Oracle® Banking Digital Experience

Mobile Application Builder-Android Guide

Release 25.1.0.0.0
G38599-02
October 2025

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

G38599-02

Contents

Preface

Purpose

This guide is designed to help acquaint you with the Oracle Banking application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

Before you Begin

Kindly refer to our **Getting Started User Guide** for common elements, including Symbols and Icons, Conventions Definitions, and so forth.

Pre-requisites

Specify **User ID** and **Password**, and login to **Home** screen.

Audience

This document is intended for the following audience:

- Customers
- Partners

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 customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

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.

Related Resources

For more information on any related features, refer to the following documents:

- Oracle Banking Digital Experience Installation Manuals
- Oracle Banking Digital Experience Licensing Manuals

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.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.

Acronyms and Abbreviations

The list of the acronyms and abbreviations used in this guide are as follows:

Table 1 Acronyms and Abbreviations

Abbreviation	Description
OBDX	Oracle Banking Digital Experience

Post-requisites

After finishing all the requirements, please log out from the **Home** screen.

1

OBDX Servicing Application

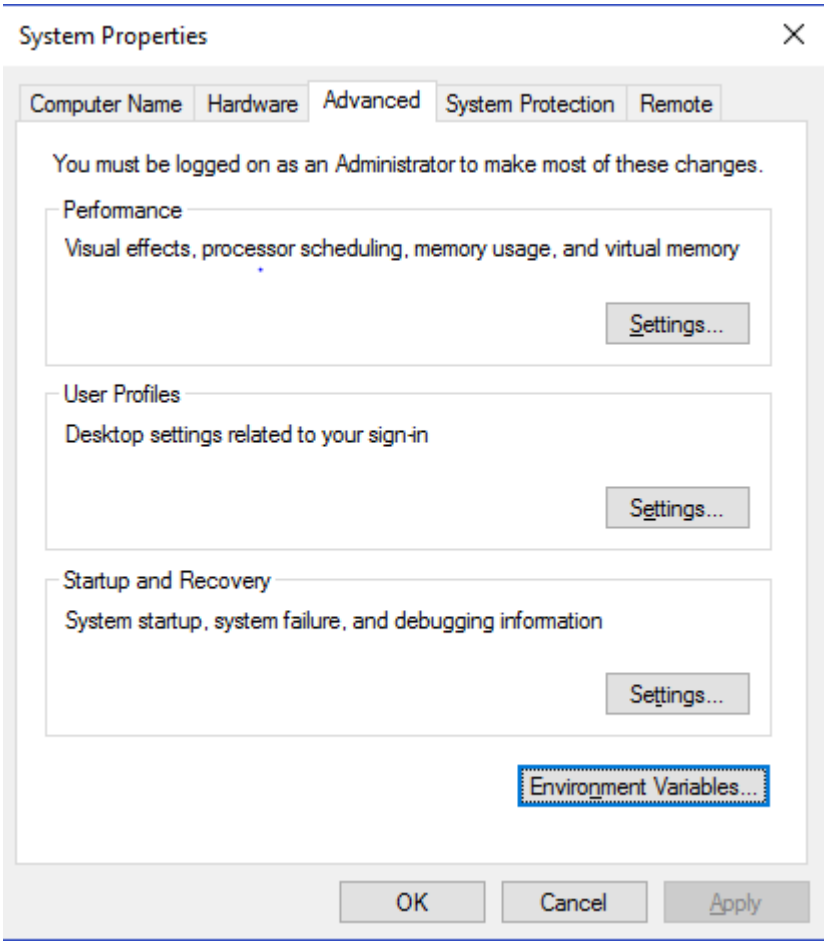
1.1 Prerequisites

This topic provides information on **Prerequisites**.

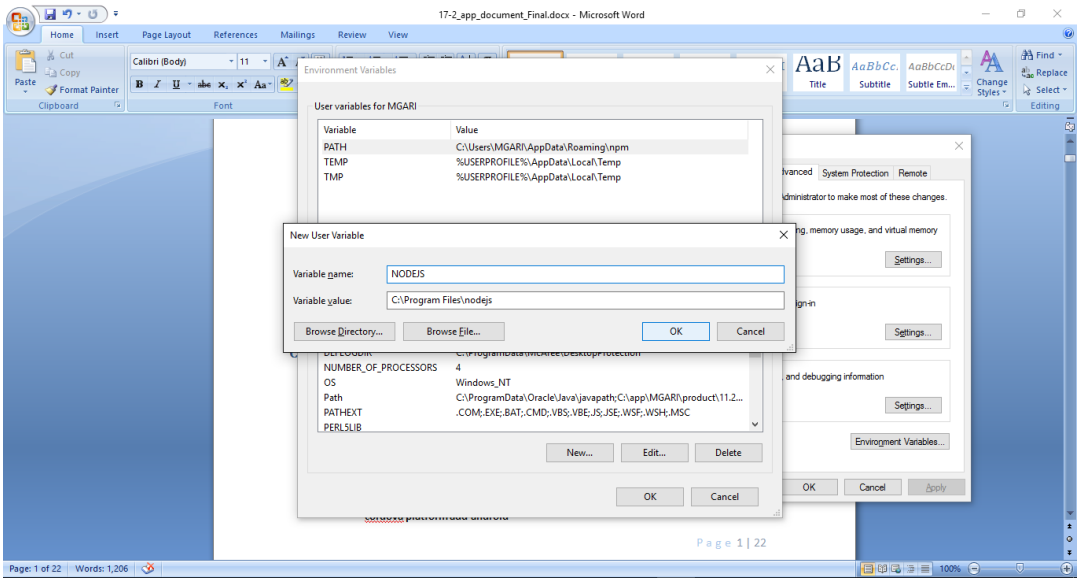
OBDX Android App is supported only on versions n (current) and n-1 release.

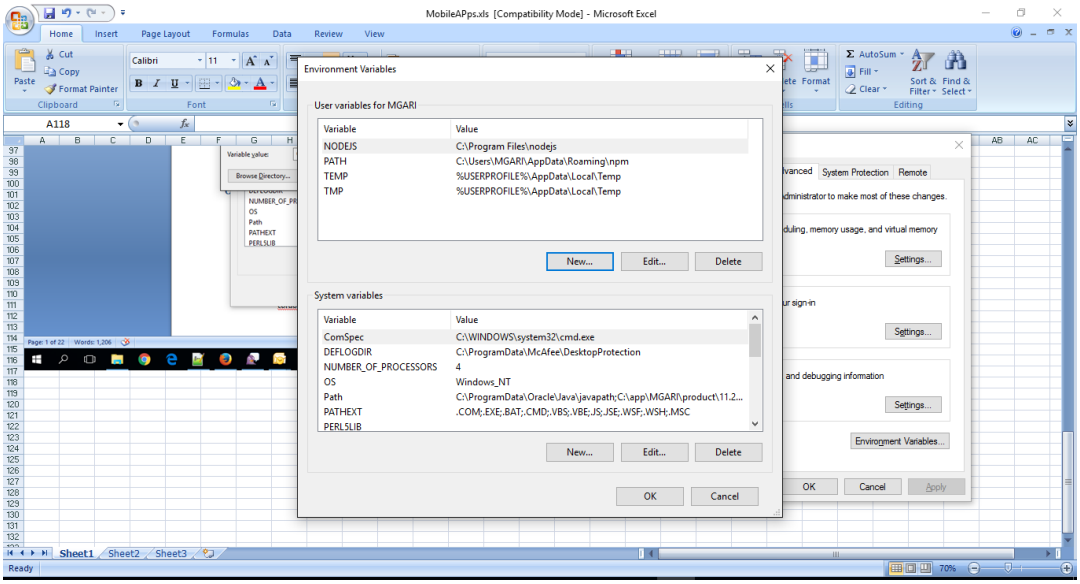
1. Download and Install node JS (will be downloaded to default path).
2. Install node js from https://nodejs.org.

3. Download and Install Android Studio.
4. Download and install Android Studio from https://developer.android.com/studio/index.html.
5. Download and Install Android platforms.
6. Update Android SDK to latest API Level.
7. Gradle Version: gradle-7.5
8. Android Gradle Plugin Version (7.4.2): 'com.android.tools.build:gradle:7.4.2' or above
9. Set Environment variables.
10. Set following system variables:
    a. Click on Windows key and type Environment Variables.
    b. A dialog box will appear. Click on the **Environment Variables** button as shown below:



11. NODEJS <nodejs_path> Example: *"C:\Program Files\nodejs\"*.
12. Add the above variables in *"PATH"* system variable.





1.2 Create project using Remote UI

This topic provides information on **Create project using Remote UI**.

1. 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 **Section1.4: Importing in Android Studio** directly.

1.3 Local UI by running on local machine or local server

This topic provides information on **Local UI by running on local machine or local server**.

Building un-built UI (required in case of customizations)

1. For this version, since the UI is built with webpack, the built UI cannot be modified from with the mobile workspace as it is minified code. Hence, either bank can hoist the UI is two ways:
   - Use local machine as local server and host the UI on local development machine and connect the application using localhost.
   - OR host the UI on local development server and point the application to that server URL
2. UI is same for internet and mobile, same build process of internet to be followed.
   Bank can follow the UI build steps from “Oracle Banking Digital Experience User Interface Guide”.
3. Additionally, building UI for mobile, Open scripts->webpack->webpack.dev.js and add below line in devServer object:
   As below

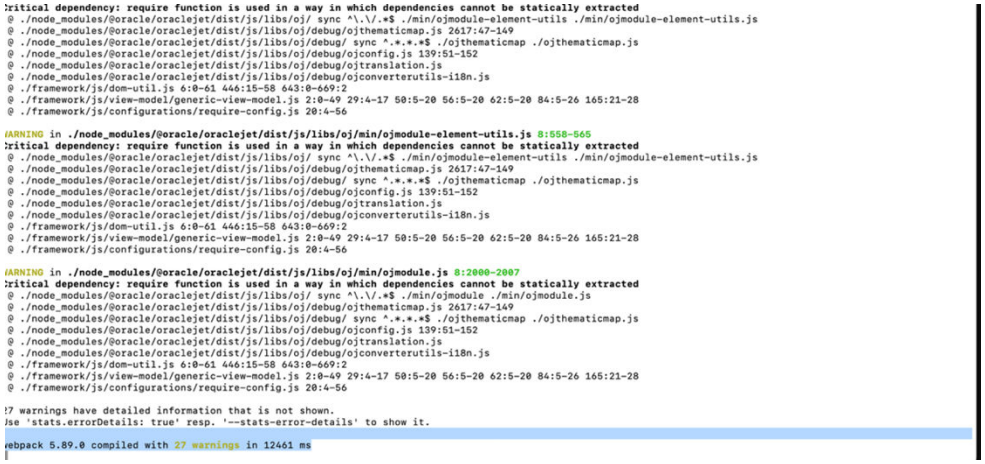
```
        headers:
{
"Access-Control-Allow-Origin":"*"
},
SAMPLE: devServer:
{
static: path.join(__dirname,"../../dist"),
compress: true,
 port: 4000,
hot: false,
client: false,
headers:
{
"Access-Control-Allow-Origin":"*"
},
```

4. Also, in webpack.dev.js comment out below lines inside “entry” key.

```
entry: {//  main:
          "framework/js/configurations/require-config.js",// Runtime code
for hot module replacement//hot:
'webpack/hot/dev-server.js',// Dev server client for web socket transport,
hot and live reloadlogic//client:
          'webpack-dev-server/client/index.js?hot=true&live-//
reload=true',
},
```

5. Once the UI is built, run below command to start a local server on the development machine using below command:
   - npm run start

   - Once this server starts, below is the window which appears. This indicates local server is started.



- Point the `key_server_url` to http://localhost:4000 and run the application on simulator. To run on device, the internet proxy should allow localhost domain to accept incoming requests.
  If it is blocked, UI should be built and “npm start” command should be executed on a development server machine which is accessible in the network. They `key_server_url` will then point to that local server URL instead of localhost

> **Note**
>
> Proper SSL & proper domain needs to be configure to run this on android as androd won’t support for http url.

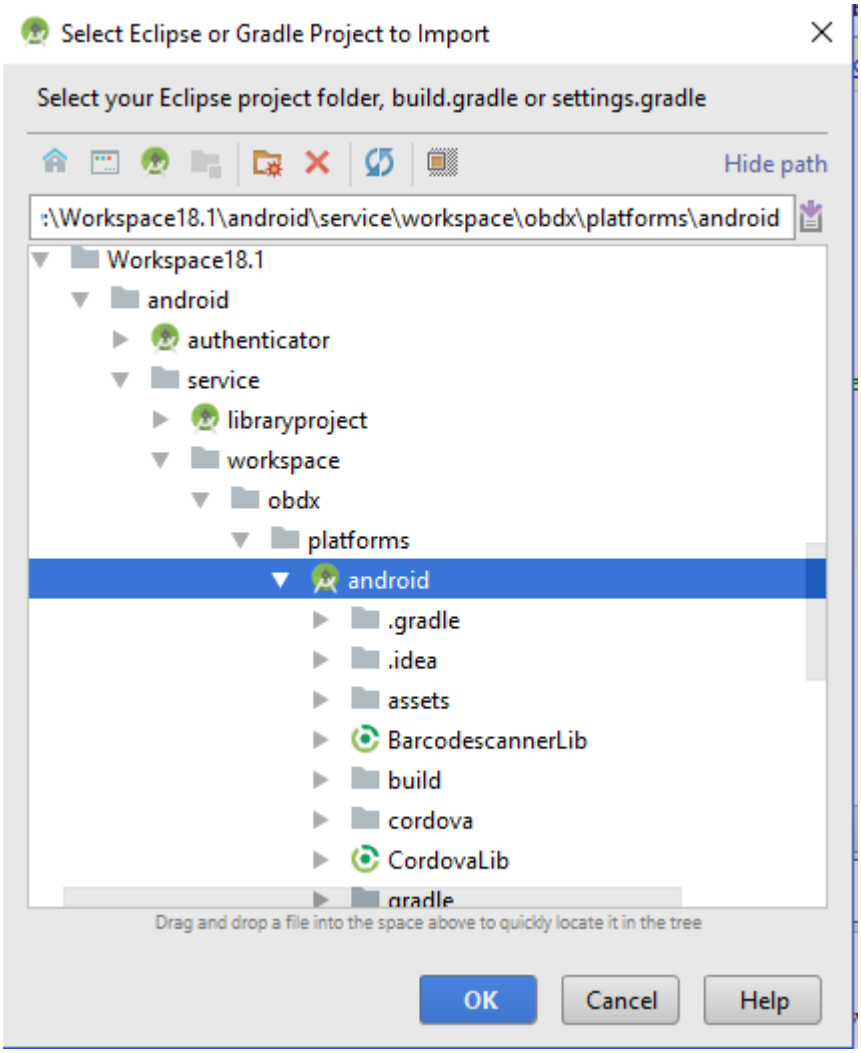
6. If banks want to debug UI the update “devtool” configuration. Refer Webpack documentation https://webpack.js.org/configuration/devtool/ for more details.

1.4 Importing in Android Studio

This topic describes the systematic instruction to **Importing in Android Studio** option.

Open Android Studio

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



1.5 Widget Functionality

This topic provides information on **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>

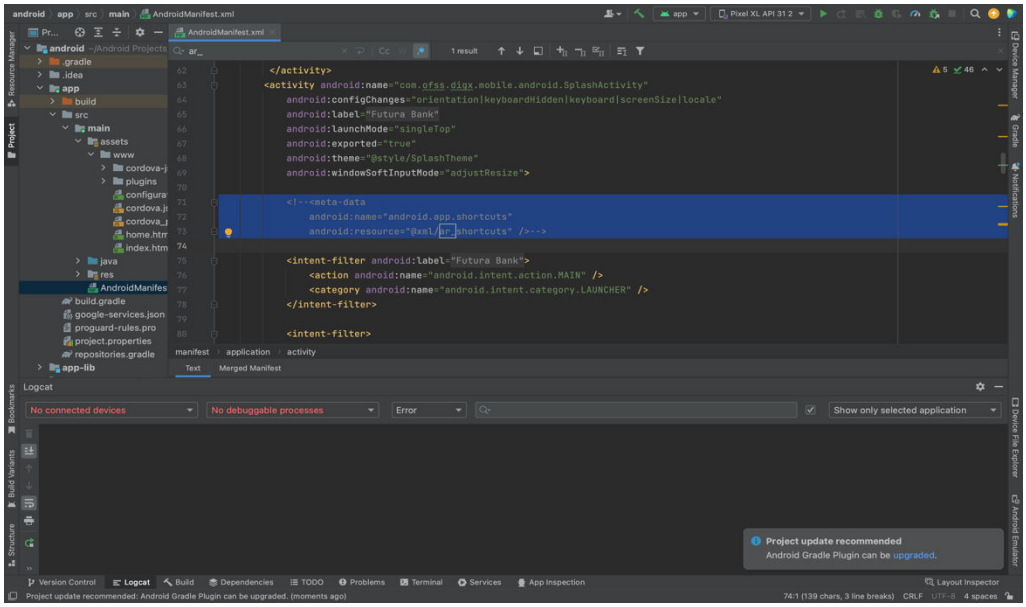
If bank does not want this feature, then they can disable this by making above flag to false.

1.6 Scan to Pay from Application Icon

This topic provides information on **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



1.7 Passkey (Passwordless login)

This topic describes the systematic instruction to **Passkey (Passwordless login)** option.

Passkeys are a safer and easier replacement for passwords. With passkeys, users can sign in to apps and websites using a biometric sensor (such as a fingerprint or facial recognition), PIN, or pattern. This provides a seamless sign-in experience, freeing your users from having to remember usernames or passwords.

Passkeys are supported only on devices that run Android 9 (API level 28) or higher

TO DISBALE THIS OPTION :

By doing this, passkey option will not be available to users withing the application. User will not be able to register for passkey and also will not be able to login using passkey. Follow below steps

1. Remove RTM access from Client Servicing → Authentication → Passkey Setup for Mobile Application/Mobile (Responsive)/Internet touch points

2. Set this flag in channel-framework-js-configurations-config.js to false
thirdPartyAPIs → passkey → required → false

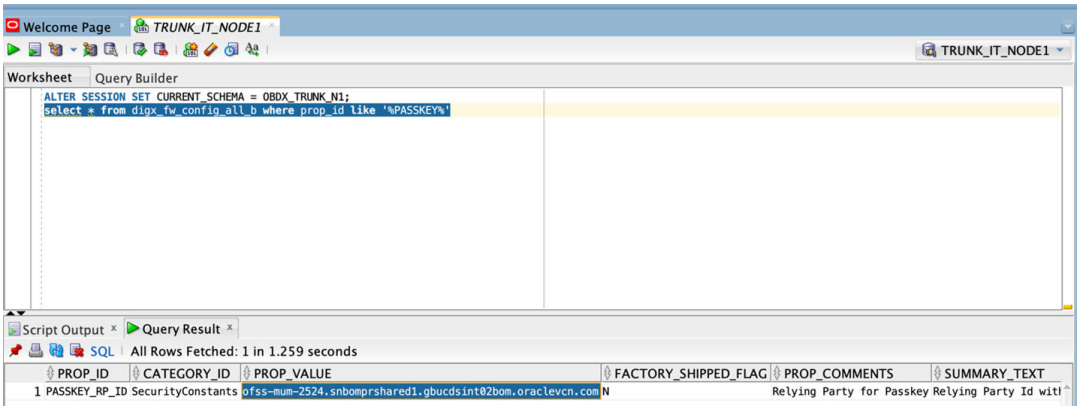
TO ENABLE THIS OPTION:

1. Add RTM access from Client Servicing → Authentication → Passkey Setup for Mobile Application,Mobile (Responsive) and Internet touch points

2. Set this flag in channel-framework-js-configurations-config.js to true
thirdPartyAPIs → passkey → required → true
3. Along with above, we need below server side and application side settup

Server-Side Setup:

1. Update the relying party in below property select prop_value from digx_fw_config_all_b where prop_id='PASSKEY_RP_ID'



2.

> **Note**
>
> Relying partId is the domain name if the website to which credentials will be associated. (Eg google.com, example.com etc)

Relying party origin is the relying party of website prefixed with protocol without the port.

(Example: https://google.com, https://example.com)

a. Create assetlinks file (assetlinks.json) -
A Digital Asset Links JSON file must be published on your website to indicate the Android apps that are associated with the website and verify the app's URL intents.

The following example assetlinks.json file grants link-opening rights to a com.example Android app:

```
[{
"relation":
        [
"delegate_permission/common.handle_all_urls"],
"target":
{
```

```
"namespace":
        "android_app",
        "package_name":
"com.example",
"sha256_cert_fingerprints":
["14:6D:E9:83:C5:73:06:50:D8:EE:B9:95:2F:34:
FC:64:16:A0:83:42:E6:1D:BE:A8:8A:04:96:B2:3F:CF:44:E5"
]
}
}]
```

The JSON file uses the following fields to identify associated apps:

package_name: The application ID declared in the app's build.gradle file.

sha256_cert_fingerprints: The SHA256 fingerprints of your app’s signing certificate. You can use the following command to generate the fingerprint via the Java keytool:

keytool -list -v -keystore my-release-key.keystore

b. Publish assestlinks.json file-
This file needs to be on https server with valid SSL certificate

You must publish your JSON verification file at the following location:

https://domain.name/.well-known/assetlinks.json

For example, if your sign-in domain is signin.example.com, host the JSON file at https://signin.example.com/.well-known/assetlinks.json.

Verify your assetlink json on below statement list tester-

https://developers.google.com/digital-asset-links/tools/generator

The MIME type for the Digital Assets Link file needs to be JSON. Make sure the server sends a Content-Type: application/json header in the response.

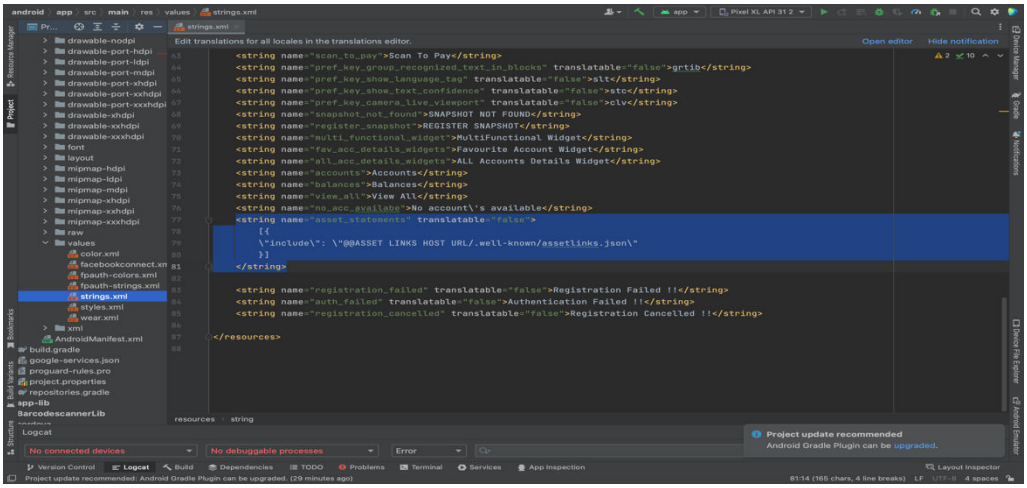
Need to change host and port in Obdx.conf as,

ProxyPass "/.well-known" "http://100.76.157.55:7003/digx-sms/v1/.well-known"

ProxyPassReverse "/.well-known" "http://100.76.157.55:7003/digx-sms/v1/.well-known"

After the setup is done, this file must be accessible on mobile browser with this url. There should not by any redirects for accessing this file.

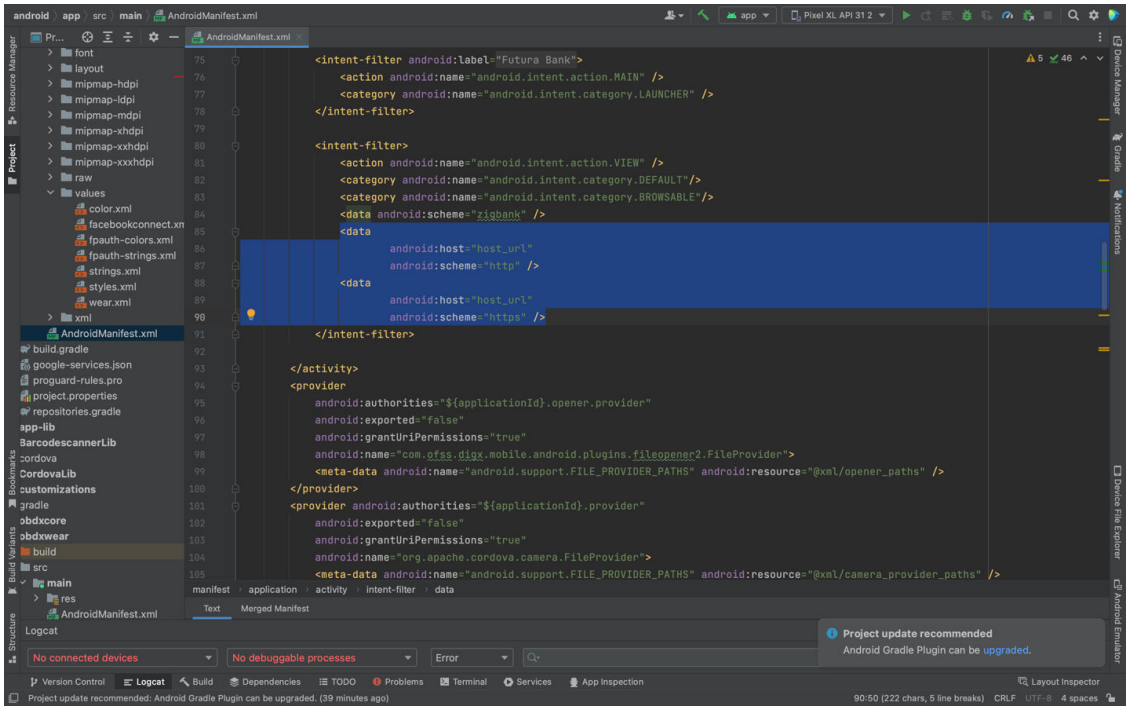
c. Add assetlinks.json file host in app’s strings.xml file.



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

This topic describes the systematic instruction to **Deeplinking - To open reset password, claim money links with the application** option.

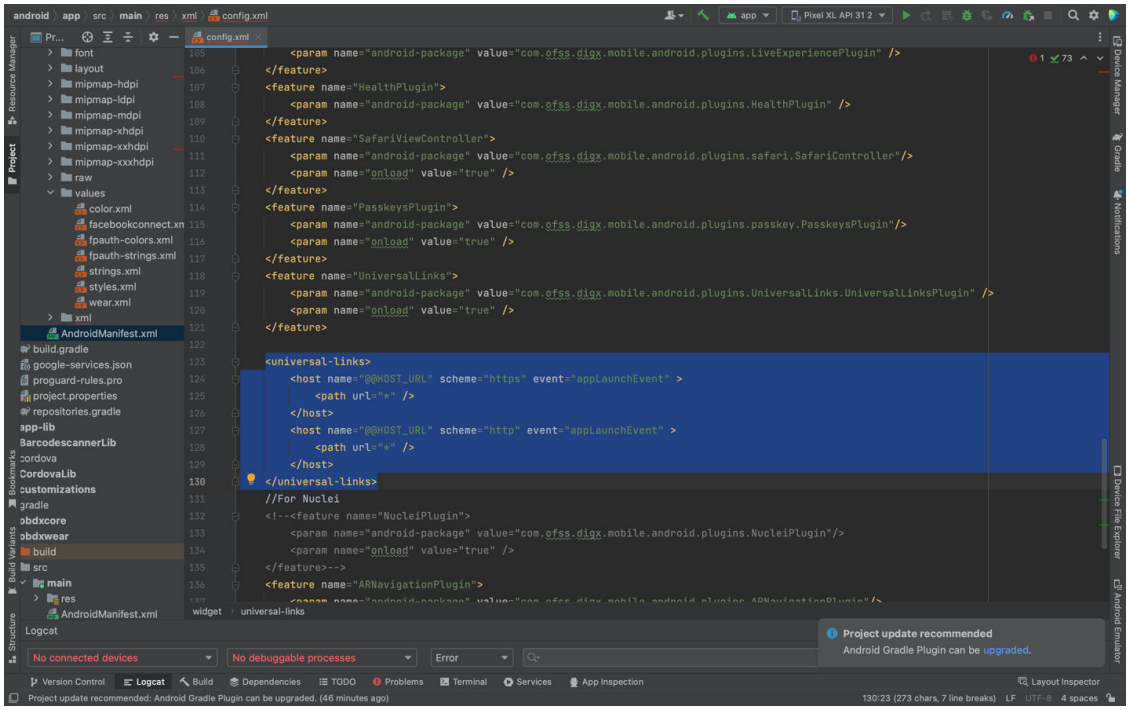
Add host url under data tag in app's AndroidManifest.xml as,



> **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,



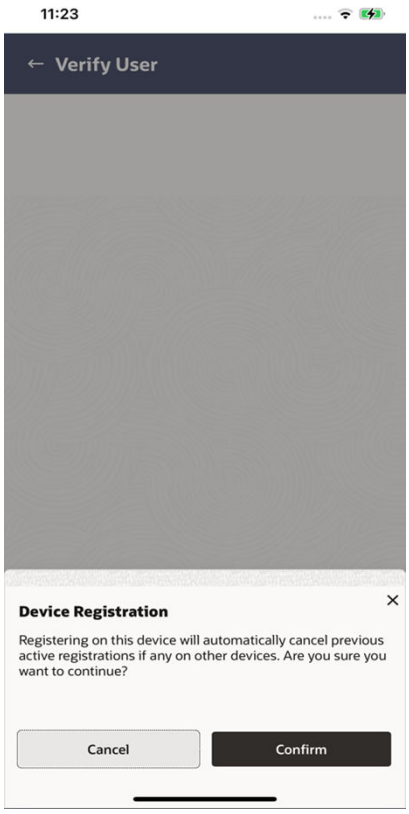
1.9 Device Registration and Push Registration Functionality

This topic provides information on **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 anyvalue 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 device to be registered for push.*
- *-1 will only multiple devices to be registered for push.*

1.10 Location Tracking Metrics

This topic provides information on **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.

1.11 Displaying Rate Option to Redirect to Playstore Page

This topic provides information on **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.

1.12 Enabling Force Update

This topic provides information on **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.

1.13 Splash Screen Migration

This topic provides information on **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**
>
> 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

1.14 App Update Manager

This topic provides information on **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.

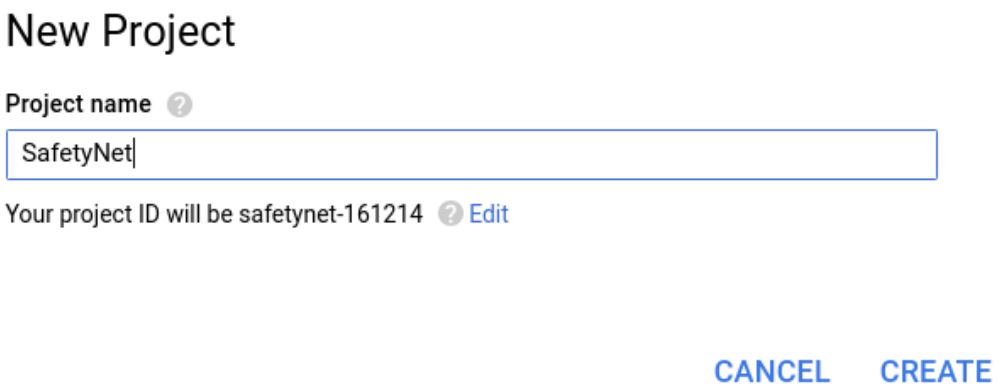
https://developer.android.com/guide/playcore/in-app-updates/test

2

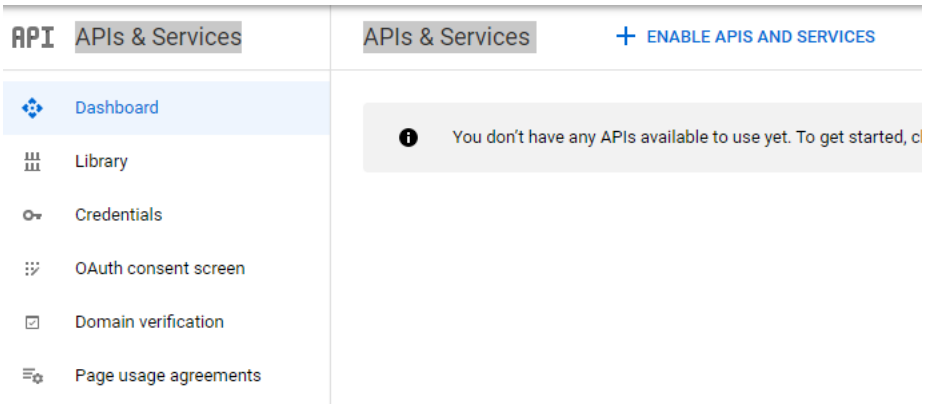
Google Play Integrity

This topic describes the systematic instruction to **Google Play Integrity** option.

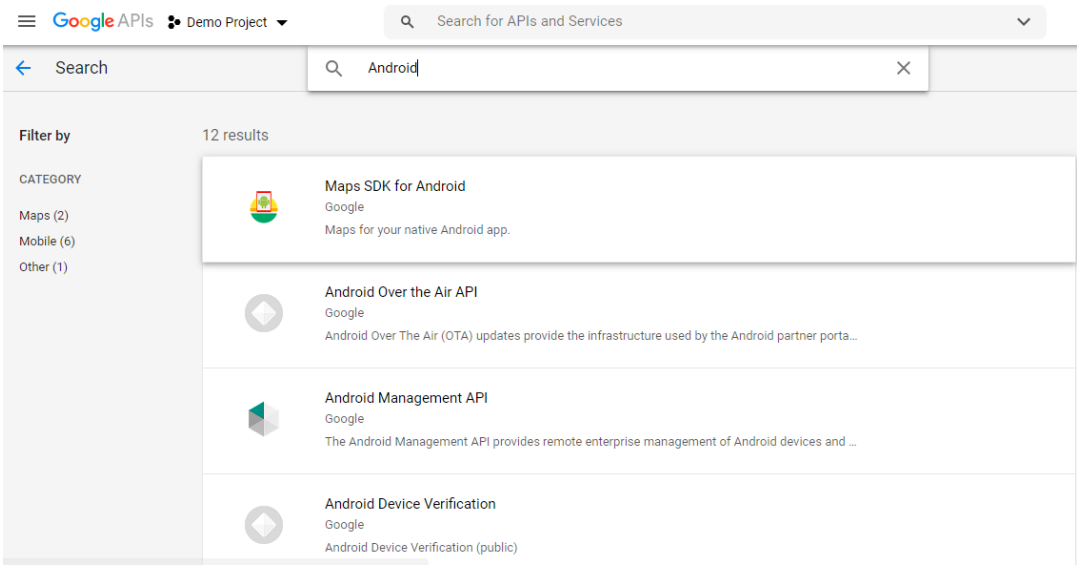
1. Go to URL https://console.developers.google.com/
2. Create a new Project and set name of you project.

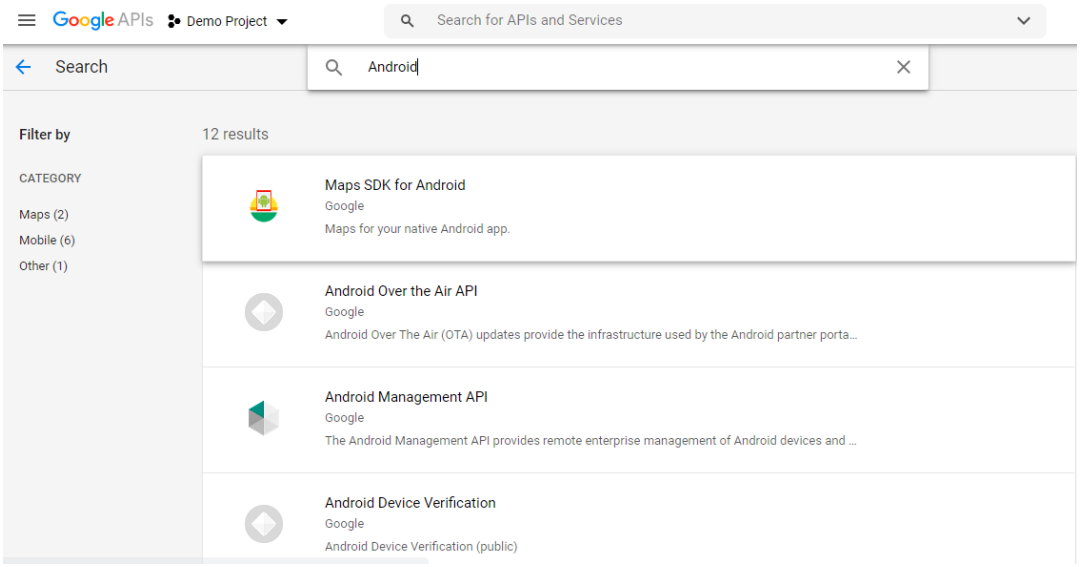


3. Choose**API's & Services** option from side bar.
4. In API's & Services → Dashboard → Choose **Enable APIS AND SERVICES**.

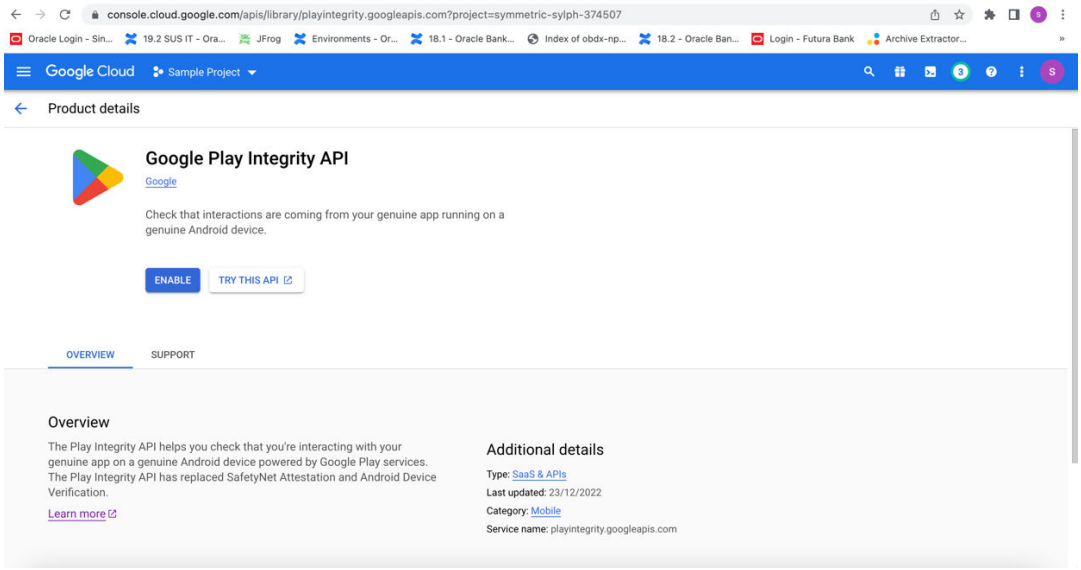


5. This will redirect to **Library** where we need to search **Google Play Integrity API.**.

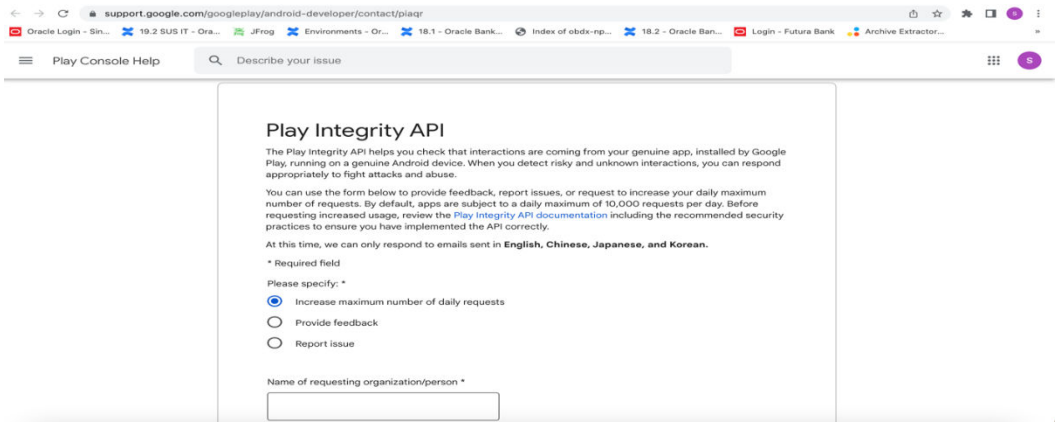


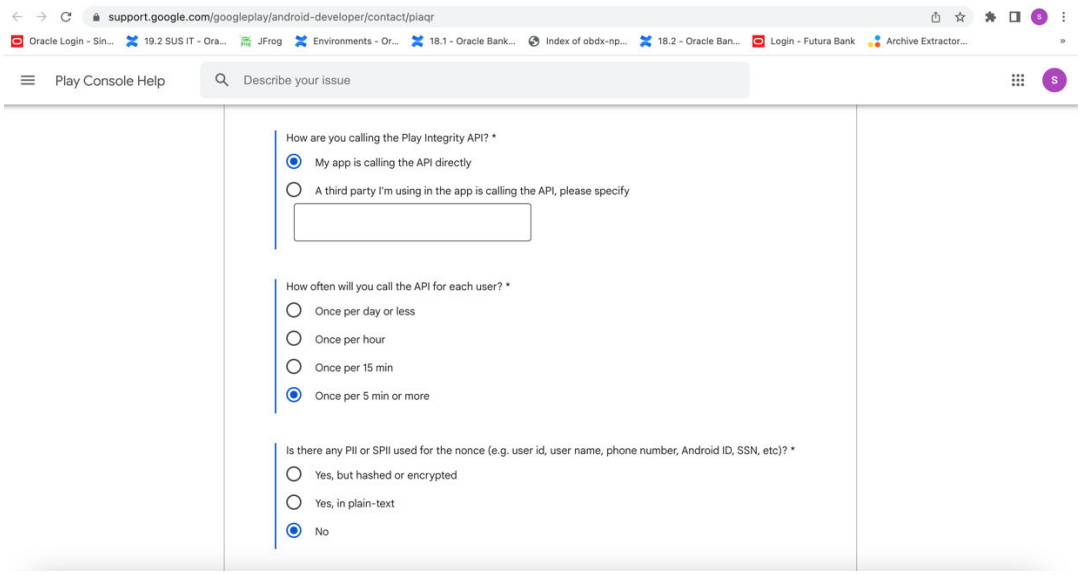


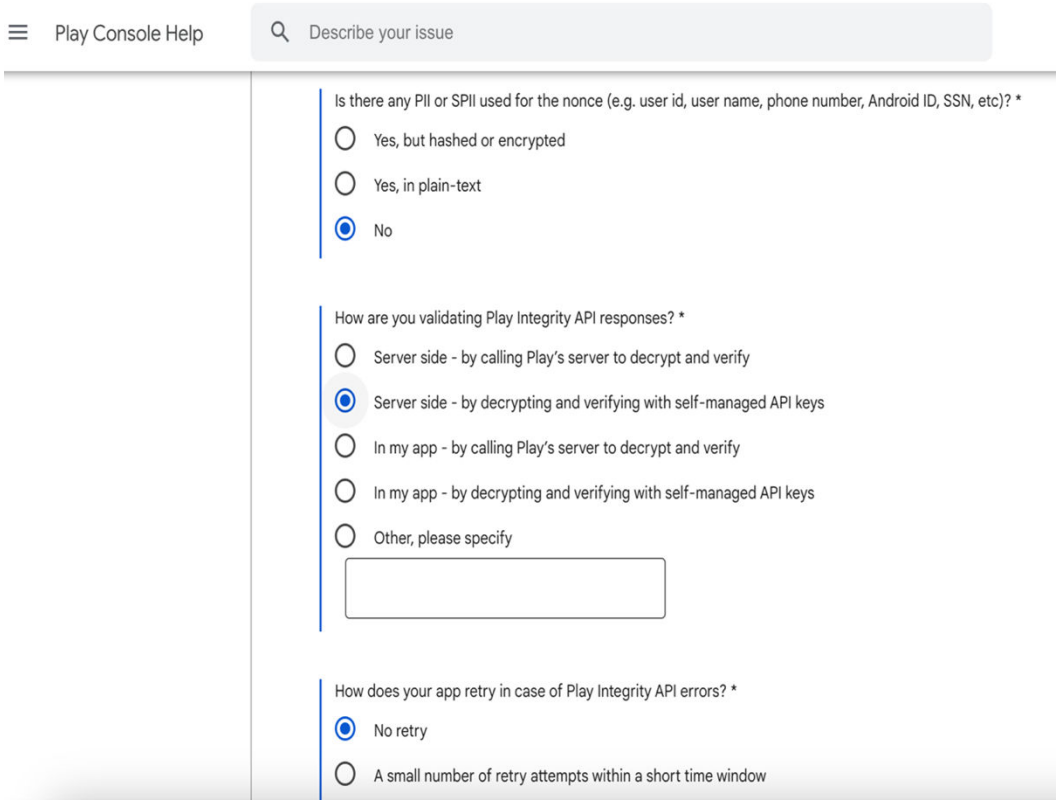
6. Click on Google Play Integrity API and enable it.

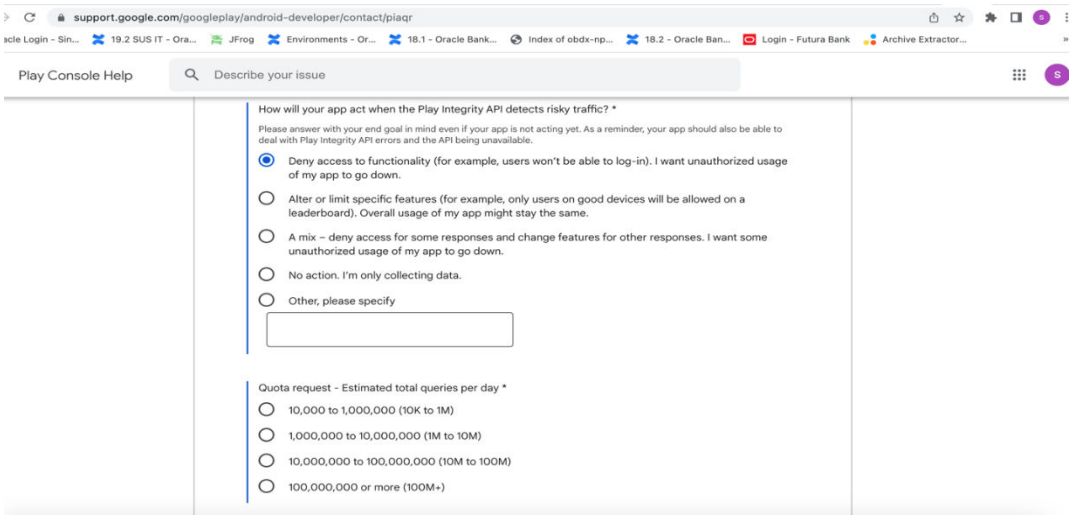


7. If the application usage is high, the quota request form needs to be submitted. Fill quota request form from below site. Also select below options.
https://support.google.com/googleplay/android-developer/contact/piaqr



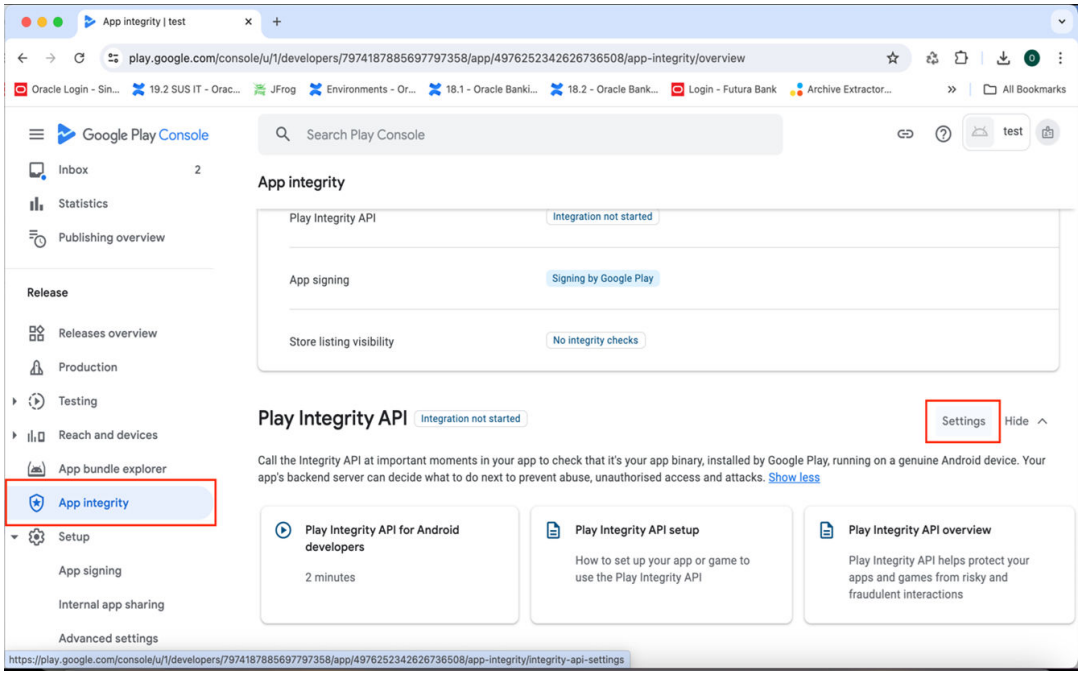




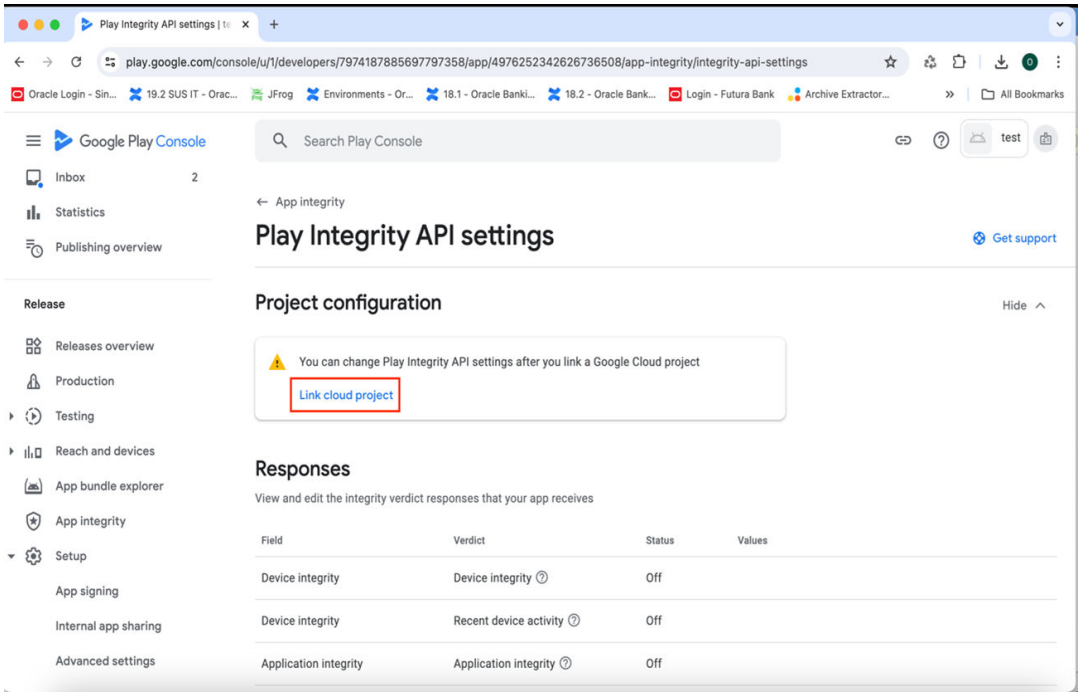


8. Quota request - Estimated total queries per day * → The approximate load, Play Integrity API is called once each time the app in opened
   Quota request - Estimated peak queries per second → Leave blank

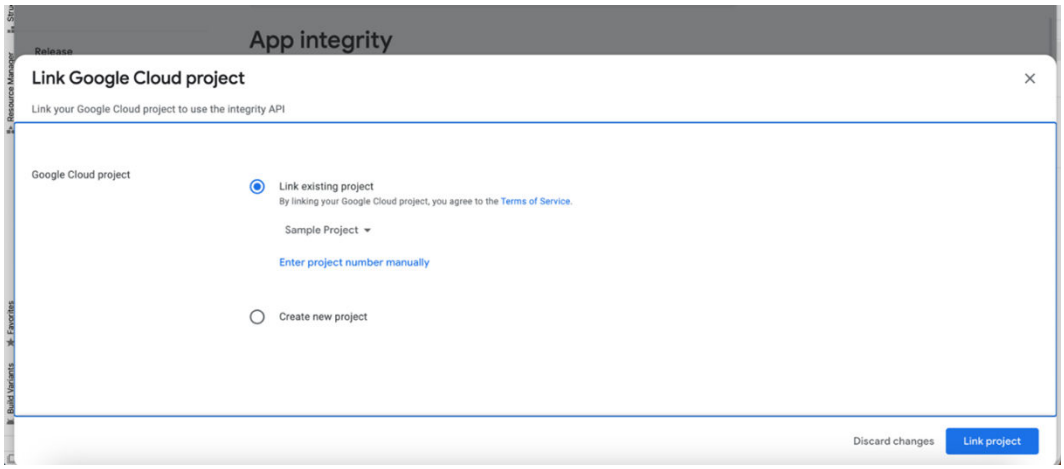
9. To enable Play Integrity responses follow below steps:
   Go to Google Play Console → Side Menu → App Integrity



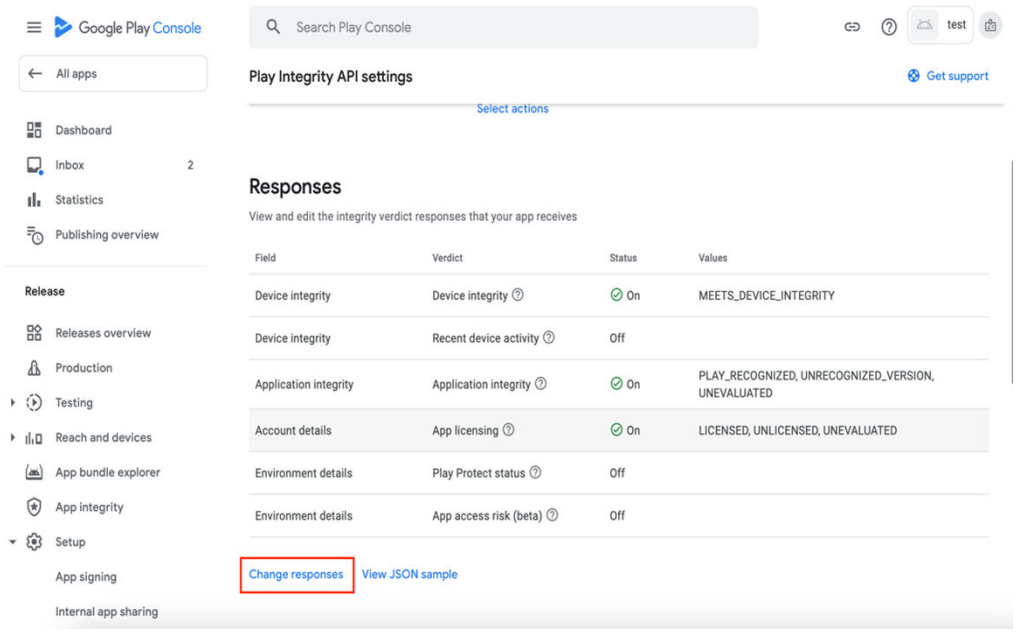
Click on **Settings**.



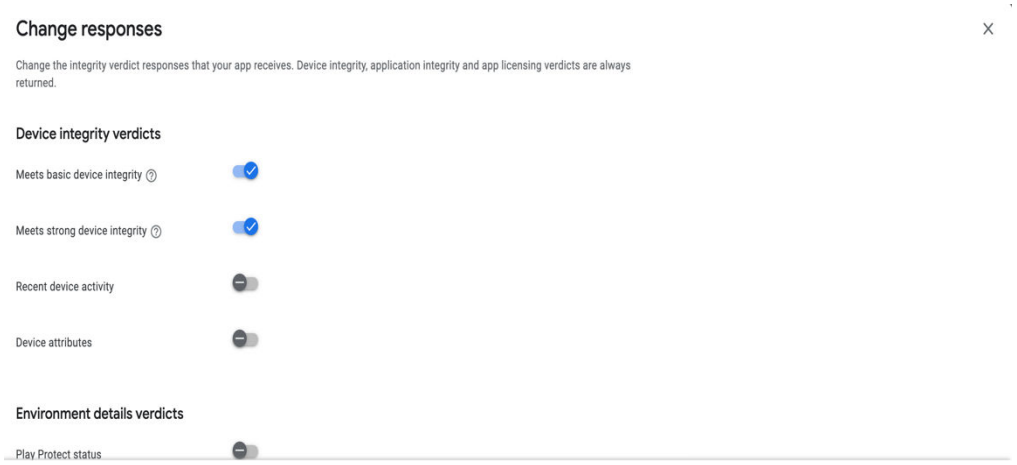
Click on **Link project** and then link your existing google cloud project. If it is not created then create new and link the same.



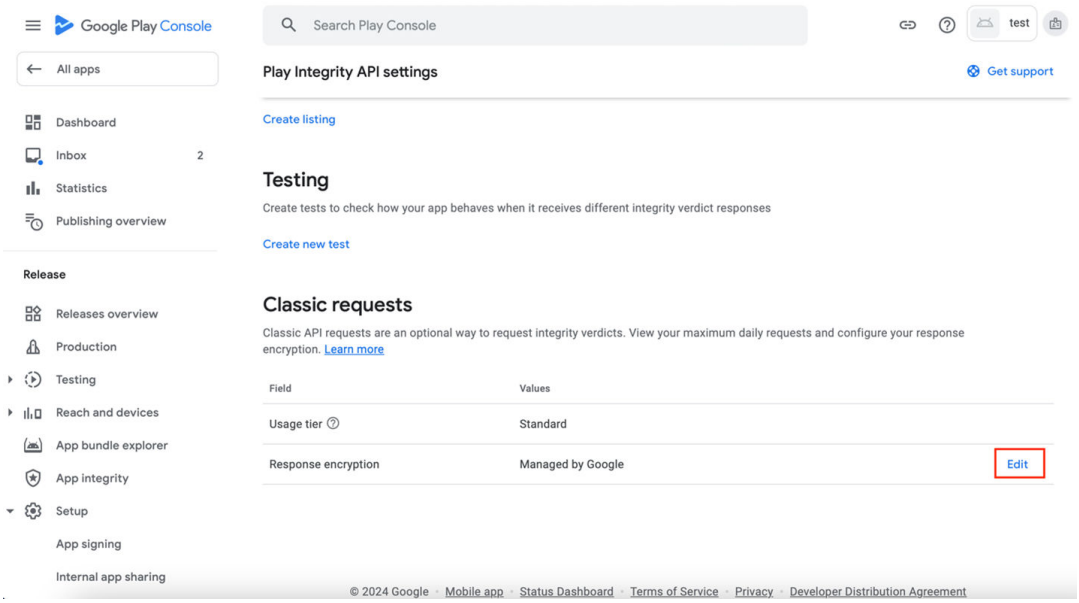
10. Scroll down on the same screen and click on **Change Responses**.



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



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



13. 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';
```

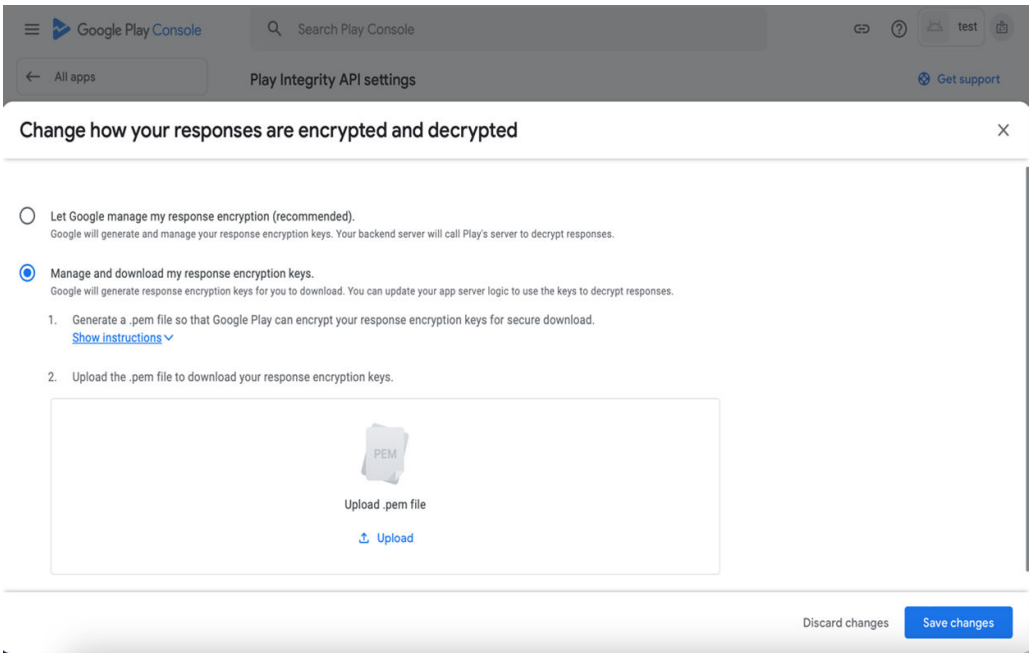
c. Similarly, Obtain the same keys for authenticator app by using above steps and then maintain those in **`DIGX_FW_CONFIG_ALL_B`** table corresponding to the following keys respectively:

`PLAY_INTEGRITY_ENCRYPTION_KEY_AUTHENTICATOR`

and `PLAY_INTEGRITY_DECRYPTION_KEY_AUTHENTICATOR`

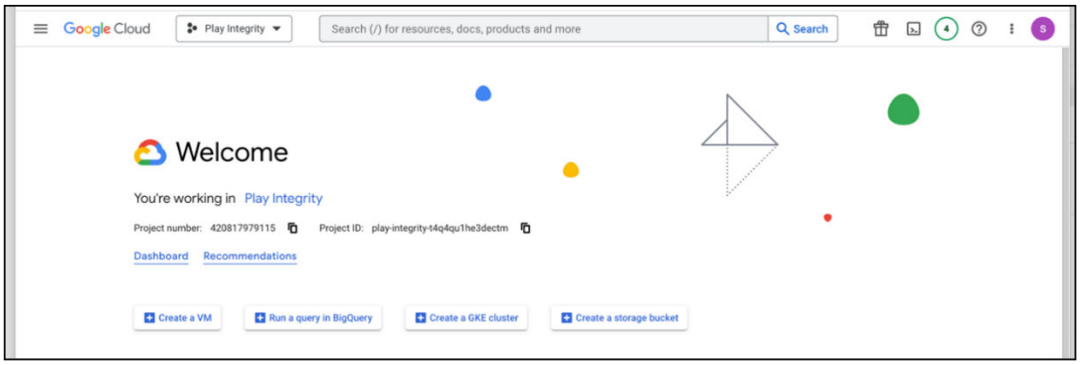
An example query will be:

```
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';
```



14. 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



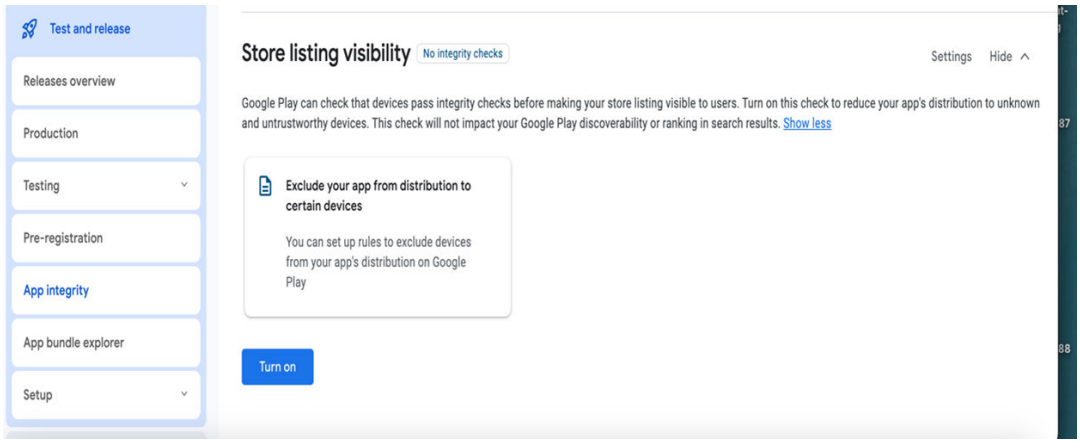
15. Mention the time in seconds to which app can hit the play integrity api. By default it is 300seconds but you can configure as per the requirement.
Use below property in RootCheckFlags.java(workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/) long playIntegrityAPICallTime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

    long playIntegrityAPICallTime = your_time_in_seconds;

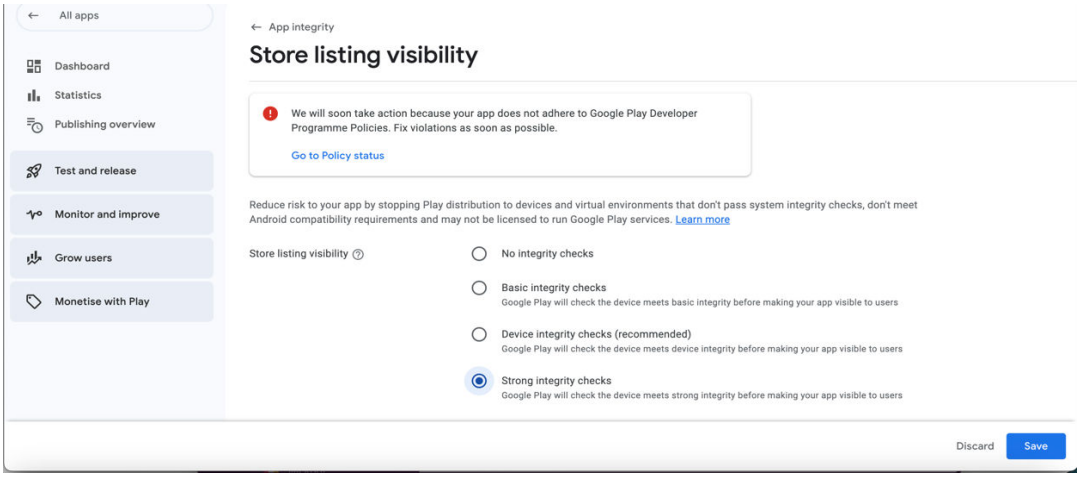
16. Scroll down on the **App Integrity**page.

    Navigate to Store listing visibility.

    Click on **Settings** button.



Select **Strong Integrity checks** option and **Save**.



Note

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

3

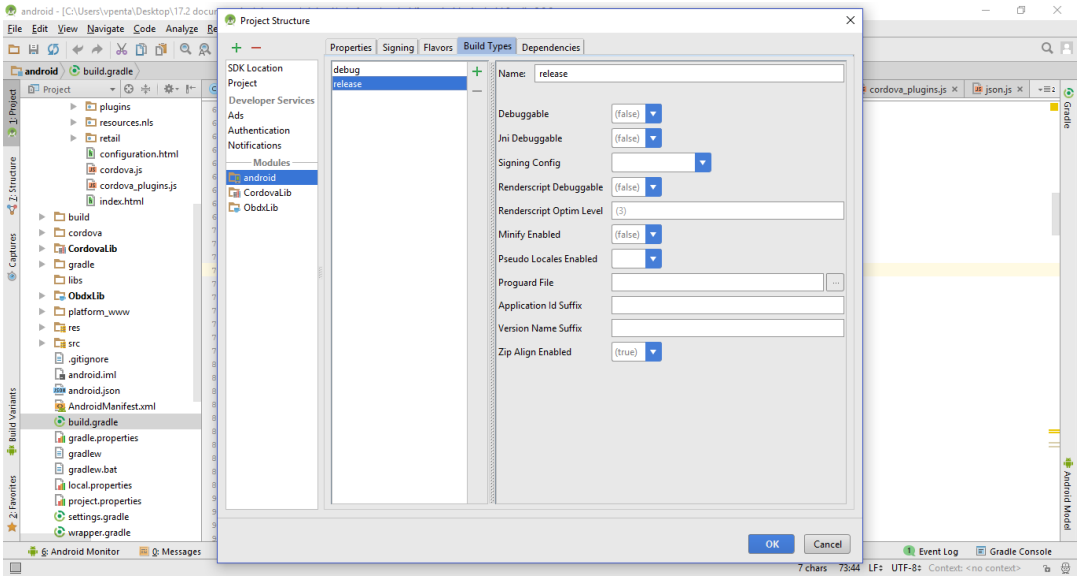
FCM Push Notifications

4

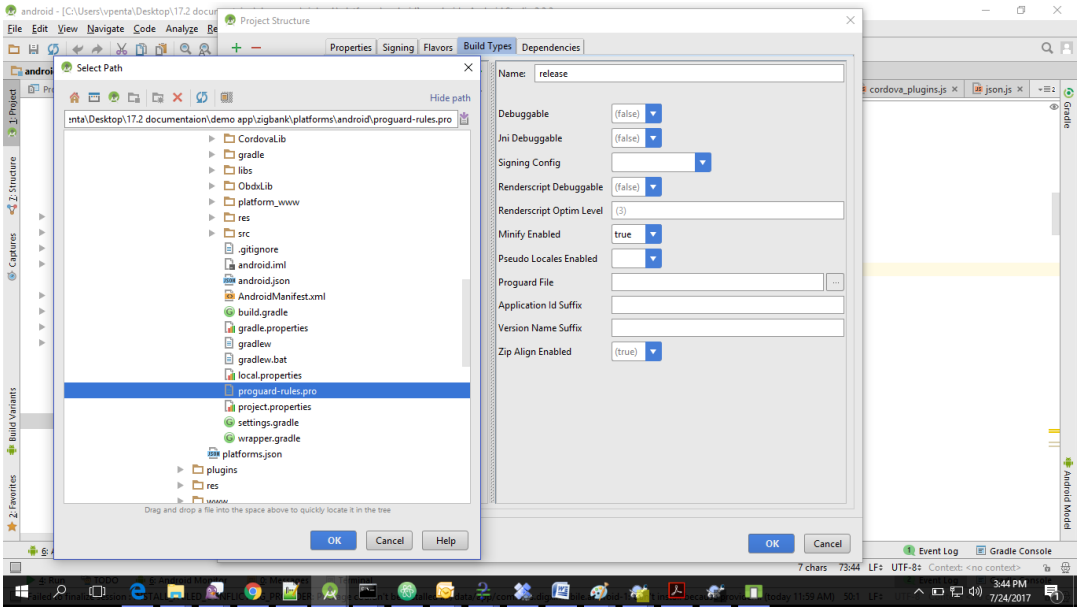
Build Release Artifacts

This topic describes the systematic instruction to **Build Release Artifacts** option.

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 **OK** → again click **OK**.
5. Adding URLs to app.properties.xml (customizations/src/main/res/values/)
   a. NONOAM (DB Authenticator setup)
   For more information on fields, refer to the field description table.

Table 4-1 NONOAM (DB Authenticator setup)

SERVER_TYPE	NONOAM
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:1844
WEB_URL	Eg. https://mumaa012.in.oracle.com:1844
SERVER_CERTIFICATE_KEY	Refer steps 6.7

   b. OBDXTOKEN (Token based mechanism)
   For more information on fields, refer to the field description table.

Table 4-2 OBDXTOKEN (Token based mechanism)

SERVER_TYPE	NONOAM
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:1844 (This URL must be of OHS without webgate)
WEB_URL	Eg. https://mumaa012.in.oracle.com:1844
SERVER_CERTIFICATE_KEY	Refer point 6.7

   c. OAM Setup (Refer to installer pre requisite documents for OAuth configurations)
   For more information on fields, refer to the field description table.

Table 4-3 OAM Setup

SERVER_TYPE	OAM
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:1844 (This URL must be of OHS without webgate)
WEB_URL	Eg. https://mumaa012.in.oracle.com:1844
KEY_OAUTH_PROVIDER_URL	http://mum00aon.in.oracle.com:14100/
APP_CLIENT_ID	<Base64 of clientid:secret> of Mobile App client
APP_DOMAIN	OBDXMobileAppDomain
WATCH_CLIENT_ID	<Base64 of clientid:secret> of wearables

Table 4-3 (Cont.) OAM Setup

SERVER_TYPE	OAM
WATCH_DOMAIN	OBDXWearDomain
SNAPSHOT_CLIENT_ID	<Base64 of clientid:secret> of snapshot
SNAPSHOT_DOMAIN	OBDXSnapshotDomain
LOGIN_SCOPE	OBDXMobileAppResServer.OBDXLoginScope
SERVER_CERTIFICATE_KEY	Refer steps 6.7

6. 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}
   ```

7. IDCS Setup
   For more information on fields, refer to the field description table.

Table 4-4 IDCS Setup

SERVER_TYPE	IDCS
KEY_SERVER_URL	Eg. https://mumaa012.in.oracle.com:18443/ (This URL must be of OHS without webgate)
WEB_URL	Eg. https://mumaa012.in.oracle.com:18443/
KEY_OAUTH_PROVIDER_URL	http://obdx-tenant01.identity.c9dev0.oc9qadev.com/oauth2/v1/token
APP_CLIENT_ID	<Base64 of clientid:secret> of Mobile App client
WATCH_CLIENT_ID	<Base64 of clientid:secret> of wearables
SNAPSHOT_CLIENT_ID	<Base64 of clientid:secret> of snapshot
LOGIN_SCOPE	obdxLoginScope
OFFLINE_SCOPE	urn:opc:idm:__myscopes__ offline_access
SERVER_CERTIFICATE_KEY	Refer steps 6,7

8. 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

9. 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.

10. 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>
    ```

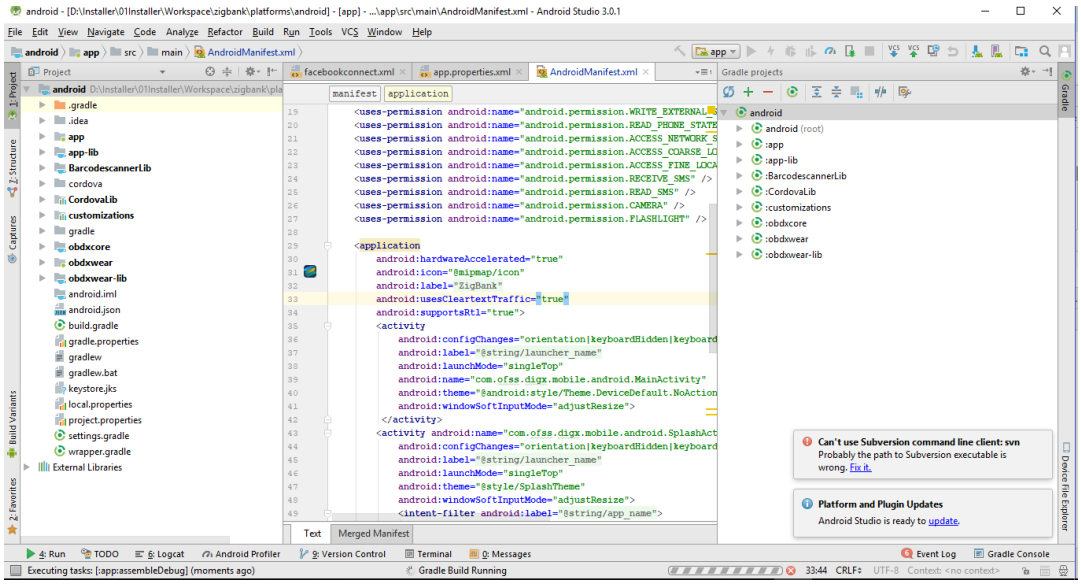
    If you are using local UI then enable below flag in config.js(platforms/android/app/src/main/assets/www/framework/js/configurations/config.js) file -

    ```
    domainDeployment:
    {
    enabled: true
    }
    ```

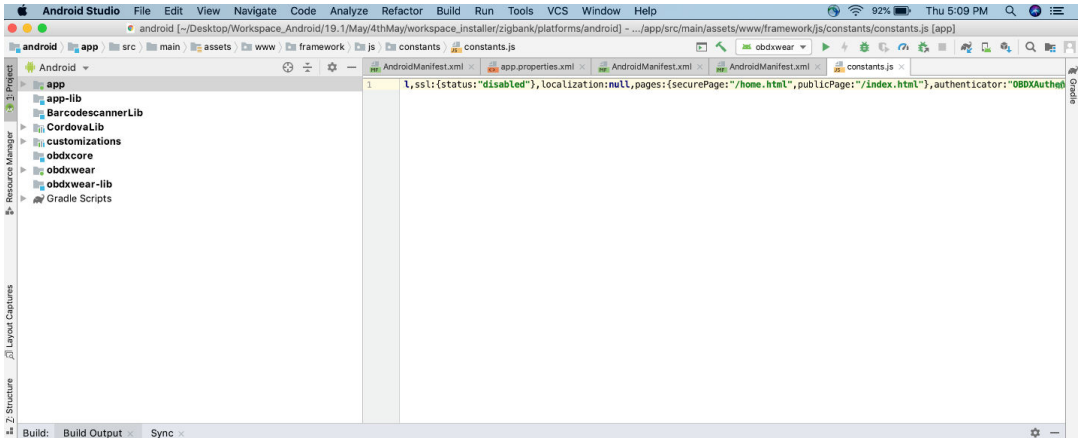
11. Adding chatbot support to mobile application (Optional).

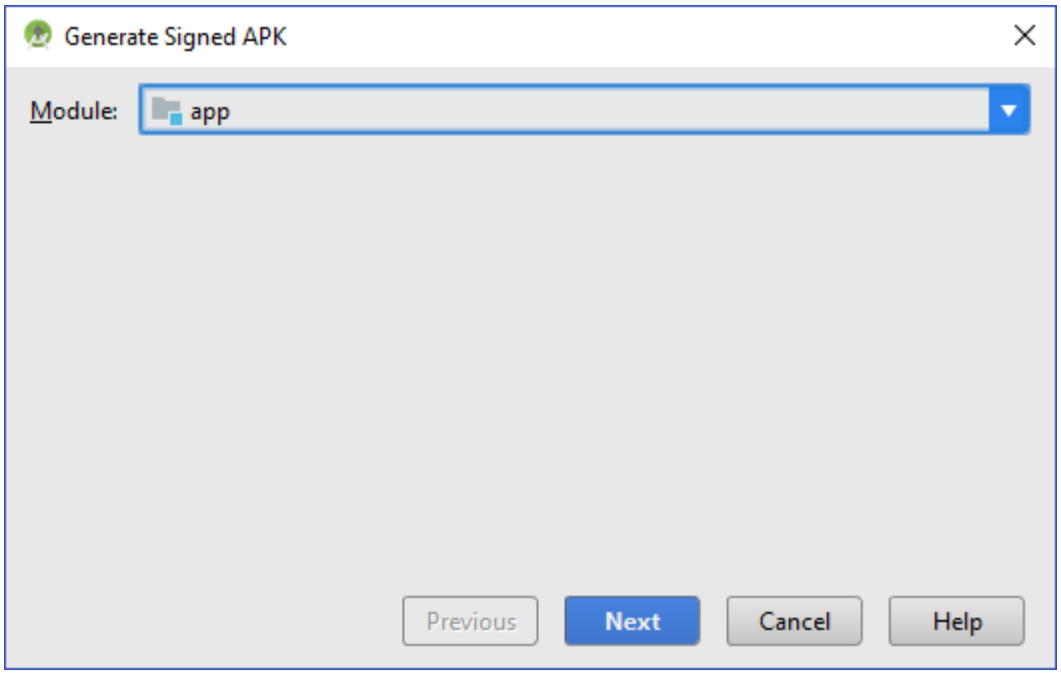
CHATBOT_ID	The tenant ID
CHATBOT_URL	The URL for the ChatApp application in ODA

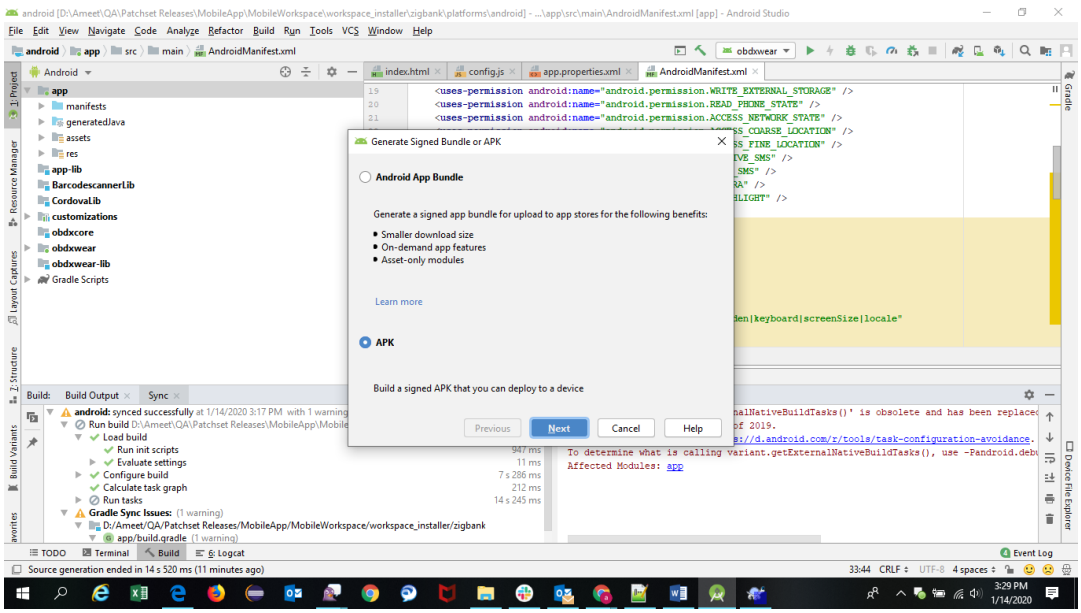
12. If using http protocol for development add (android:usesCleartextTraffic="true") to application tag of AndroidManifest.xml (on app & obdxwear target).



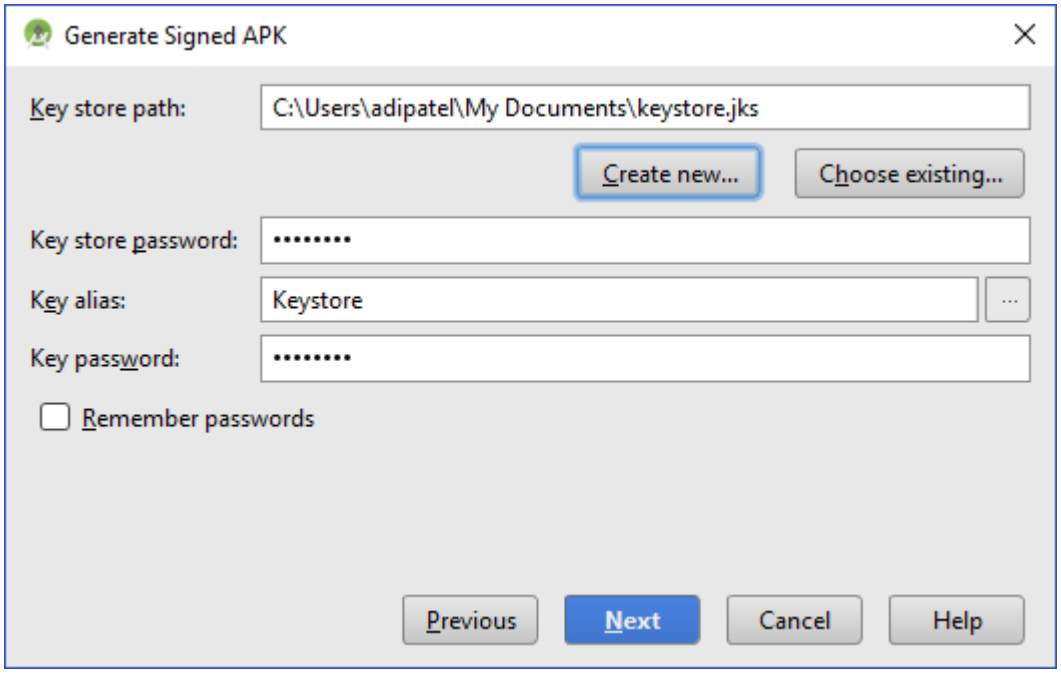
13. **For Generating Signed Apk:** To Generate release-signed apk as follows:
    On menu bar click on Build → Generate Signed Apk

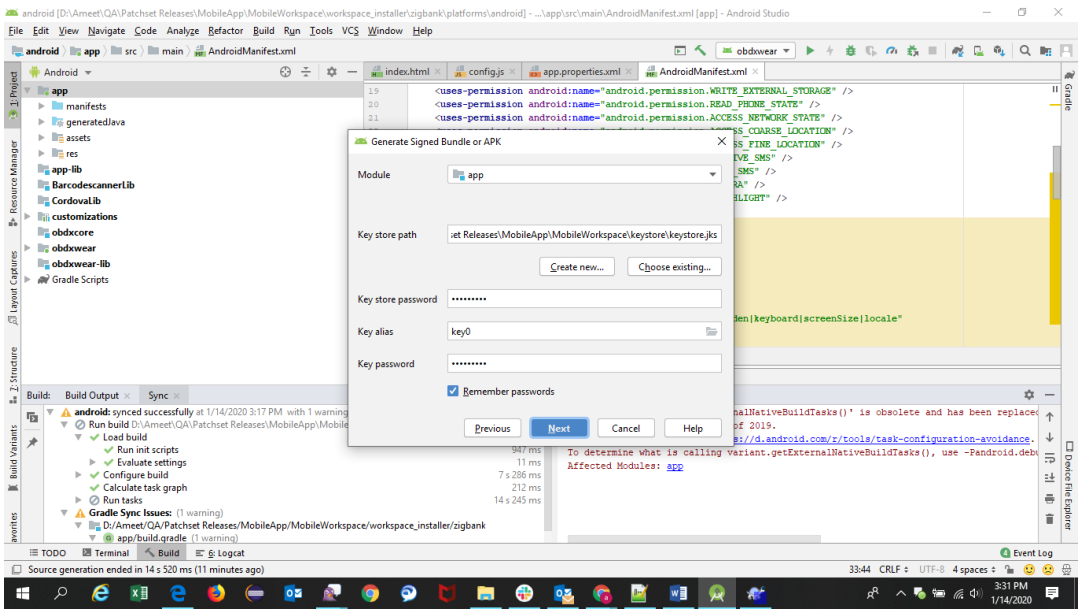




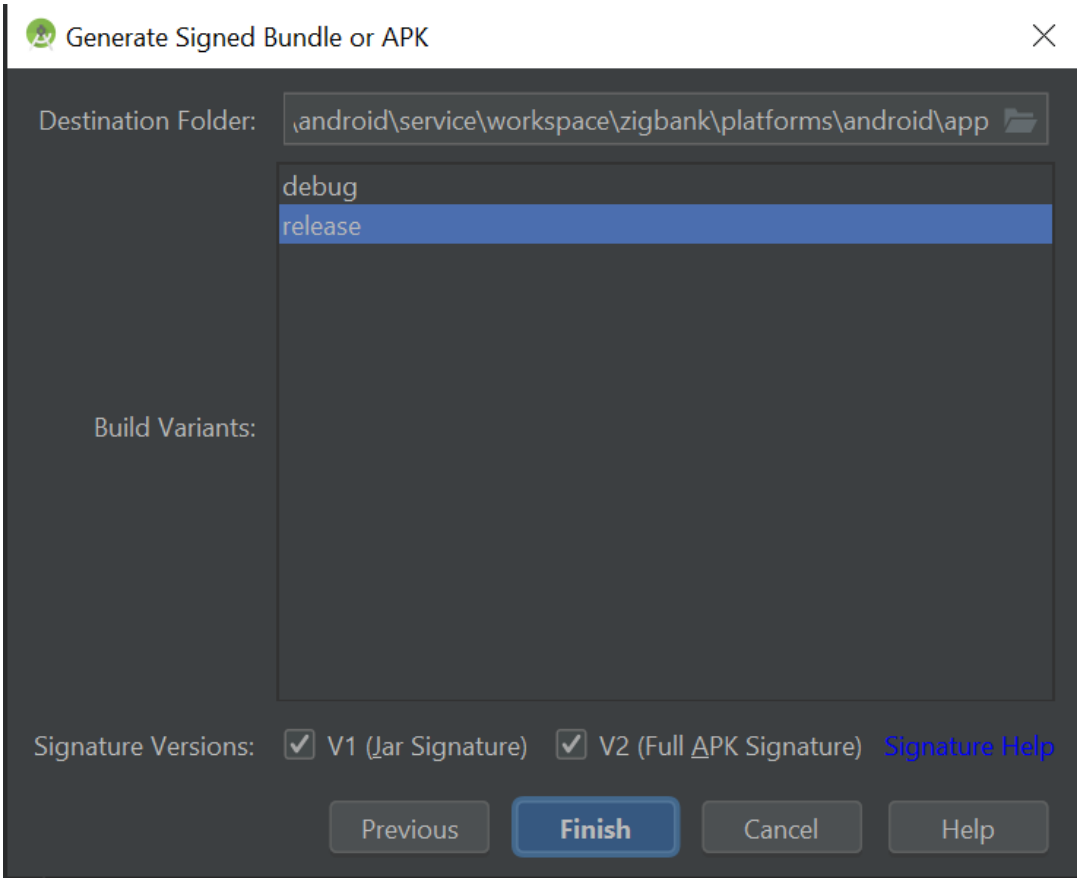


14. If you have an existing keystore.jks file then select choose **Existing** else click **Create New**.





15. 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 **Finish** .

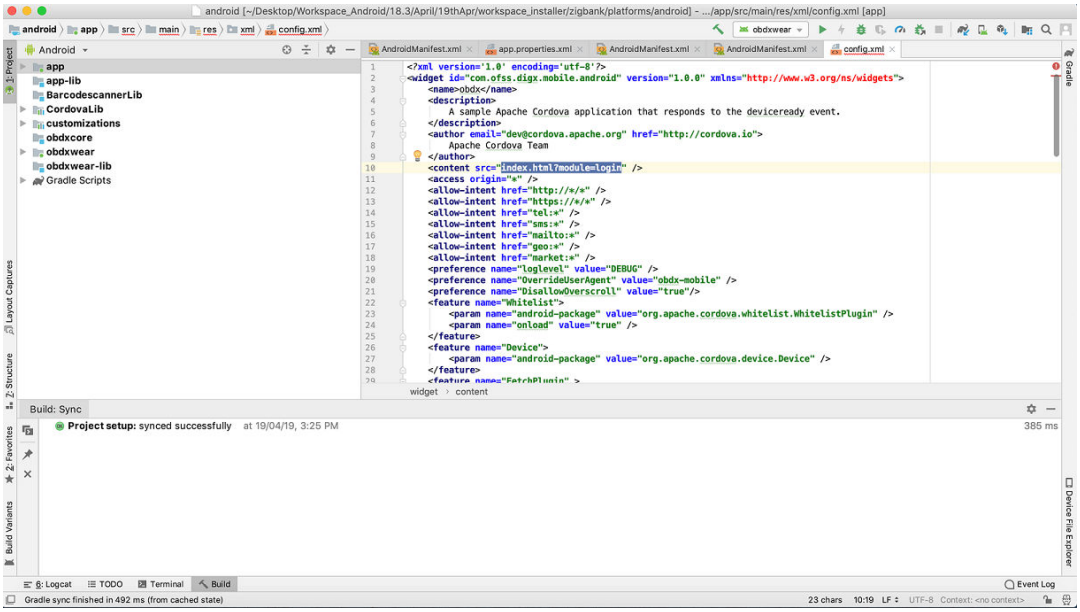


16. This will generate APKby the given name and destination folder. Default APK Destination folder is **`zigbank\platforms\android\app\release`**.

17. Run the App and select Device or Simulator.

18. **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.

19. 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).



20. Application will work on https only, there is no support for http url further.

21. To enable App widget, enable below flag in app.properties file:

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

22. 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.

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

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

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

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

25. 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

This topic provides information on **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”

- Authenticator UI (Follow any one step below)
  This topic provides information on **Authenticator UI (Follow any one step below)**. Please refer section **Authenticator UI (Follow any one step below)** of **Mobile Application Builder Guide-iOS Guide** for Authenticator UI build steps. UI is same for Android & iOS.
- Authenticator Application Workspace Setup
  This topic describes the systematic instruction to **Authenticator Application Workspace Setup** option.

5.1 Authenticator UI (Follow any one step below)

This topic provides information on **Authenticator UI (Follow any one step below)**. Please refer section **Authenticator UI (Follow any one step below)** of **Mobile Application Builder Guide-iOS Guide** for Authenticator UI build steps. UI is same for Android & iOS.

- Using built UI
  This topic provides information on **Using built UI**.
- Using Un-built UI
  This topic provides information on **Using Un-built UI**.
- Building UI Manually
  This topic describes the systematic instruction to **Building UI Manually** option.

5.1.1 Using built UI

This topic provides information on **Using built UI**.

For TOKEN-BASED - Unzip dist.tar.gz directory fromOBDX_Patch_Mobile\authenticator\TOKEN-BASED

5.1.2 Using Un-built UI

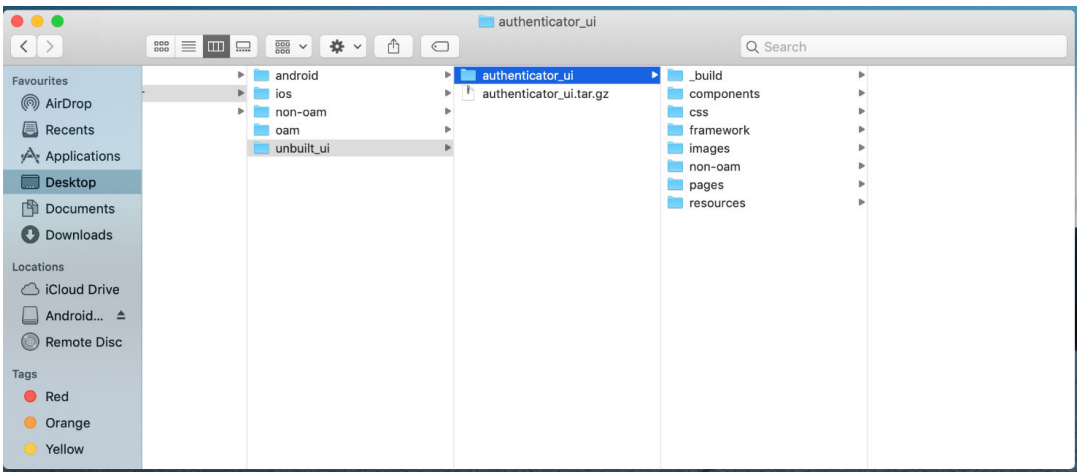
This topic provides information on **Using Un-built UI**.

1. Extract authenticator_ui.tar.gz from OBDX_Patch_Mobile\authenticator\unbuilt_ui. Copy the **token-based/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.

5.1.3 Building UI Manually

This topic describes the systematic instruction to **Building UI Manually** option.

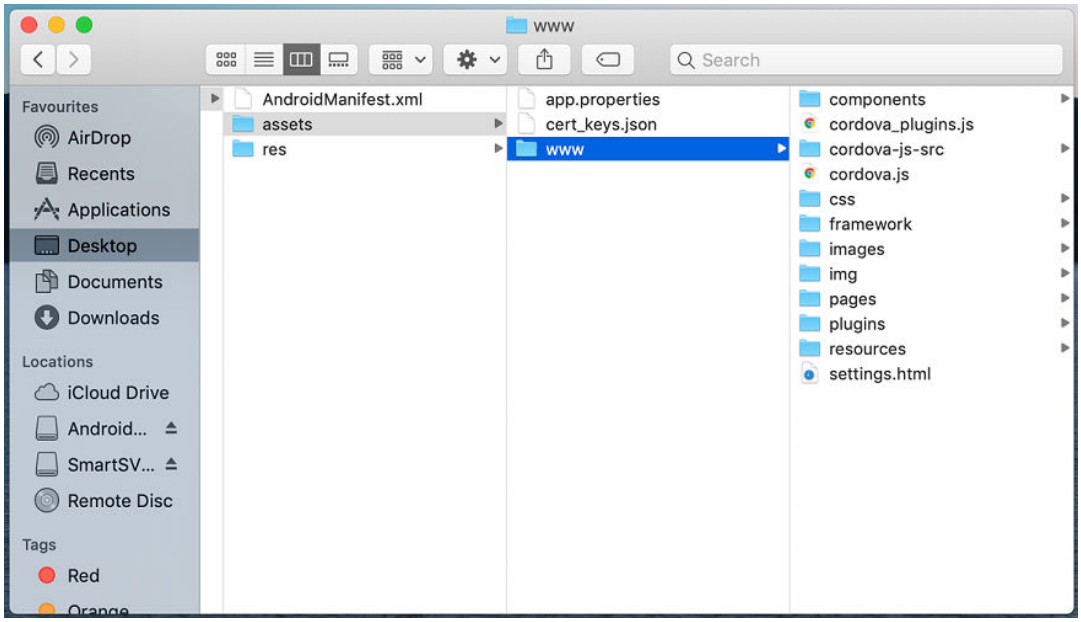
1. Extract authenticator_ui.tar.gz from OBDX_Patch_Mobile\authenticator\unbuilt_ui. The folder structure is as shown :



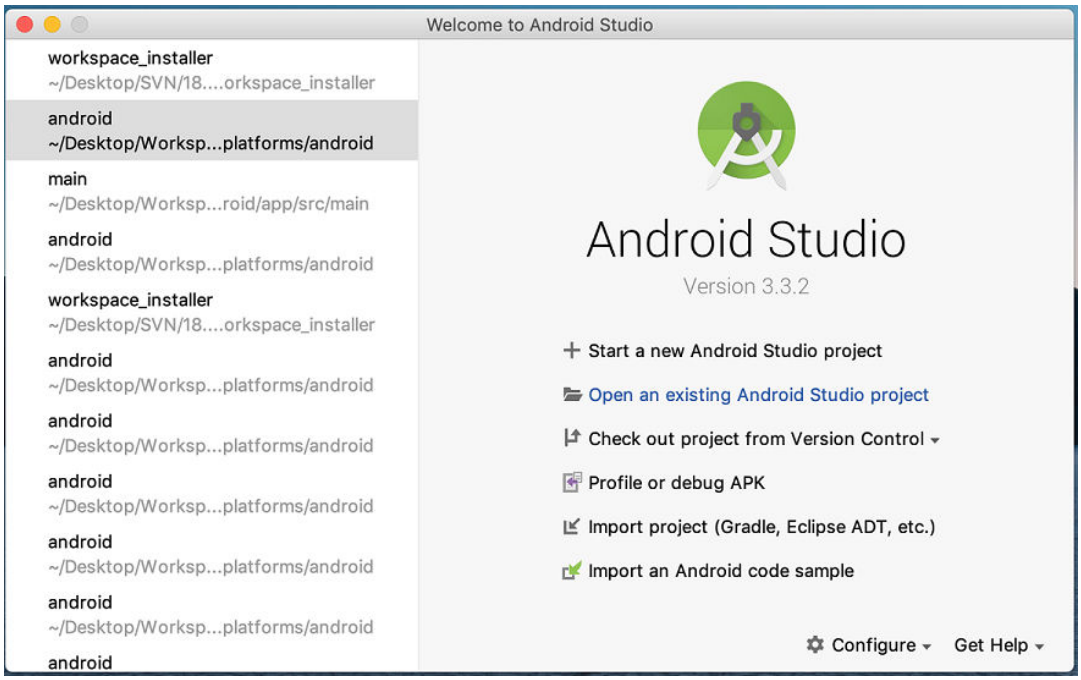
5.2 Authenticator Application Workspace Setup

This topic describes the systematic instruction to **Authenticator Application Workspace Setup** option.

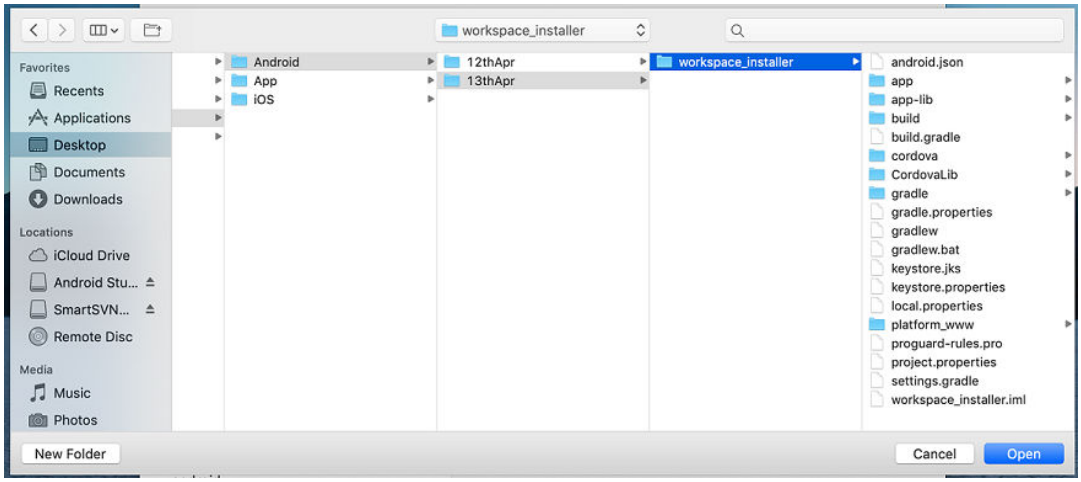
1. Copy UI (Directories – components, css, framework, images, pages, resources)from /dist directory to workspace/installer/app/src/main/assets/www/
   In case any popup appears, click **Replace**.



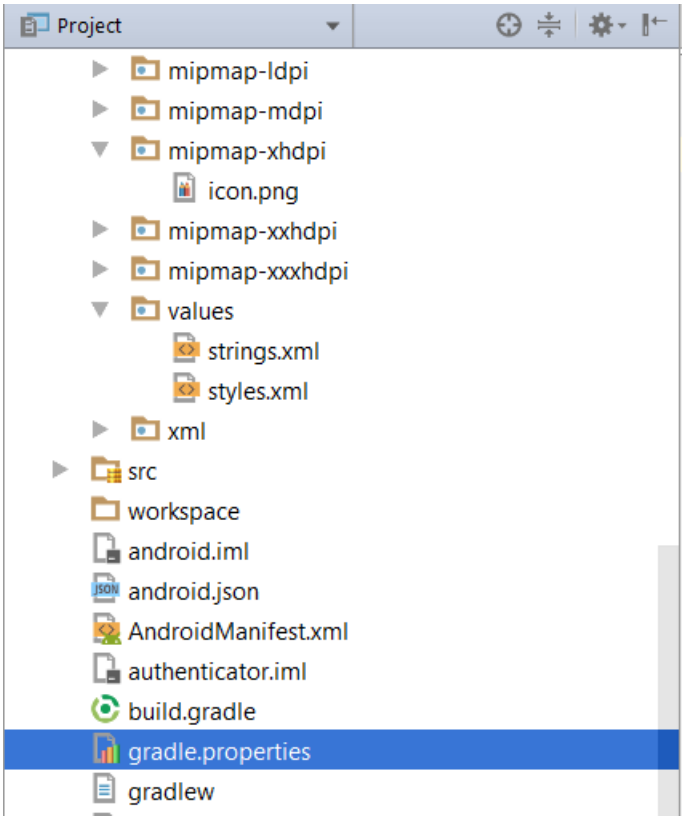
2. Launch Android Studio and open existing project.



3. Open OBDX_Installer/workspace_installer folder in Android Studio.

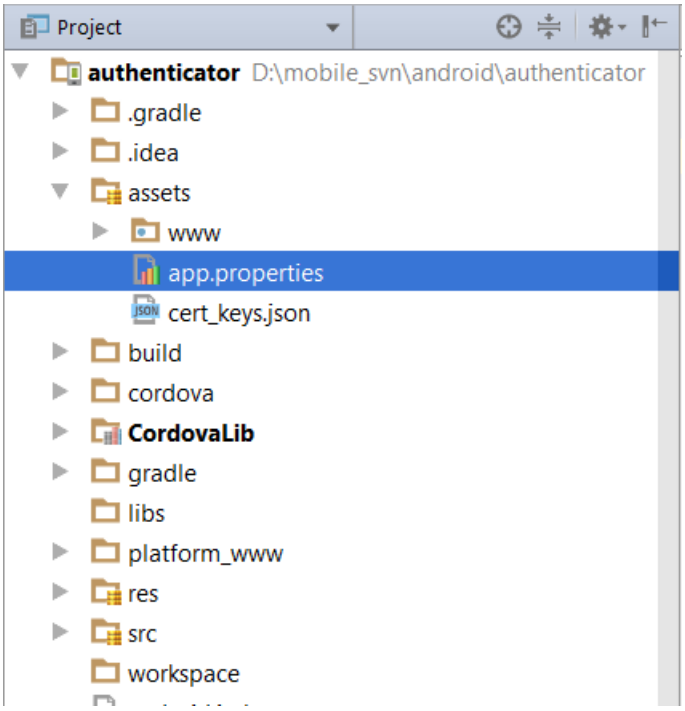


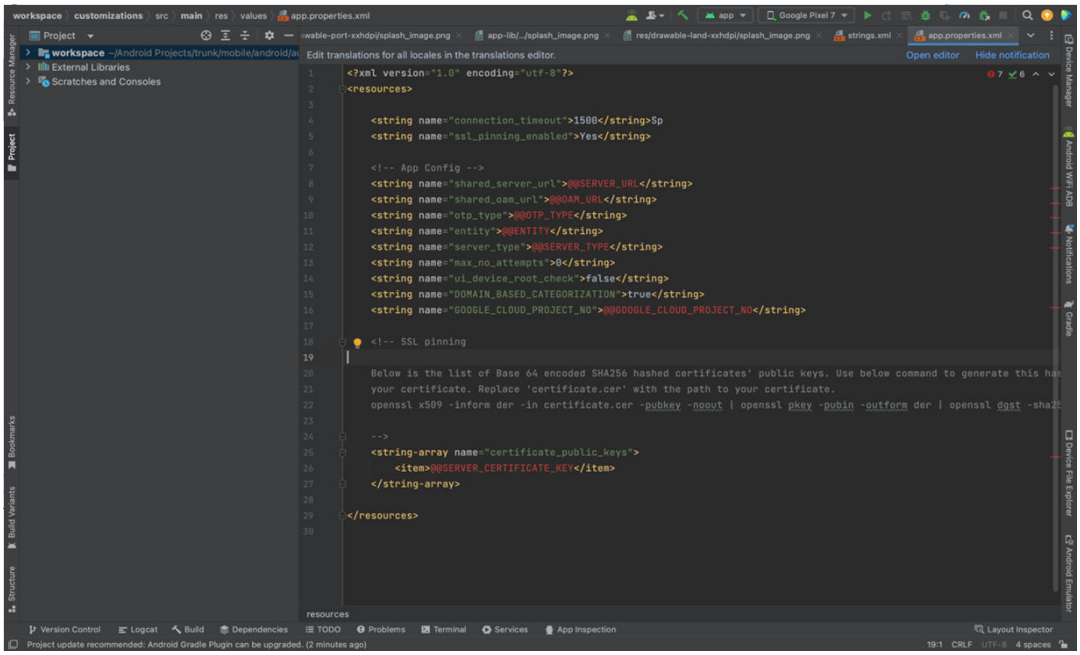
4. Open **gradle.properties** file and update following properties with relevant proxy address if required.




```
systemProp.http.proxyHost = <proxy_address>
        systemProp.https.proxyPort = <port_number>
        systemProp.https.proxyHost = <proxy_address>
        systemProp.http.proxyPort = <port_number>
```

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





connection_timeout =
<timeout_in_milliseconds>
ssl_pinning_enabled = <YES or

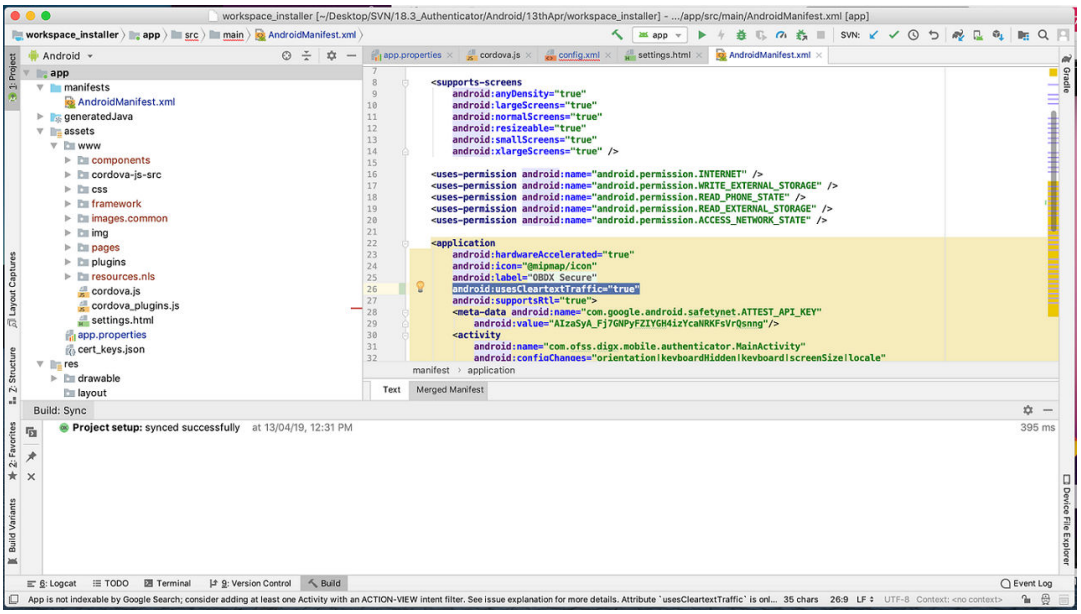
Set OTP type to HOTP/TOTP as per requirement.
Set Server Type to OBDXTOKEN
Set MAX No Attempts greater than 0
Set UI Device root check to true if you want to add check on login button.

> **Note**
>
> If selected authentication mechanism is not OAM based then remove `shared_oam_url` property.

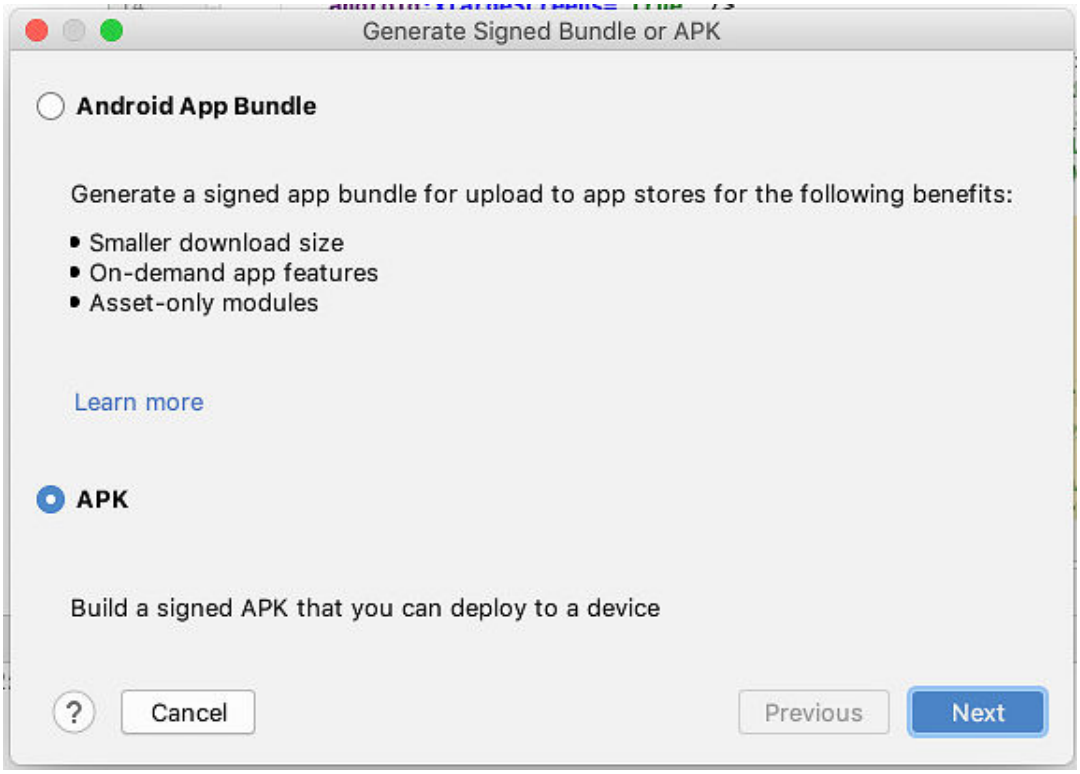
6. Click Build → Clean & Build → Rebuild project in Android Studio.
7. Click on Build → Edit Build Type → app → release.
   Enable minify →true

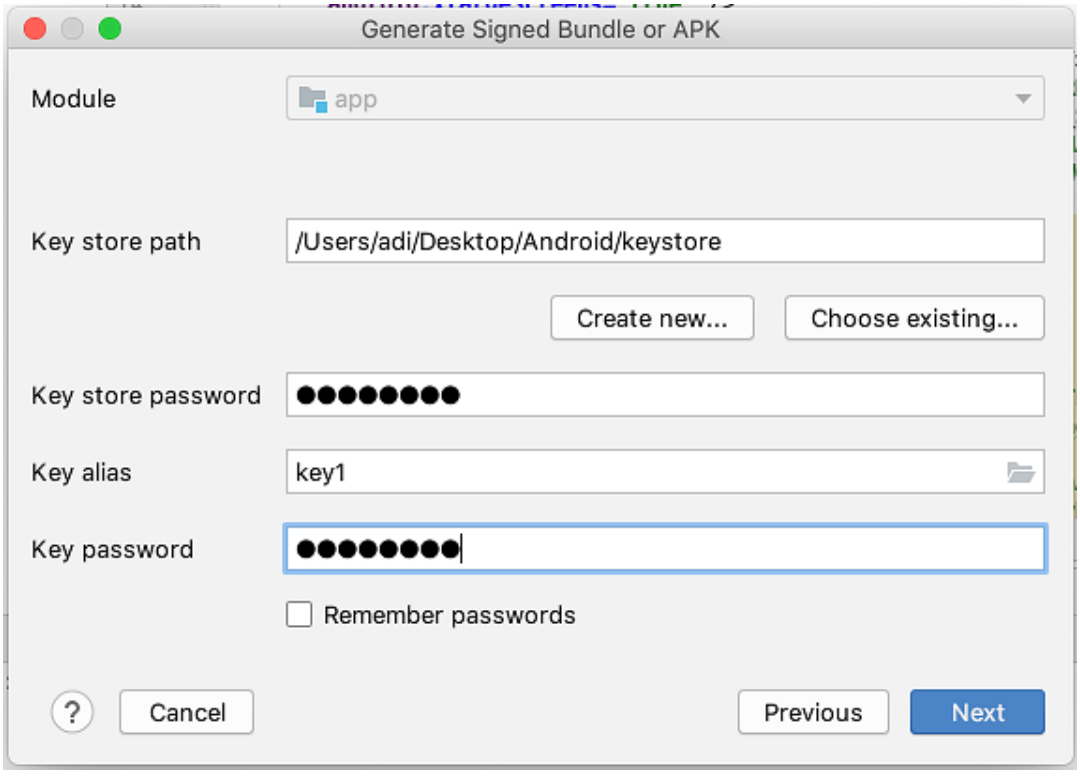
   Add progurard file from workspace_installer/proguard-rules.pro

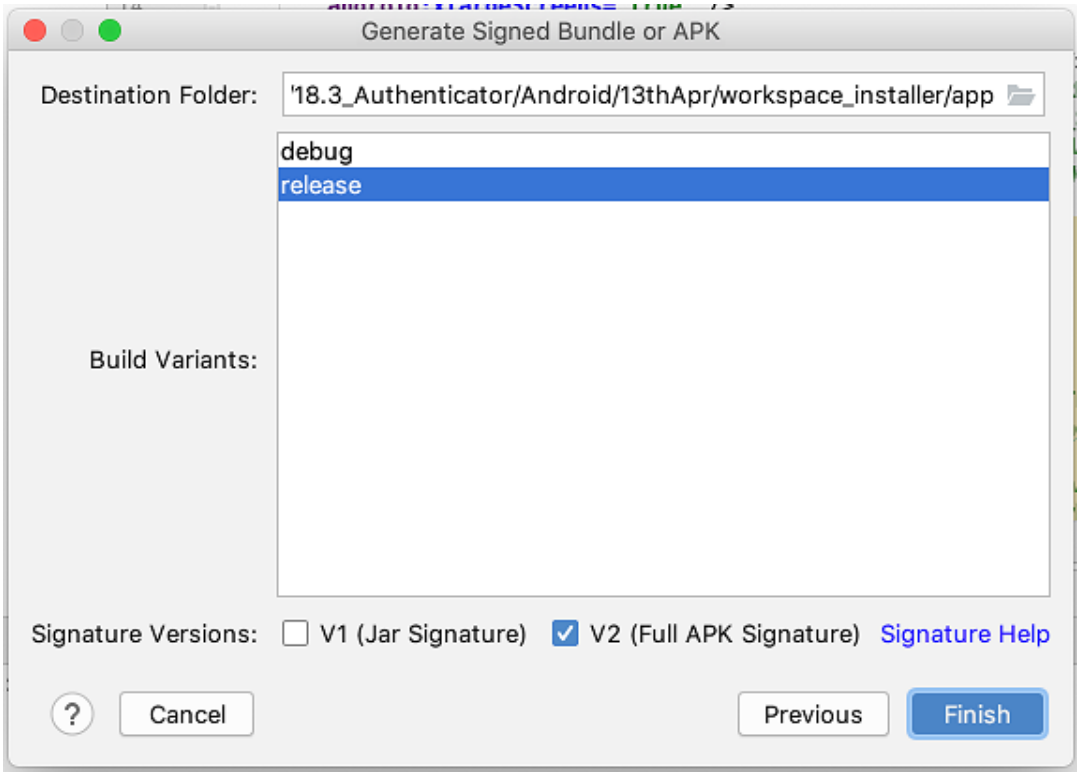
   Click **OK**.
8. If using http protocol for development add (android:usesCleartextTraffic="true") to application tag of AndroidManifest.xml.



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

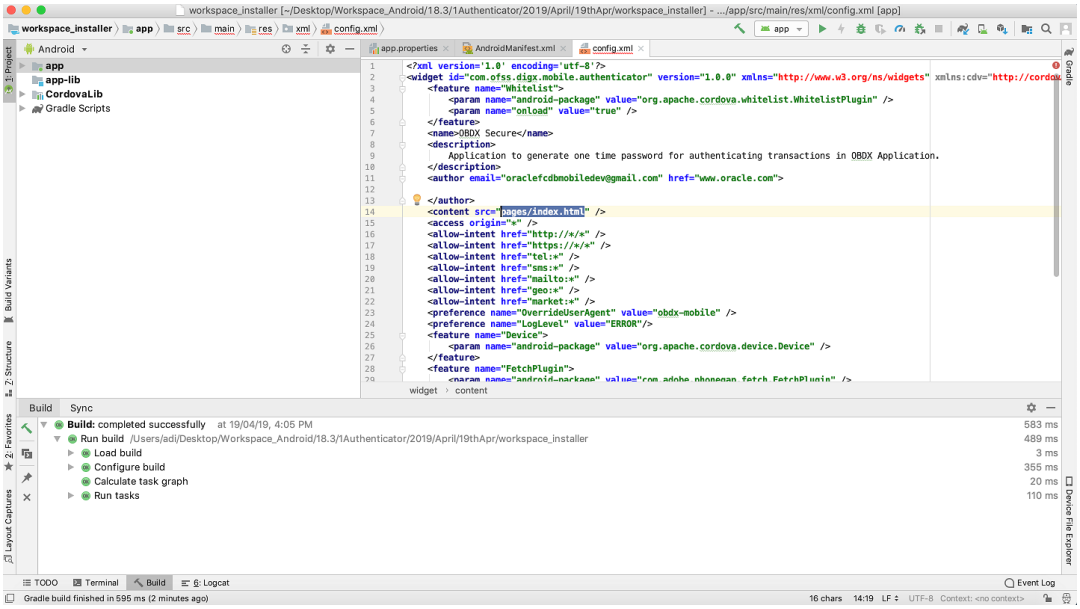






Click **Finish** to generate .apk

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

This topic provides information on **Application Security Configuration**.

Root Check à 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.

Example:

```
<string-array name="certificate_public_keys">

    <item>sha256/5kJvNEMw0KjrCAu7eXY5HZdvyCS13BbA0VJG1RSP91w=</item>

</string-array>
```

Example: 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

This topic provides information on **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\ww

w\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\sr

c\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\r

es\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

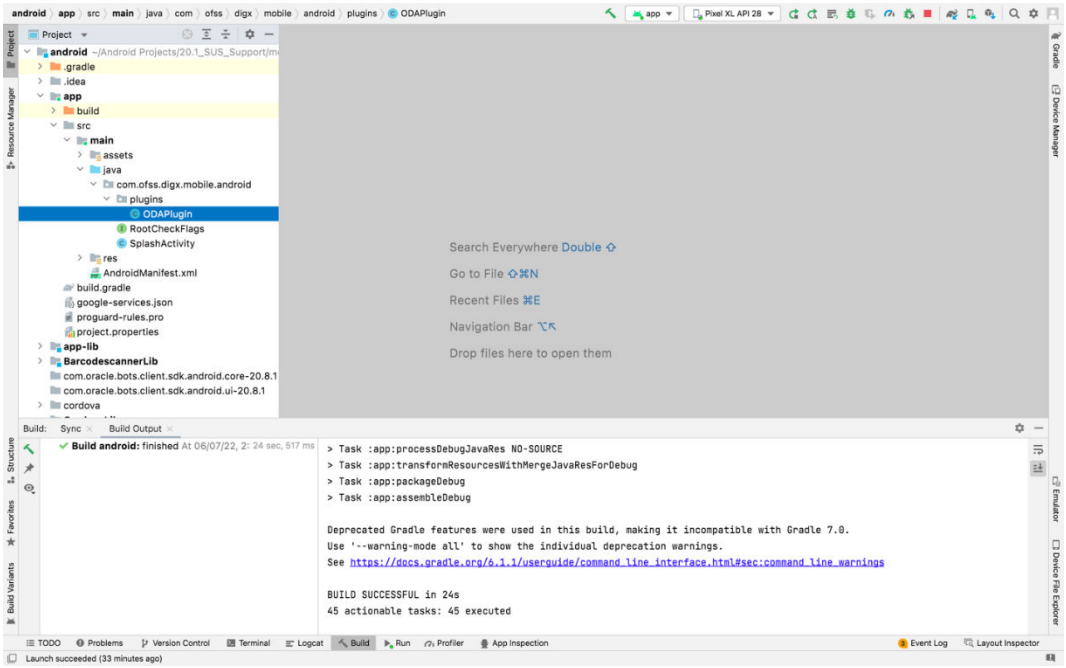
8

ODA Chatbot Inclusion

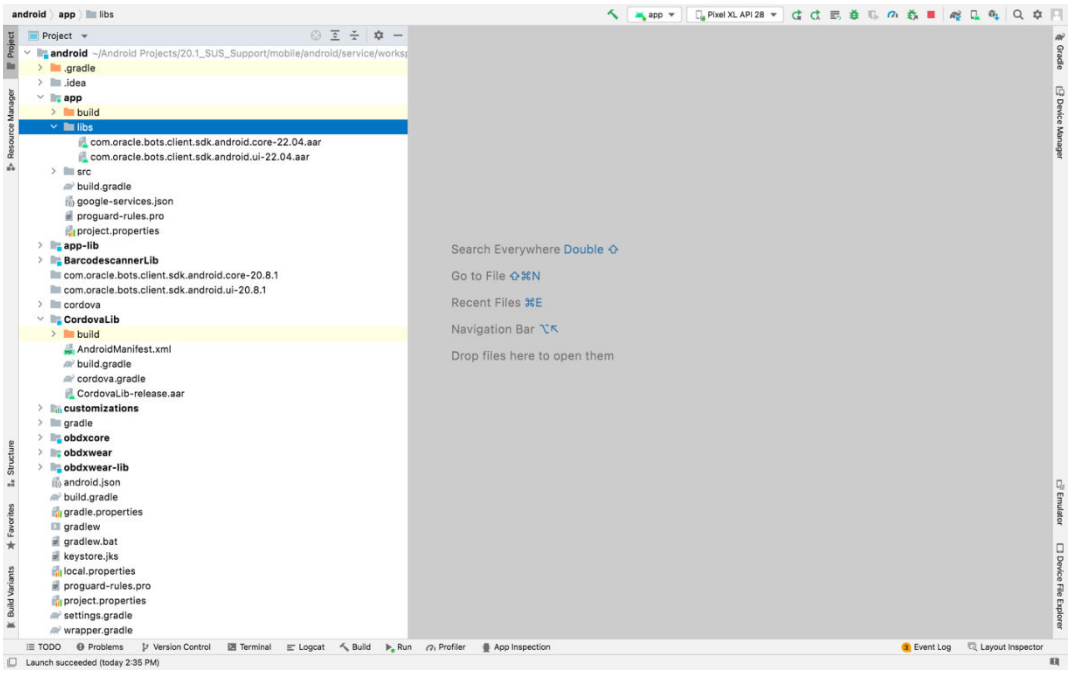
This topic describes the systematic instruction to **ODA Chatbot Inclusion** option.

To enable ODA Chatbot services in the mobile app, the following changes needs to be made:

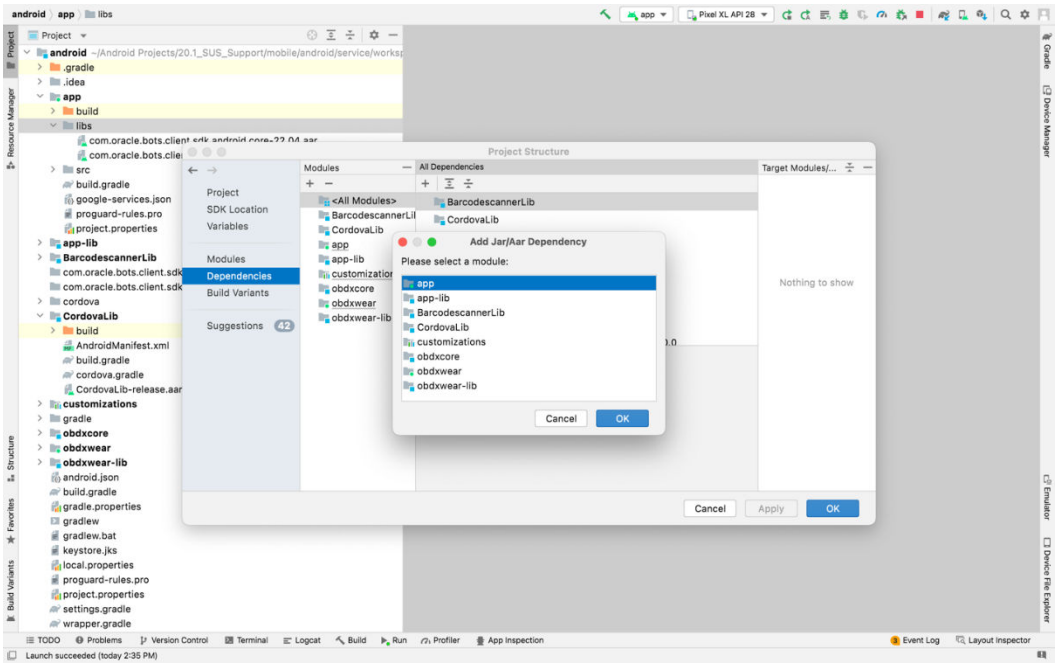
1. Copy ODAPlugin.java from `workspace_installer/AppExtension/oda` to `workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/plugins/`



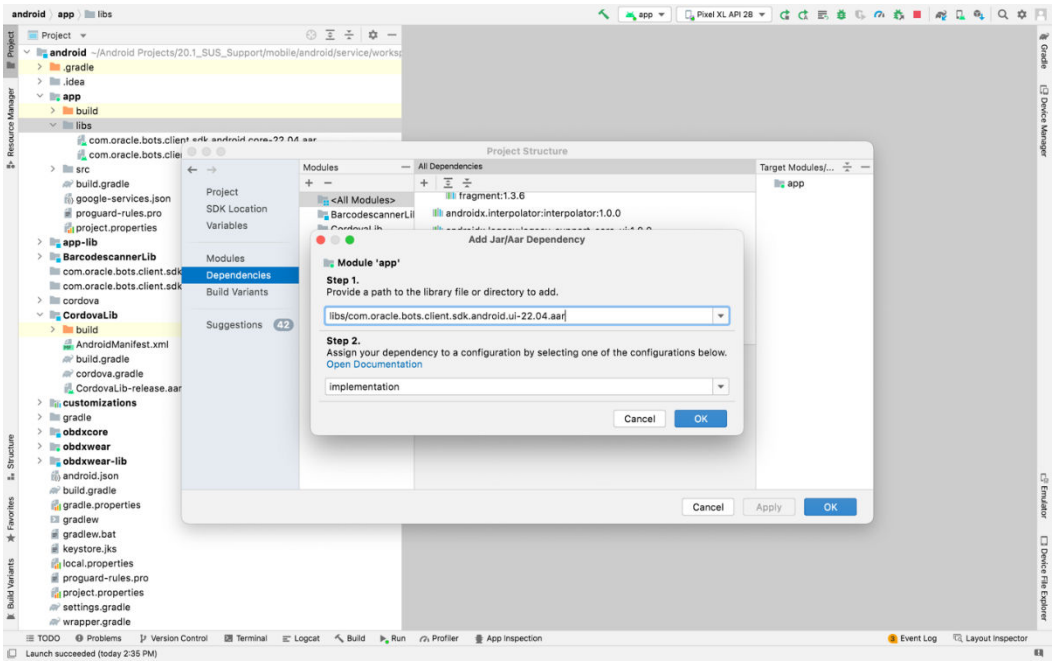
2. Download ODA Android sdk from below link-https://www.oracle.com/downloads/cloud/amce-downloads.html
3. Add libs folder at zigbank\platforms\android\app and copy below files from downloaded sdk folder in it.
   a. com.oracle.bots.client.sdk.android.core-xx.aar
   b. com.oracle.bots.client.sdk.android.ui-xx.aar

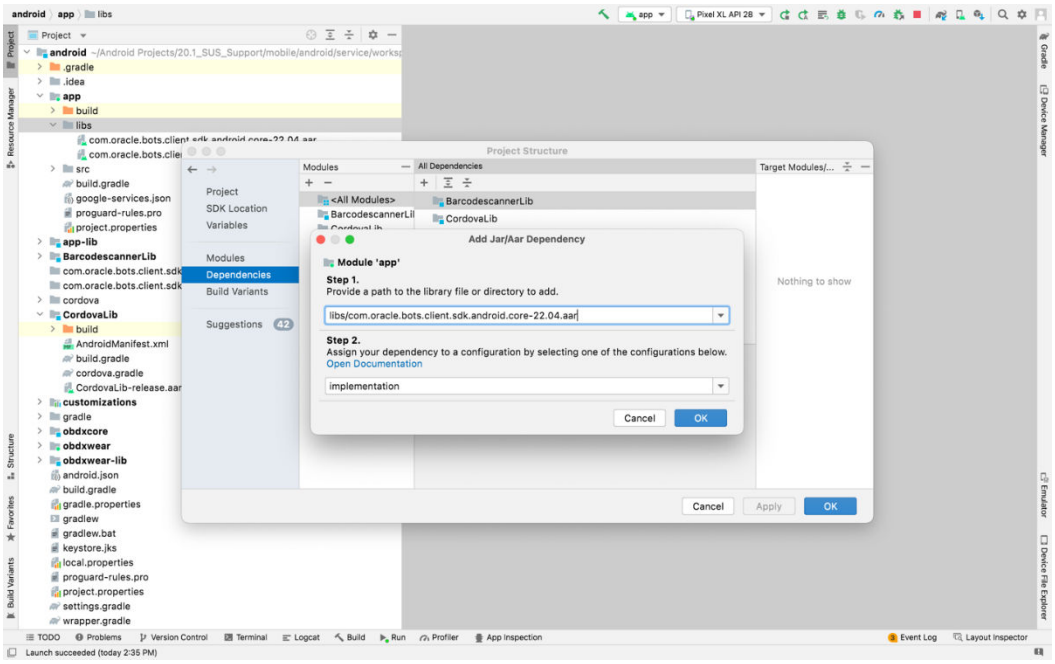


4. In Android Studio follow below steps-
   File → Project Structure →Dependencies
5. Click on **"+"** icon and select **JR/AAR Dependency** and select app module and click **Ok**.



6. Add both .aar file paths from step3. Then click **Apply** and **Ok**.





7. Add Chatbot ID and Chatbot URL in app.properties.xml (`zigbank\platforms\android\customizations\src\main\res\values`)
<string name="CHATBOT_ID">@@CHATBOT_ID</string>

<string name="CHATBOT_URL">@@CHATBOT_URL</string>

9

Push Notification 2FA configuration

This topic provides information on **Push Notification 2FA configuration**.

1. This is 2fa authentication set for any transaction. With the setup, whenever any user initiates any transaction, they will receive a push notification on the registered device. They have to click on the notification to accept/reject the transaction. Based on the action, the transaction will be proceeded.
2. Note: PUSH notifications are received only if user has allowed push notification when the application was installed and logged in the mobile application for the first time.
3. If user disallows the notification when the application for installed for the first time., they will not receive any push notifications on their devices.
4. If Push notification 2fa is enabled at bank side for any transaction then, the screen displays message to wait for the push notification to accept/reject the transaction authentication. The message displayed on the text as well contains a timer of 5 minutes displayed on the UI. This value is set in the UI code. If bank needs to change this value, bank needs to update the value in UI code:
   File path: channel/metadata/user-components/push-out-of-band/push-out-of-band/hook.js

   Code to be changed: const mins = <<value>>;

   Update the value to what bank needs to set it. This value is in minutes.

   So, ideally 5 minutes (existing value in base UI code) is an ideal time. Any changes made in this value should satisfy below pre-condition.
5. There is an OTP expiration time set in “digx_fw_config_ALL_b” table.
6. Also, there is business policy check set to 10 minutes for validation of the generated 2fa token. Bank can write their own business policy where they can modify the 10 minutes time.
   So, the time in UI code should not exceed 10 minutes and OTP expiration time in “digx_fw_config_ALL_b” table.

Index