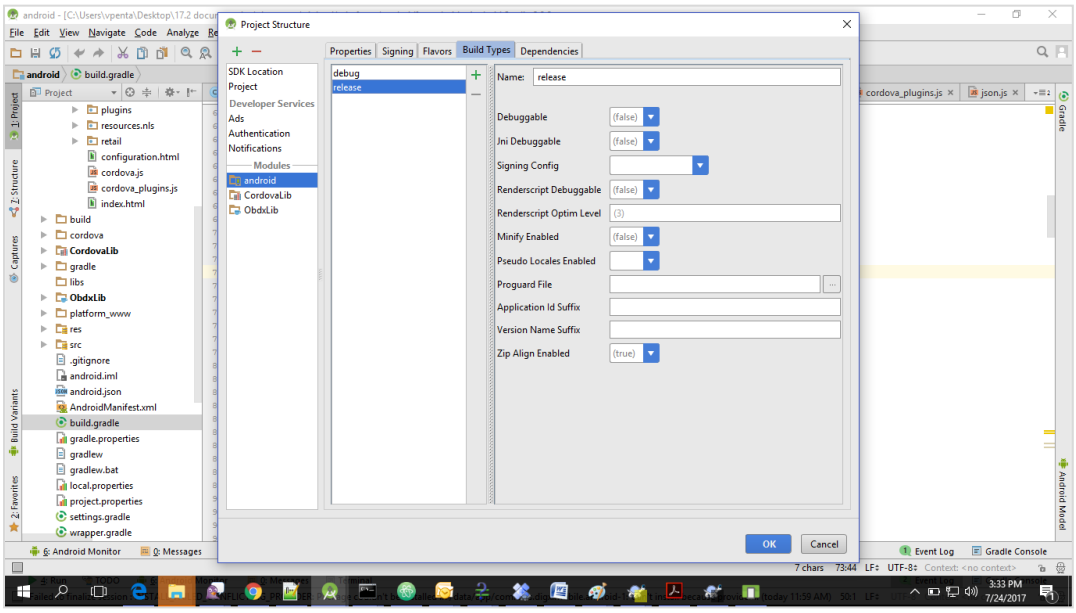
Select Strong Integrity checks option & Save



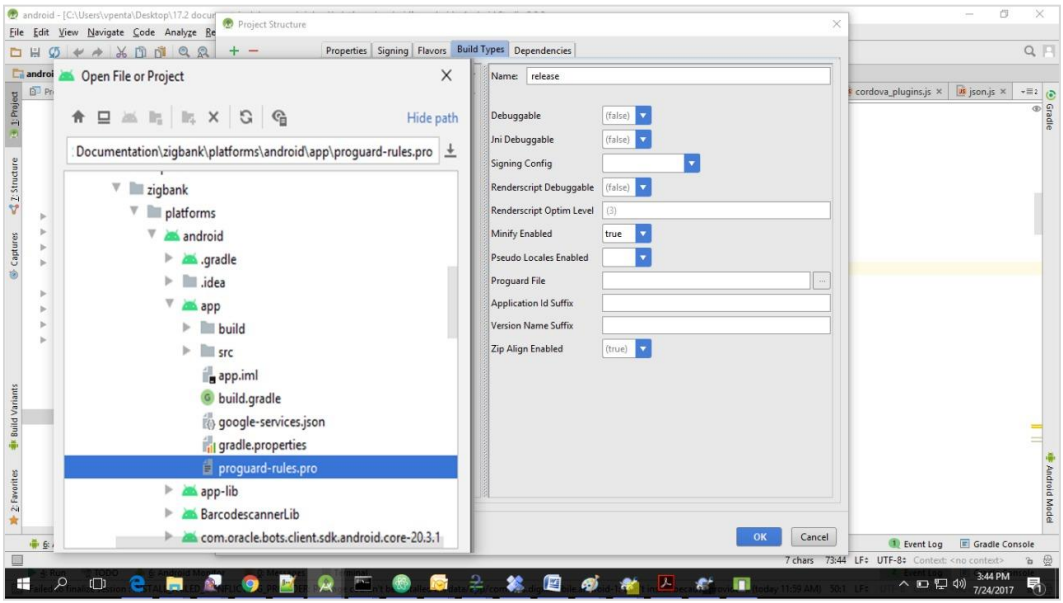
Note: By enabling this setting your app will not be listed on play store of rooted device.

4. Build Release Artifacts

1. Clean and Rebuild your project in Android Studio.
2. In Android Studio, on the menu bar Click on **Build -> Edit Build Types ->** select **release**



3. Set Minify Enabled -> True & click on Proguard File selection -> Navigate to proguard-rules.pro (zigbank\platforms\android\app)



4. Click on OK -> again click on OK.
5. Adding URLs to app.properties.xml (customizations/src/main/res/values/)
   a. Cloud Setup

SERVER_TYPE	OAUTH
KEY_SERVER_URL	Cloud OHS URL where UI is hosted
IDCS_URL	IDCS URL
ClientIdClientSecret	Base64 of ClientId:ClientSecret generated from cloud console.
IDCS_Scope	OAuth_Access/consumer::**Cookie**
Cookie	The value after the :: in IDCS_Scope is the cookie
WATCH_CLIENT_ID	Same as ClientIdClientSecret
SNAPSHOT_CLIENT_ID	Same as ClientIdClientSecret

6. To Enable SSL

   There are 2 levels of SSL checks added in the app. One is to check SSL on app launch only and another one is to check SSL for every api calls in UI.

   By default app launch SSL is enabled & UI SSL check is disabled. Bank can enable/disable SSL by using below properties.

ENABLE_SSL	true
ENABLE_SSL_FOR_UI	false

7. Enable/Disable Face biometric

   Below flag is use to enable or disable Face biometric for alternate login in OBDX app.

ALLOW_FACE_BIOMETRIC	true

   By default product support both biometric type i.e. Face & Fingerprint for alternate login.

Note: Face biometric option is shown by google's biometric api. It may possible that if your device has face registered, but in OBDX app its not showing up while registering biometric alternate login. This is because of sensors of your device is not compatible with the google's biometric api.

If you want to reset alternate login on Add/Remove of fingerprint in your device then disable this flag.

8. Domain Based Setup (This is same for OBDX servicing App and Authenticator App)

   To use domain based setup, enable below flag in app.properties file -

   <string name="DOMAIN_BASED_CATEGORIZATION">true</string>

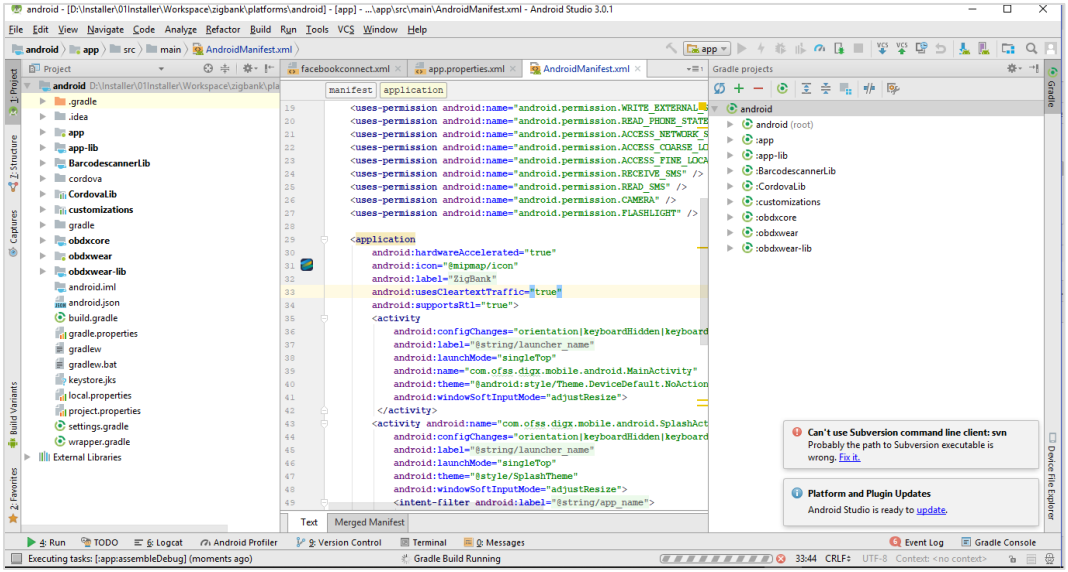
   If you are using local UI then enable below flag in config.js(platforms/android/app/src/main/assets/www/framework/js/configurations/config.js) file -

   domainDeployment: {

   enabled: true

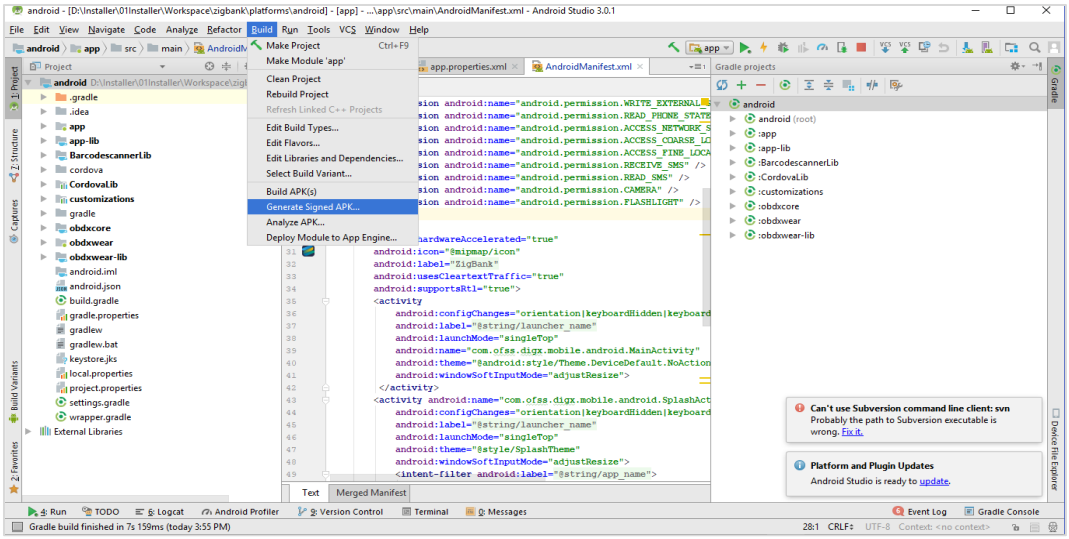
   }

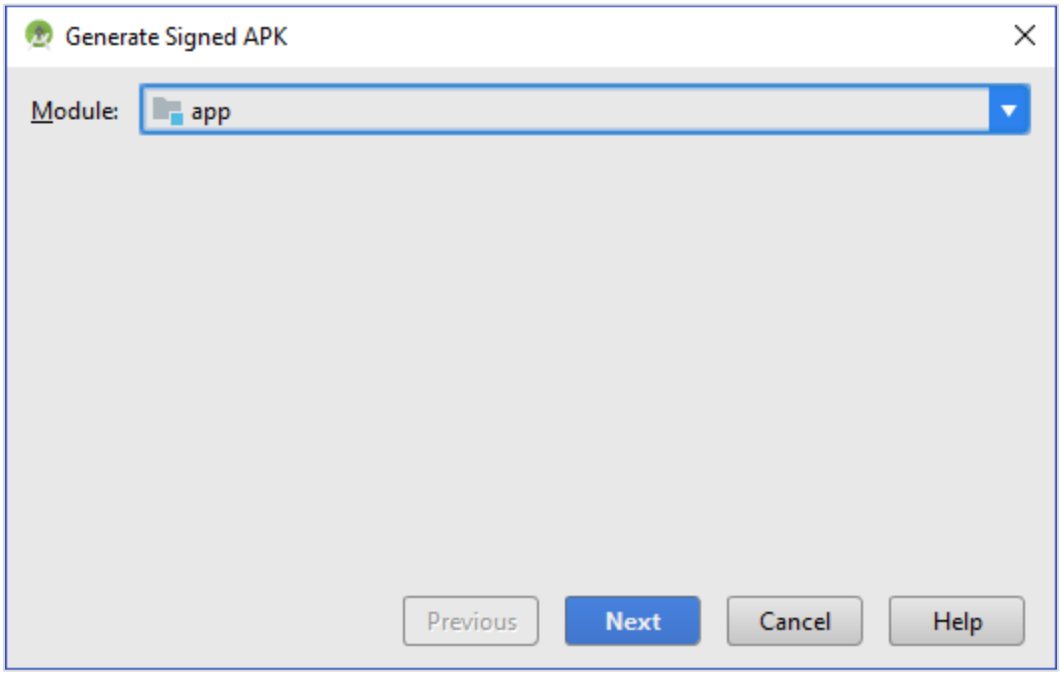
9. If using http protocol for development add (android:usesCleartextTraffic="true") to application tag of AndroidManifest.xml (on app & obdxwear target)

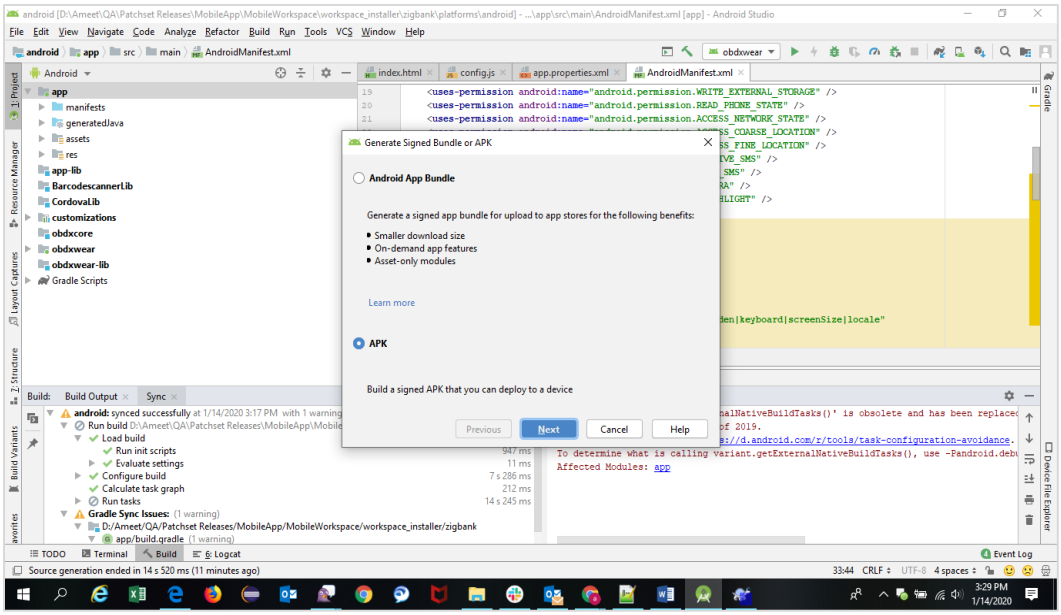


10. **For Generating Signed Apk:** To Generate release-signed apk as follows:

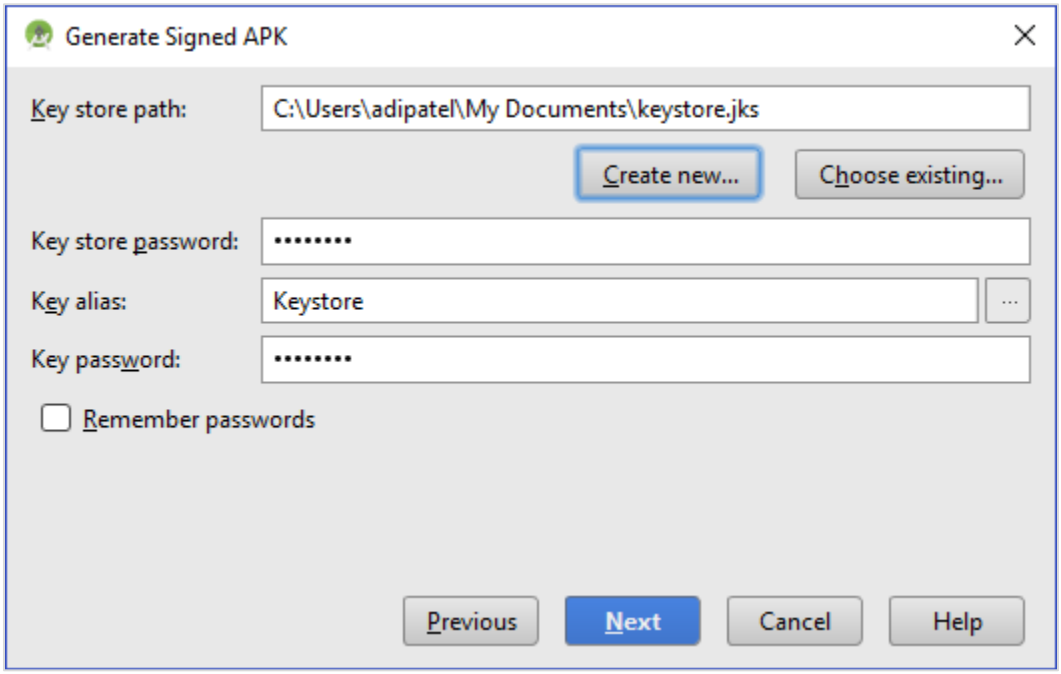
On menu bar click on Build -> Generate Signed Apk

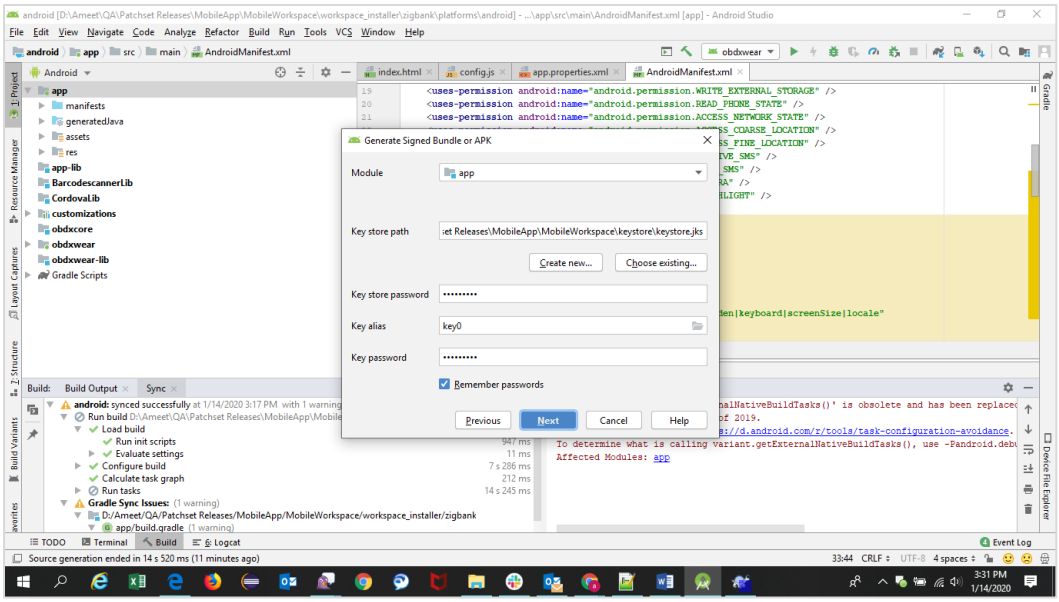

Build
Make Project
Make Module 'app'
Clean Project
Rebuild Project
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Select Build Variant...
Build APK(s)
Generate Signed APK...
Analyze APK...
Deploy Module to App Engine...
Can't use Subversion command line client: svn
Platform and Plugin Updates


Generate Signed APK
Module:
app
Previous
Next
Cancel
Help

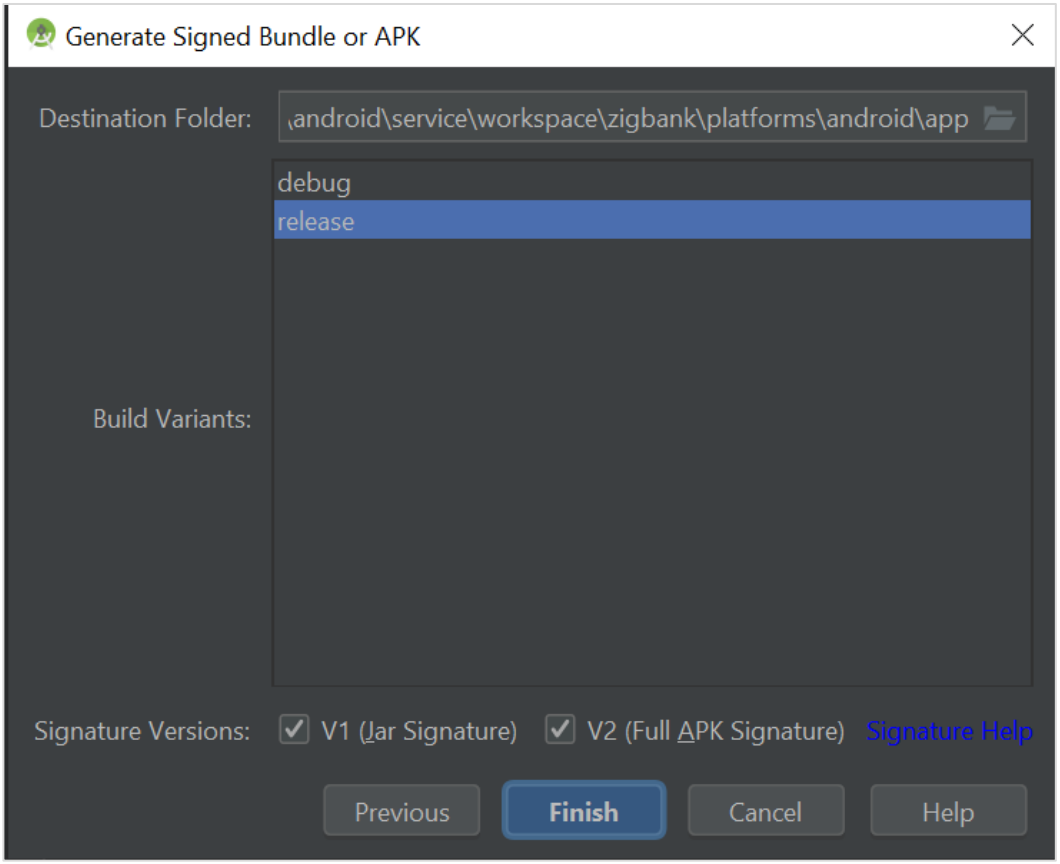


11. If you have an existing keystore.jks file then select choose Existing else click on Create New





12. Select **Build Type** as **Release**, **Signature Version as V1(JAR Signature) and V2(Full APK Signature)** and Change APK Destination folder if you want and click on Finish

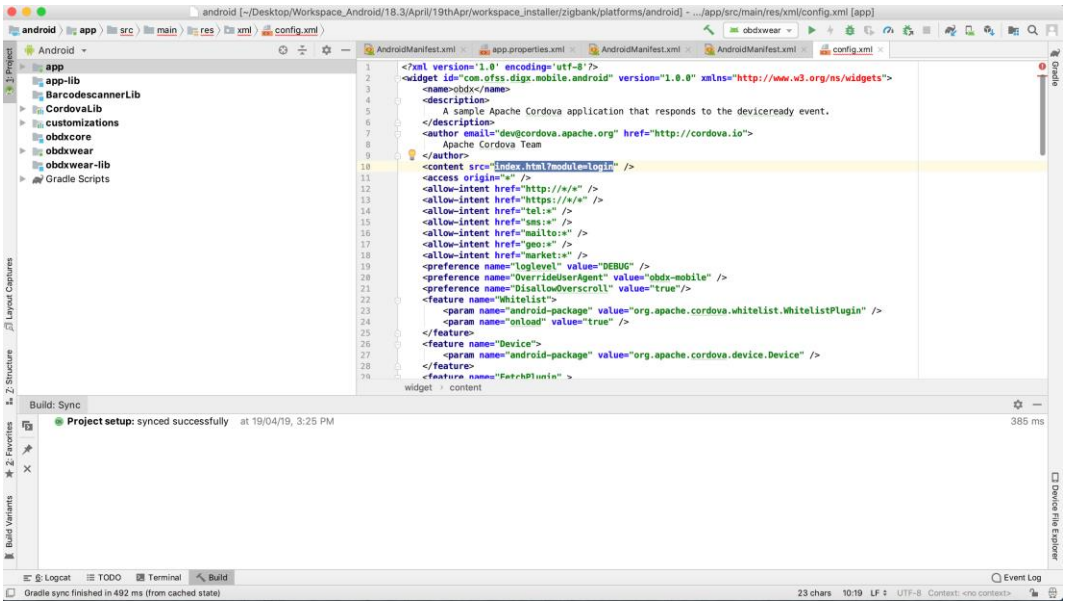


13. This will generate APK by the given name and destination folder. Default APK Destination folder is **zigbank\platforms\android\app\release**

14. Run the App and select Device or Simulator.

15. **Repeat same steps (From step 8 and obdxwear as module) for OBDX Wear App for Release Signing.** Use proguard-rules.pro from **workspace_installer\zigbank\platforms\android\obdxwear** using explorer. The select obdxwear as the module and follow same signing steps with same keystore.
16. The application has a config page at launch to enter the URL of the server (for development only). To remove this page, update the config.xml as shown below

The application has config page to add URL. This is for development purpose only and can be removed using below step. (Update content src tag)



17. Application will work on https only, there is no support for http url further.
18. To enable App widget, enable below flag in app.properties file:
19. <bool name="ENABLE_WIDGET">true</bool>Maintenance page configs-

    Enable below flag to show maintenance page when server is under maintenance

    <string name="SHOW_MAINTENANCE_PAGE">true</string>

    Also add the status code returned when server is under main in below property-

    <string-array name="MAINTENANCE_PAGE_STATUS_CODE">

      <item>Your Status Code</item>

    </string-array>

Note: You can add multiple status code.

20. To disable caching in app, make below flag to false

<bool name="ENABLE_CACHING">true</bool>

21. To disable ssl pinning in app, make below flag to false

<bool name="ENABLE_SSL">true</bool> in app.properties.

22. To disable ssl pinning for ui in app, make below flag to false

<bool name="ENABLE_SSL_FOR_UI ">true</bool> in app.properties.

5. OBDX Authenticator Application

1. This is an Authenticator Application which is used when bank has enabled Soft Token Authentication as Authentication mechanism for any transaction. This application basically supports one of below authentication:
   - HOTP: Random based Soft Token
   - TOTP: Time based Soft Token

2. Users should have this application installed and logged in and PIN is set before initiating any transaction which needs this token.
3. Based on the configuration set, user can any time log in with PIN and check the token and use that token for completing any transaction based on “Soft Token Authentication”

5.1 Authenticator UI (Follow any one step below)

5.1.1 Using Un-built UI

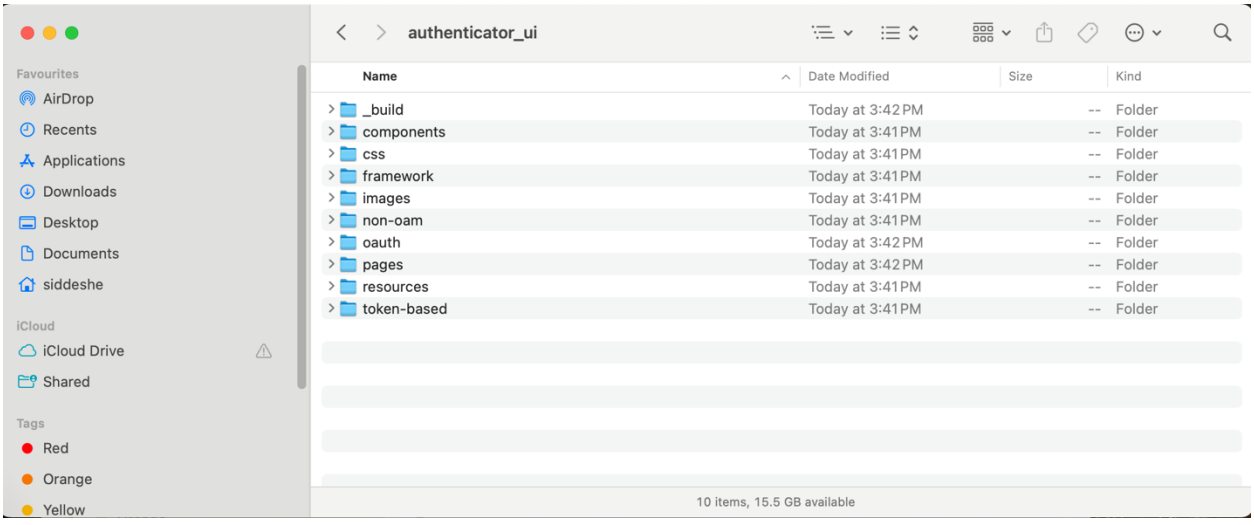
1. Extract authenticator_ui.tar.gz from OBDX_Patch_Mobile\authenticator\unbuilt_ui. Copy the “*oauth/login*” folder and replace it at the “components/modules/” location. This will replace the existing the login folder.

2. Copy the contents except _build folder to Authenticator workspace->platform/ios/www folder

5.1.2 Building UI manually

Extract authenticator_ui.tar.gz from OBDX_Patch_Mobile\authenticator\unbuilt_ui.

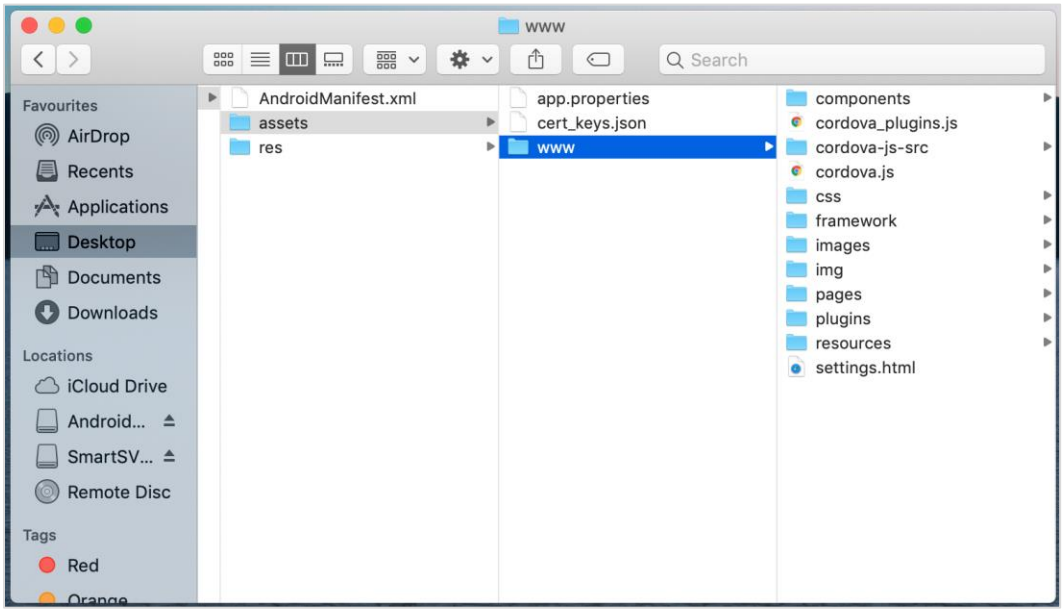
The folder structure is as shown:



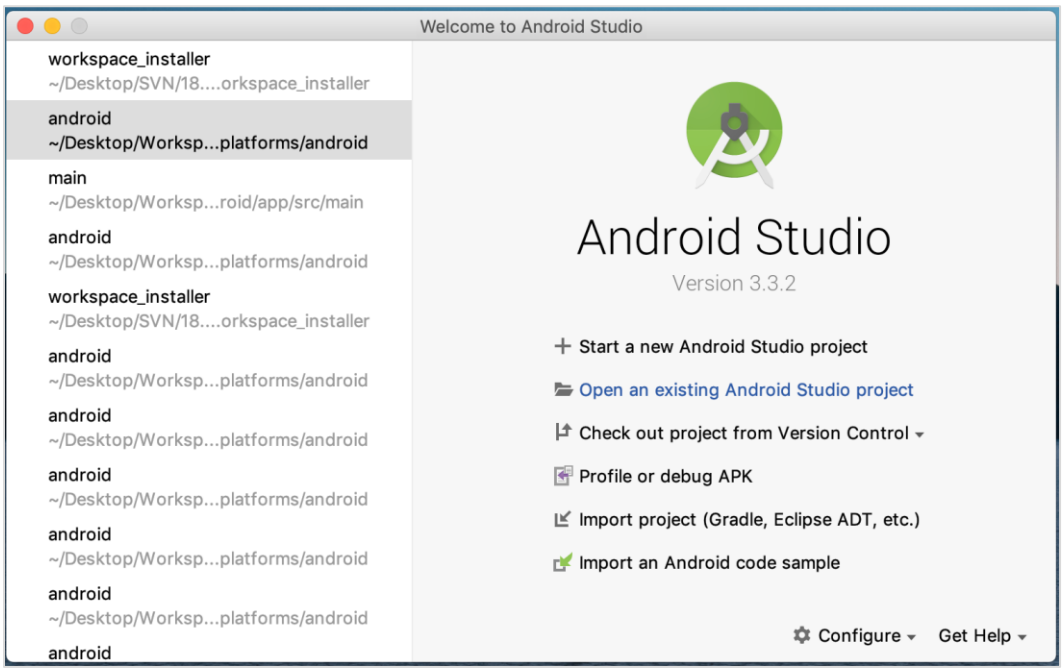
5.2 Authenticator Application Workspace Setup

1. Copy UI (Directories – components, css, framework, images, pages, resources)from /dist directory to workspace/installer/app/src/main/assets/www/

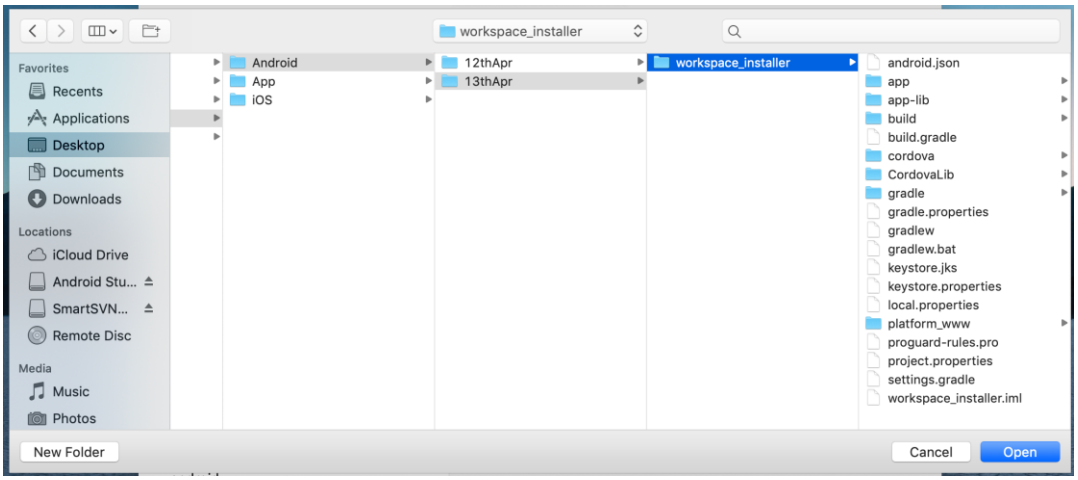
   In case any popup appears, click replace



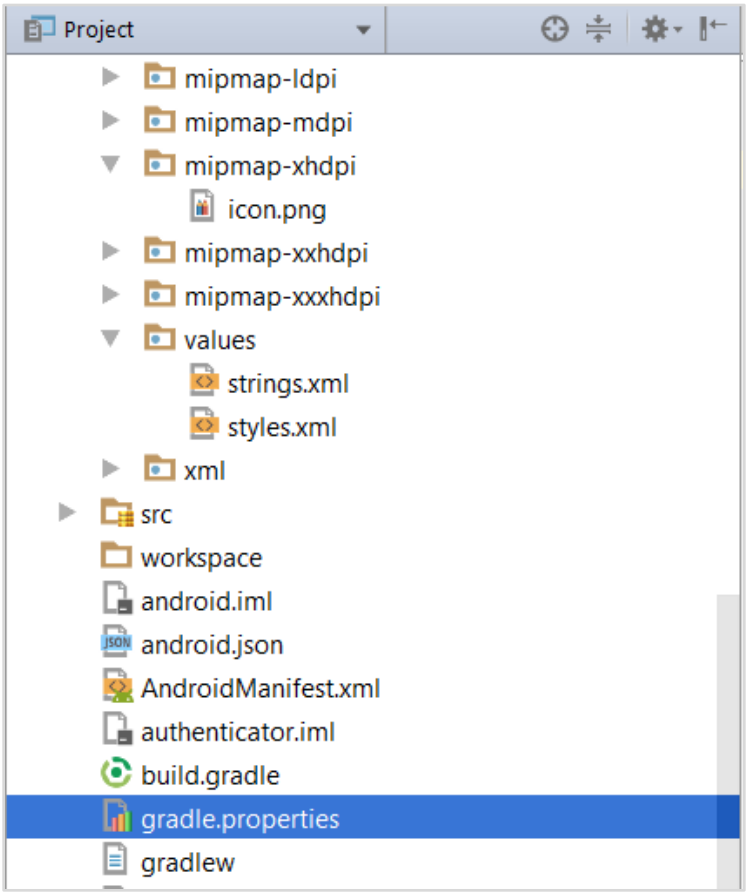
2. Launch Android Studio and open existing project



3. Open OBDX_Installer/workspace_installer folder in Android Studio.



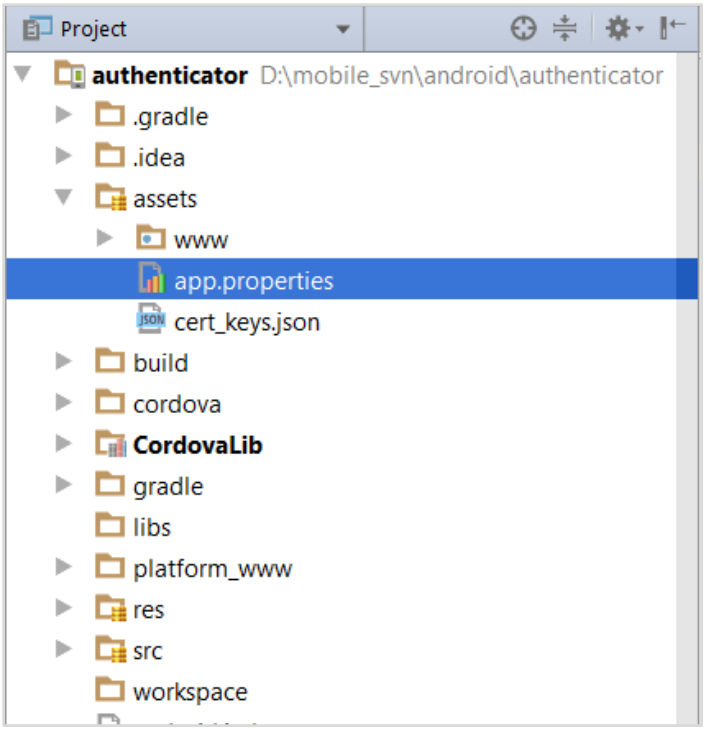
4. Open gradle.properties file and update following properties with relevant proxy address if required

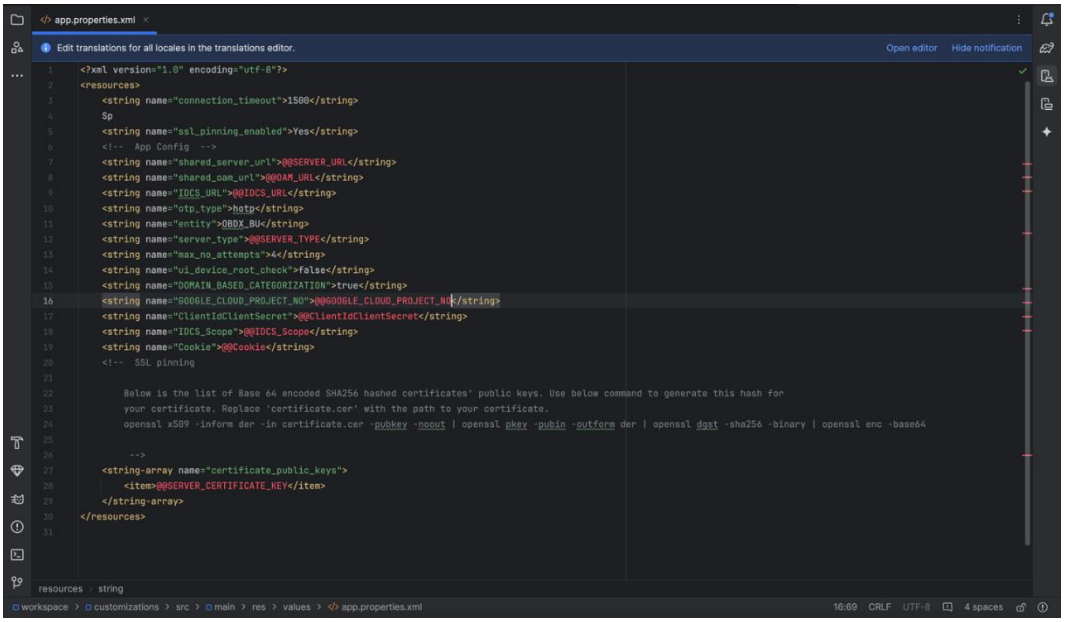


```
systemProp.http.proxyHost = <proxy_address>

systemProp.https.proxyPort = <port_number>
```

5. Open "*assets\app.properties*" file and update following properties as per requirement





Set OTP type to HOTP/TOTP as per requirement.

Set Server Type to OAUTH

Set MAX No Attempts greater than 0

Set UI Device root check to true if you want to add check on login button.

Remove "*shared_oam_url*" property.

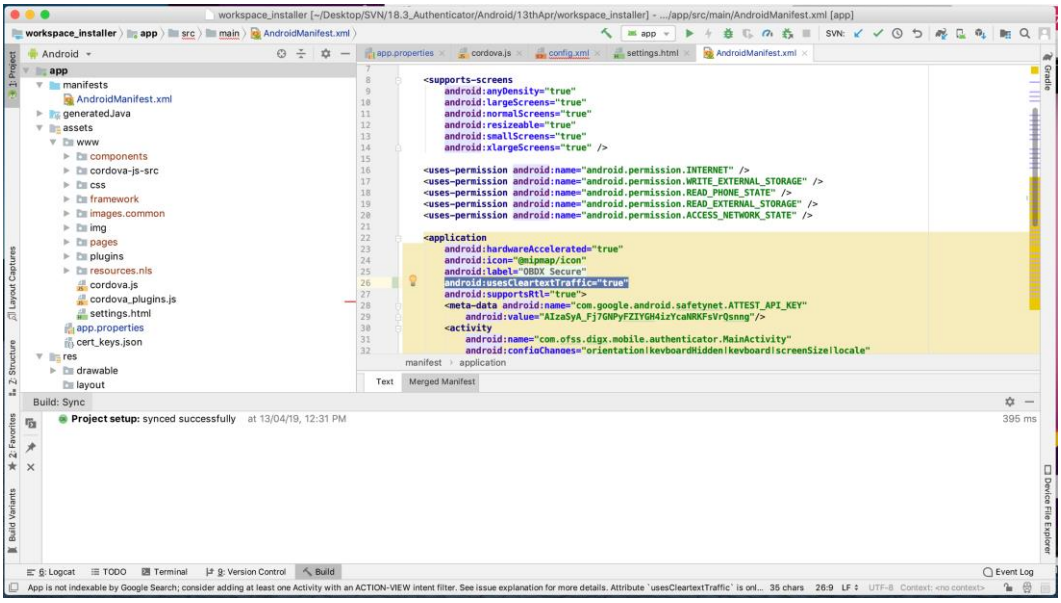
SERVER_TYPE	**OAUTH**
Shared_server_url	Cloud OHS URL where UI is hosted
IDCS_URL	IDCS URL
ClientIdClientSecret	Base64 of ClientId:ClientSecret generated from cloud console.
IDCS_Scope	OAuth_Access/consumer::**Cookie**
Cookie	The value after the :: in IDCS_Scope is the cookie

6. Click Build → Clean & Build → Rebuild project in Android Studio.
7. Click on Build → Edit Build Type → app → release

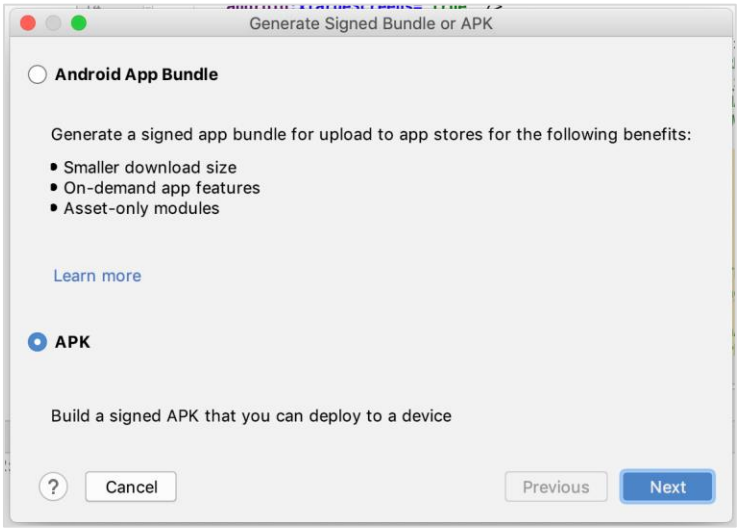
   Enable minify → true

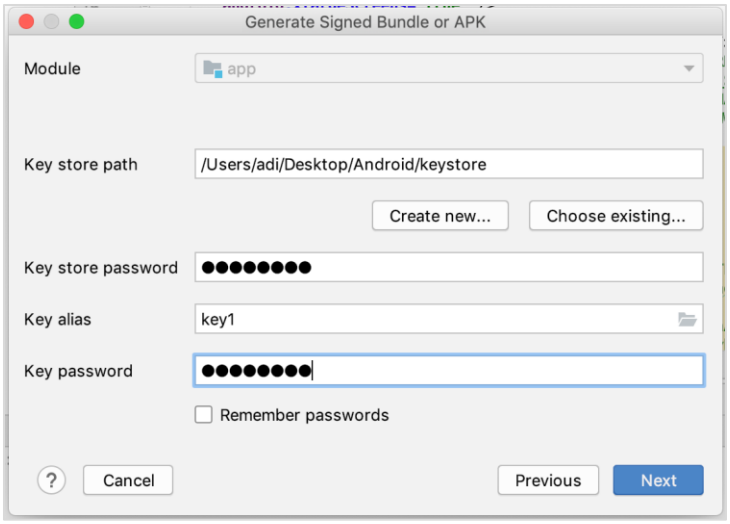
   Add progurard file from workspace_installer/proguard-rules.pro

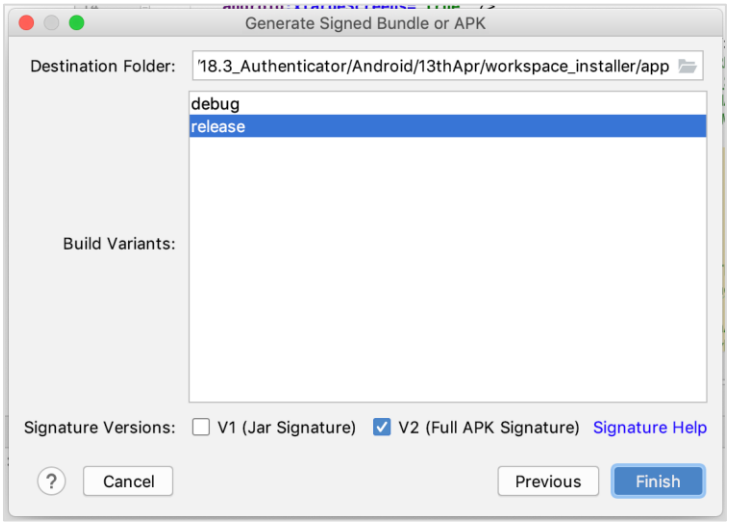
   Click OK
8. If using http protocol for development add (android:usesCleartextTraffic="true") to application tag of AndroidManifest.xml



9. **For Generating Signed Apk:** To Generate release-signed apk as follows:
10. On menu bar click on Build -> Generate Signed Apk

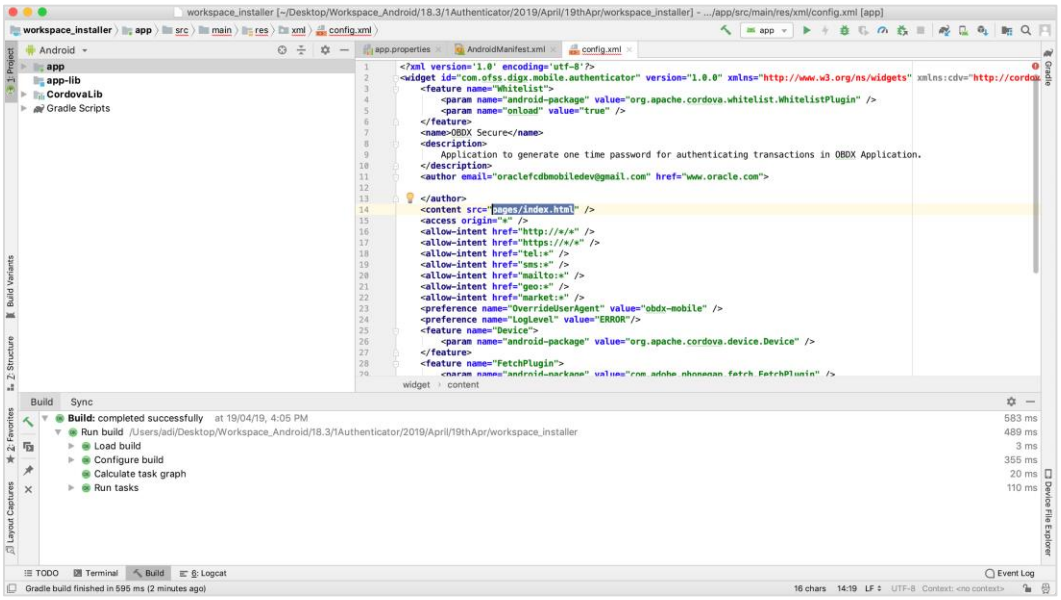



Generate Signed Bundle or APK
Module
app
Key store path
/Users/adi/Desktop/Android/keystore
Create new...
Choose existing...
Key store password
Key alias
key1
Key password
Remember passwords
Cancel
Previous
Next


Generate Signed Bundle or APK
Destination Folder:
'18.3_Authenticator/Android/13thApr/workspace_installer/app
debug
release
Build Variants:
Signature Versions:
V1 (Jar Signature)
V2 (Full APK Signature)
Signature Help
Cancel
Previous
Finish

Click Finish to generate .apk

The application has config page to add URL. This is for development purpose only and can be removed using below step. (Update content src tag)



6. Application Security Configuration

Root Check → Ensure Step 3 is completed.

1. We also have to maintain package names of Servicing and Authenticator app in the same table, i.e. **DIGX_FW_CONFIG_ALL_B** corresponding to the following keys respectively:

   ANDROID_SERVICING_PACKAGE and ANDROID_AUTHENTICATOR_PACKAGE

   An example query will be:

   insert into digx_fw_config_all_b (PROP_ID, CATEGORY_ID, PROP_VALUE, FACTORY_SHIPPED_FLAG, PROP_COMMENTS, SUMMARY_TEXT, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, LAST_UPDATED_DATE, OBJECT_STATUS, OBJECT_VERSION_NUMBER) values ('ANDROID_SERVICING_PACKAGE', 'mobileconfig', 'com.ofss.zigbank', 'N', '', 'Stores device id in OUD', 'ofssuser', sysdate, 'ofssuser', sysdate, 'Y', 1,);

SSL Pinning

2. Get the list of Base 64 encoded SHA256 hashed certificates' public keys of server's valid certificates. Use below command to generate this hash for your certificate. Replace '<certificate.der>' with the path to your certificate.

   openssl x509 -inform der -in <certificate.der> -pubkey -noout | openssl pkey -pubin -outform der | openssl dgst -sha256 -binary | openssl enc -base64

3. Add the hashed keys generated in point 6 to **zigbank\platforms\android\customizations\src\main\res\values\app.properties.xml file** in 'certificate_public_keys' array. Append this key to 'sha256/' in an <item> tag as shown below. Multiple certificate keys can be added to 'certificate_public_keys' array by adding them in <item> tags.

   Eg.:

```
<string-array name="certificate_public_keys">
    <item>sha256/5kJvNEMw0KjrCAu7eXY5HZdvyCS13BbA0VJG1RSP91w=</item>
</string-array>
```

   Eg. for multiple certificates (In case OAM/IDCS is used):

```
<string-array name="certificate_public_keys">
    <item>sha256/5kJvNEMw0KjrCAu7eXY5HZdvyCS13BbA0VJG1RSP91w=</item>

     <item>sha256/3rgsgghoqrDegekpkkgk92Fgw1w7exyYCS1okef9Oo1w=</item>
</string-array>
```

7. Adding Custom Cordova Plugin

Step 1 -

Create java folder and add yout package under app(zigbank\platforms\android\app)

Create java file under your package which will extends CordovaPlugin

Override execute method with JsonArray as a parameter

Retrive jsonobject from JsonArray and get the data which passed from js file

Example:

```
public class GetDirectionMapPlugin extends CordovaPlugin {
    @Override
    public boolean execute(String action, JSONArray args, CallbackContext callbackContext)
            throws JSONException {
        try{
            JSONObject object = args.getJSONObject(0);
            String yourKey = object.getString("your_key");
        }catch (Exception e){
            Log.e(TAG,e.getMessage());
        }
        return true;
    }
}
```

Step 2 –

Create plugin file under plugins folder of

www(zigbank\platforms\android\service\workspace\app\src\main\assets\www\plugins)

Example:

```
        cordova.define("cordova-plugin-getdirection", function(require, exports, module) {
    var exec = cordova.require('cordova/exec');
```

```
    exports.navigate = function(args, successCallback, errorCallback) {
        cordova.exec(successCallback, errorCallback, "GetDirectionMapPlugin", "direction",
            [args]);
    };
});
```

cordova-plugin-getdirection.getDirectionPlugin -> user defined id from cordova_plugin.js(zigbank\platforms\android\service\workspace\app\src\main\assets\www\cordova_plugin.js)

GetDirectionMapPlugin-> name of java plugin class

direction -> action

navigate -> this can be use in js file to this function

Step 3 –

Make entry of plugin in cordova_plugin.js(zigbank\platforms\android\service\workspace\zigbank\platforms\android\app\src\main\assets\www) as below ->

Example:

```
{
    "id": "cordova-plugin-getdirection.getDirectionPlugin", -> user defined id
    "file": "plugins/cordova-plugin-getdirection/www/mapgetdirection.js", -> path of plugin js file
    "pluginId": "cordova-plugin-getdirection",
        "clobbers": [
    "window.getDirection" -> this can be used in js file to call plugin
]
}
```

Step 4 -

Make entry of java plugin class in config.xml(zigbank\platforms\android\service\workspace\zigbank\platforms\android\app\src\main\res\xml) file of app as below -

Example:

```
<feature name="GetDirectionMapPlugin">
<param name="android-package" value="Your_Plugin_Java_Class_Path" />
</feature>
```

GetDirectionMapPlugin -> Name of java plugin class

Step 5 -

Plugin calling in js file ->

Example:

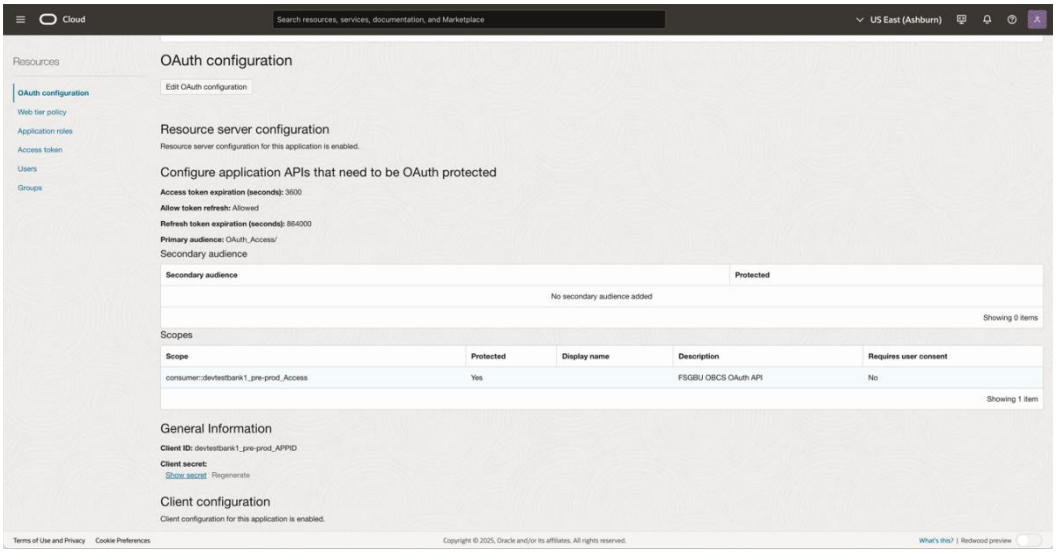
```
window.getDirection.navigate({
    originLatLng: origin,
        destinationLatLng: location
})
```

window.getDirection -> clobber define in the cordova_plugin.js file

navigate -> name of the function defined in plugin js file

8. Cloud Setup additional configurations guide

1. Set ClientIdClientSecret as Base64 of combination of ClientId:ClientSecret generated from the idcs console.



2. Set IDCS_URL as https://idcs-**IDCS_ID**.identity.pint.oc9qadev.com/oauth2/v1/token.

3. Set IDCS_Scope as "OAuth_Access/consumer::**${Cookie}**" - The value of cookie to be replaced from IDCS console.

4. Set Cookie as "${**Cookie**} - The value of cookie is the data appended after :: in IDCS_Scope.