

Oracle® Banking Digital Experience

Mobile Application Builder - Android Guide



Innovation Release 25.1.2.0.0

G51762-01

April 2026

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Copyright © 2015, 2026, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Purpose	i
Pre-requisites	i
Audience	i
Documentation Accessibility	i
Critical Patches	ii
Diversity and Inclusion	ii
Related Resources	ii
Conventions	ii
Screenshot Disclaimer	iii
Acronyms and Abbreviations	iii
Post-requisites	iii

1 OBDX Servicing Application

1.1 Prerequisites	1
1.2 Create project using Remote UI	3
1.3 Importing in Android Studio	3
1.4 Widget Functionality	4
1.5 Scan to Pay from Application Icon	5
1.6 Passkey (Passwordless login)	5
1.7 Deeplinking - To open reset password, claim money links with the application	8
1.8 Device Registration and Push Registration Functionality	9
1.9 Location Tracking Metrics	10
1.10 Displaying Rate Option to Redirect to Playstore Page	11
1.11 Enabling Force Update	11
1.12 Splash Screen Migration	11
1.13 App Update Manager	12

2 Google Play Integrity

3	FCM Push Notifications	
4	Build Release Artifacts	
5	OBDX Authenticator Application	
5.1	Authenticator UI (Follow any one step below)	1
5.1.1	Using built UI	1
5.1.2	Using Un-built UI	1
5.1.3	Building UI Manually	2
5.2	Authenticator Application Workspace Setup	2
6	Application Security Configuration	
7	Adding Custom Cordova Plugin	
8	ODA Chatbot Inclusion	
9	Push Notification 2FA configuration	
10	System Configuration	
11	Pre-requisites for Production App	
	Index	

Preface

- [Purpose](#)
- [Pre-requisites](#)
- [Audience](#)
- [Documentation Accessibility](#)
- [Critical Patches](#)
- [Diversity and Inclusion](#)
- [Related Resources](#)
- [Conventions](#)
- [Screenshot Disclaimer](#)
- [Acronyms and Abbreviations](#)
- [Post-requisites](#)

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.

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.
<i>italic</i>	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.

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

- [Prerequisites](#)
This topic provides information on **Prerequisites**.
- [Create project using Remote UI](#)
This topic provides information on **Create project using Remote UI**.
- [Importing in Android Studio](#)
This topic describes the systematic instruction to **Importing in Android Studio** option.
- [Widget Functionality](#)
This topic provides information on **Widget Functionality** .
- [Scan to Pay from Application Icon](#)
This topic provides information on **Scan to Pay from Application Icon**.
- [Passkey \(Passwordless login\)](#)
This topic describes the systematic instruction to **Passkey (Passwordless login)** option.
- [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.
- [Device Registration and Push Registration Functionality](#)
This topic provides information on **Device Registration and Push Registration Functionality**.
- [Location Tracking Metrics](#)
This topic provides information on **Location Tracking Metrics**.
- [Displaying Rate Option to Redirect to Playstore Page](#)
This topic provides information on **Displaying Rate Option to Redirect to Playstore Page**.
- [Enabling Force Update](#)
This topic provides information on **Enabling Force Update**.
- [Splash Screen Migration](#)
This topic provides information on **Splash Screen Migration**.
- [App Update Manager](#)
This topic provides information on **App Update Manager**.

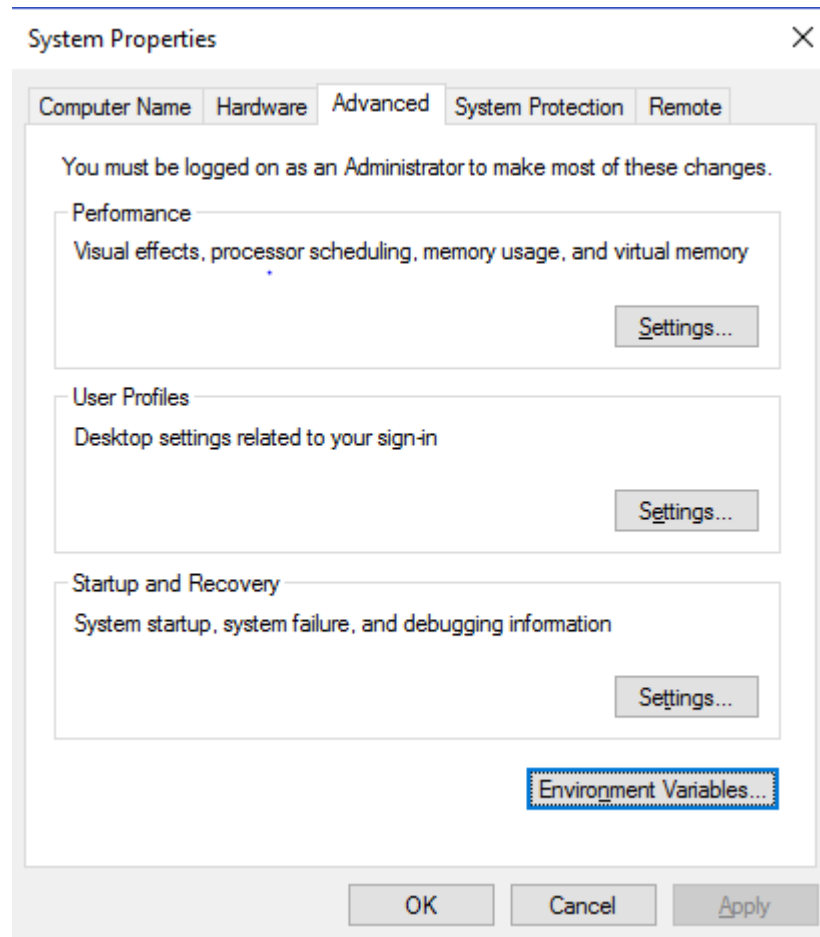
1.1 Prerequisites

This topic provides information on **Prerequisites**.

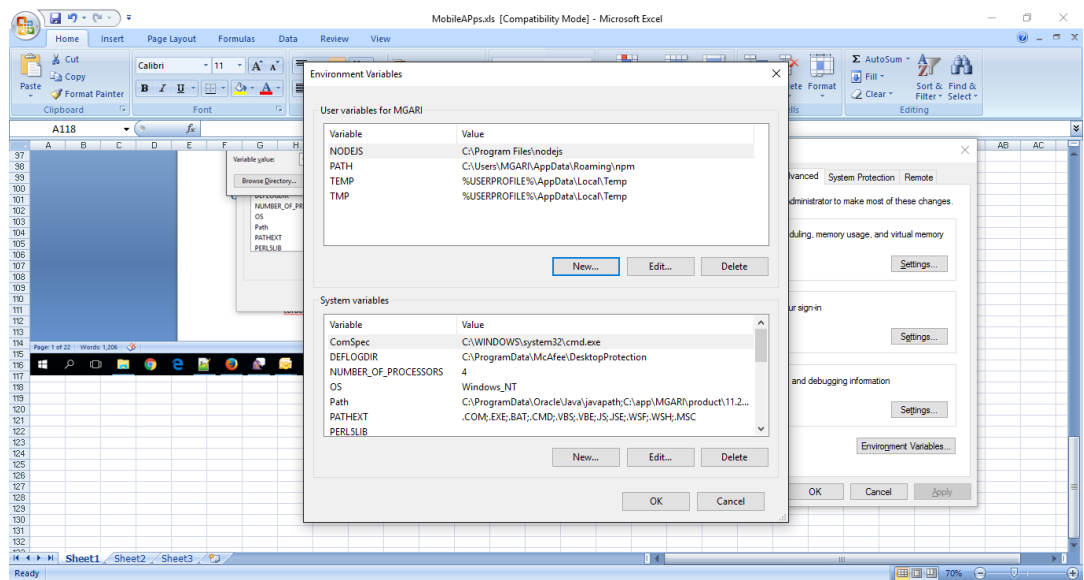
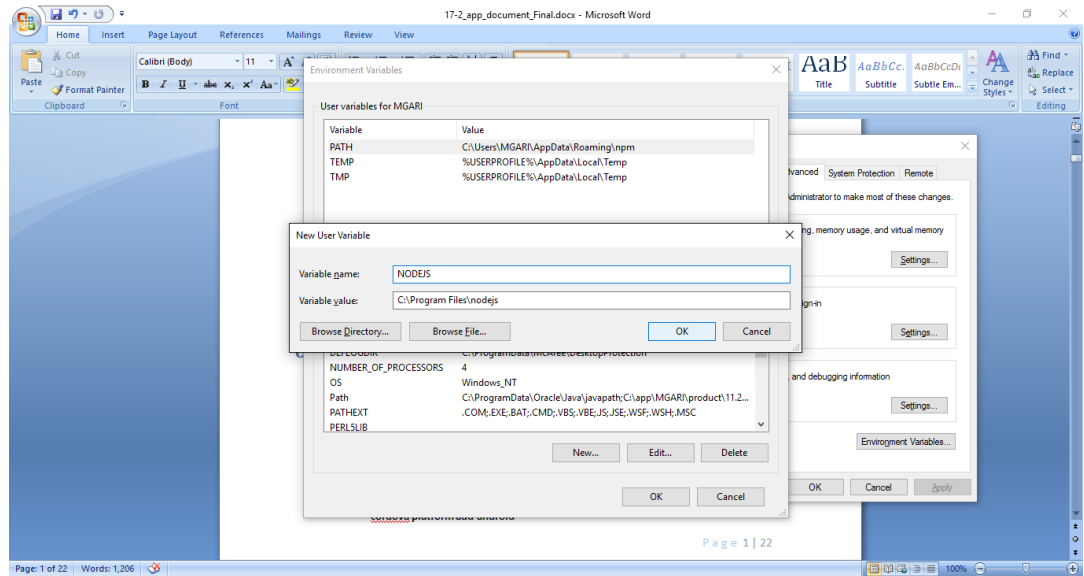
OBDX Android App is supported only on versions n (current) and n-1 release.

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.

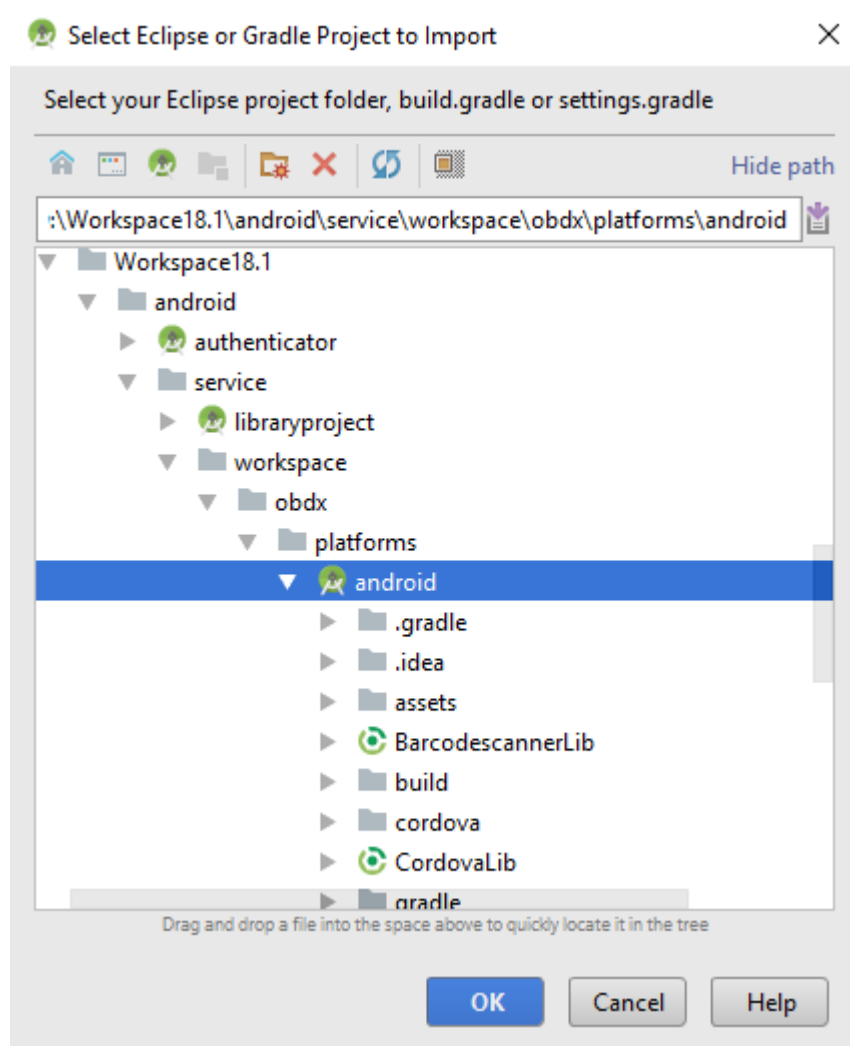
After this proceed to **Section 1.4: Importing in Android Studio** directly.

1.3 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.



1.4 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>
```

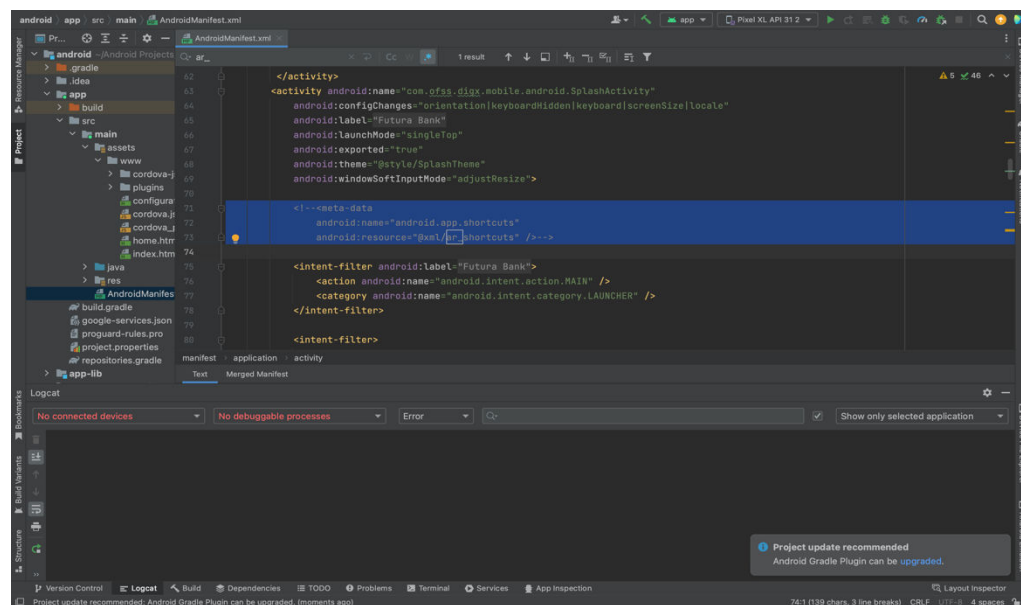
If bank does not want this feature, then they can disable this by making above flag to false.

1.5 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.



1.6 Passkey (Passwordless login)

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

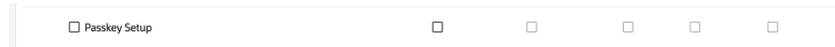
Passkeys are a safer and easier replacement for passwords. With passkeys, users can sign in to apps and websites using a biometric sensor (such as a fingerprint or facial recognition), PIN, or pattern. This provides a seamless sign-in experience, freeing your users from having to remember usernames or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

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

TO DISABLE THIS OPTION:

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

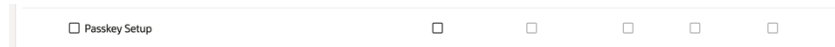
1. Remove RTM access from Client Servicing → Authentication → Passkey Setup for Mobile Application/Mobile (Responsive)/Internet touch points.



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

TO ENABLE THIS OPTION:

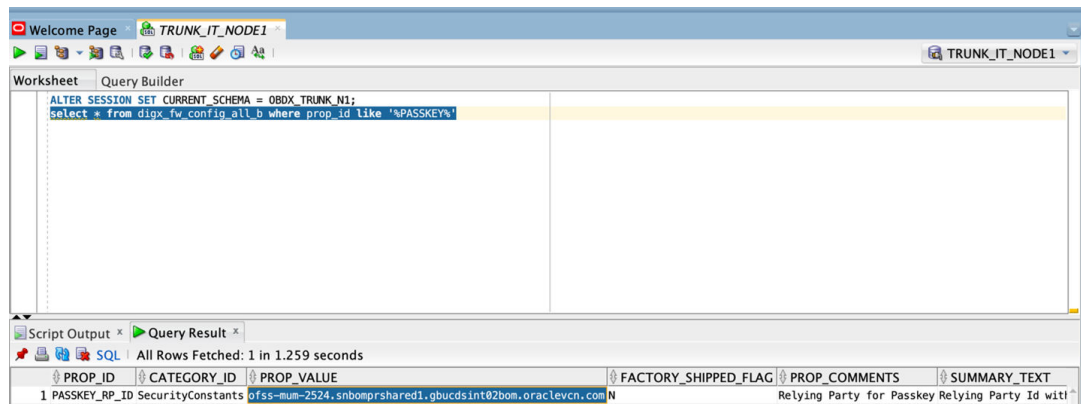
1. Add RTM access from Client Servicing → Authentication → Passkey Setup for Mobile Application, Mobile (Responsive) and Internet touch points.



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 setup.

Server-Side Setup:

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

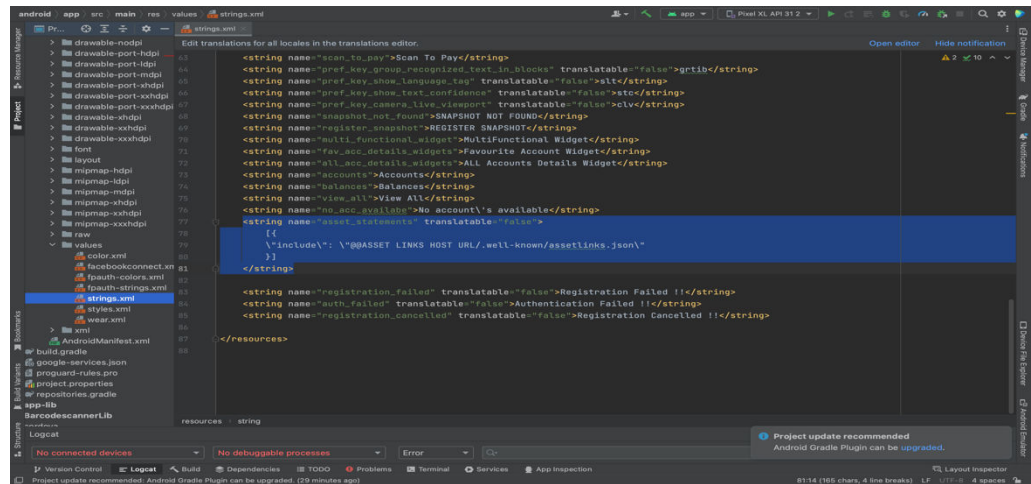
Need to change host and port in `Obdx.conf` as:

```
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 be any redirects for accessing this file.

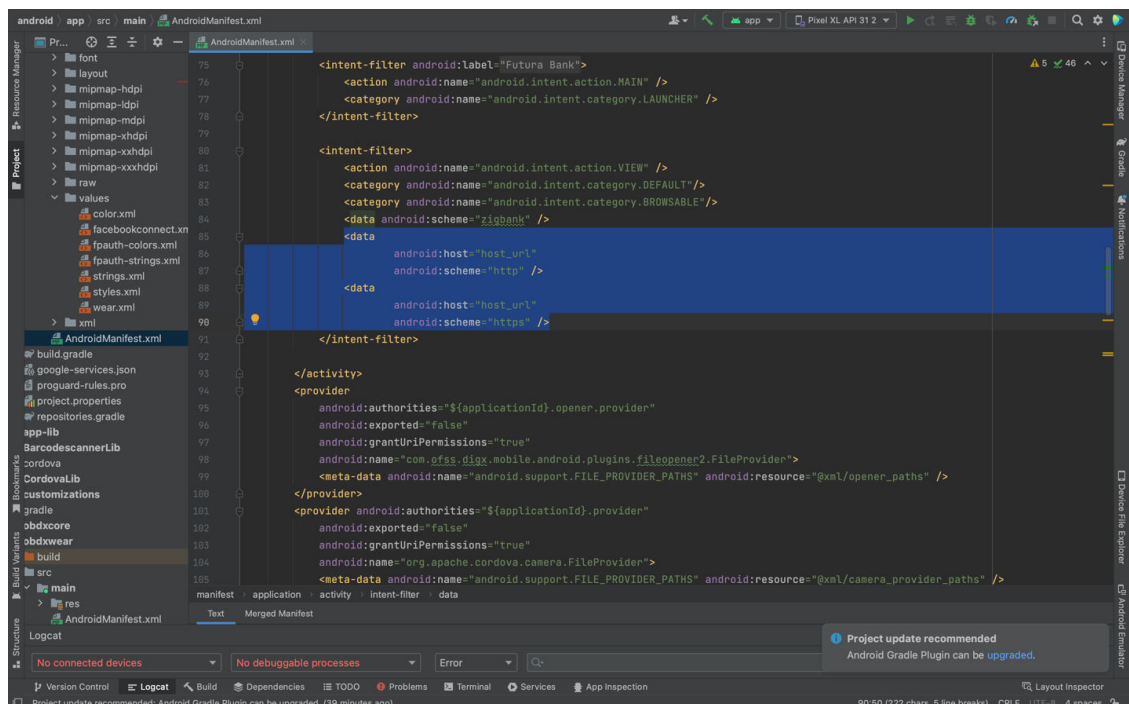
c. Add `assetlinks.json` file host in app's `strings.xml` file.



1.7 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:

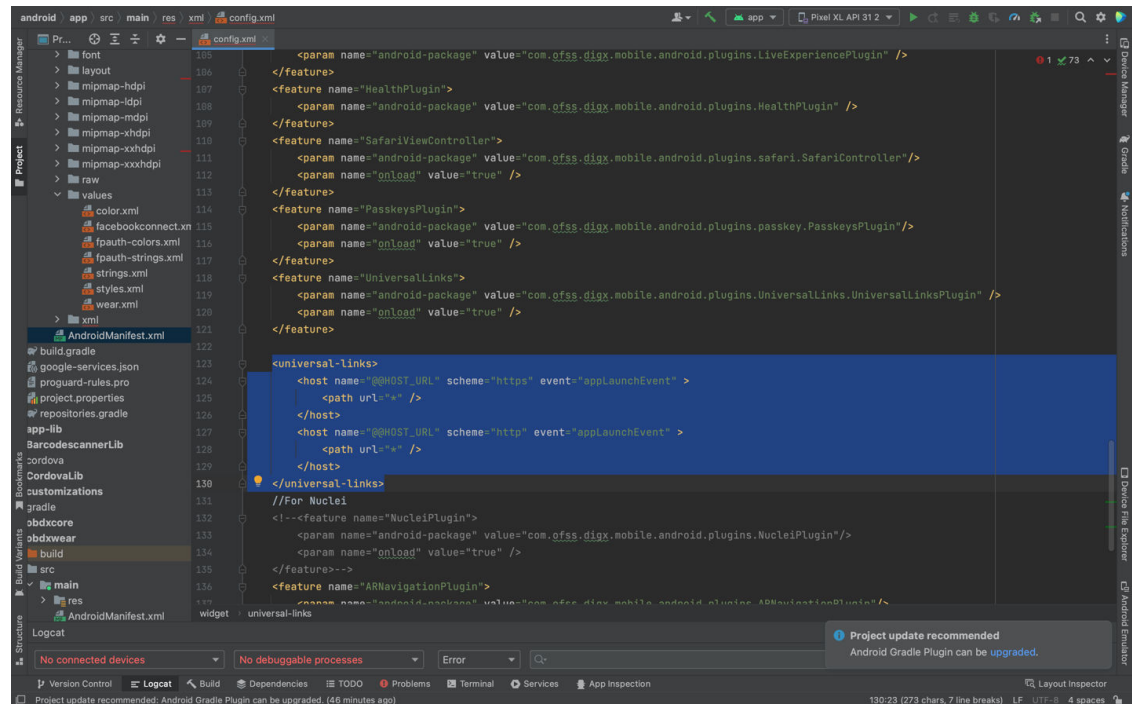


Note

Add host url without https or http.

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

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



```
105 <param name="android-package" value="com.ofss.digx.mobile.android.plugins.LiveExperiencePlugin" />
106 </feature>
107 <feature name="HealthPlugin">
108   <param name="android-package" value="com.ofss.digx.mobile.android.plugins.HealthPlugin" />
109 </feature>
110 <feature name="SafariViewController">
111   <param name="android-package" value="com.ofss.digx.mobile.android.plugins.safari.SafariController"/>
112   <param name="onload" value="true" />
113 </feature>
114 <feature name="PasskeysPlugin">
115   <param name="android-package" value="com.ofss.digx.mobile.android.plugins.passkey.PasskeysPlugin"/>
116   <param name="onload" value="true" />
117 </feature>
118 <feature name="UniversalLinks">
119   <param name="android-package" value="com.ofss.digx.mobile.android.plugins.UniversalLinks.UniversalLinksPlugin" />
120   <param name="onload" value="true" />
121 </feature>
122 <universal-links>
123   <host name="@@HOST_URL" scheme="https" event="appLaunchEvent" >
124     <path url="*" />
125   </host>
126   <host name="@@HOST_URL" scheme="http" event="appLaunchEvent" >
127     <path url="*" />
128   </host>
129 </universal-links>
130 //For Nuclei
131 <!--<feature name="NucleiPlugin">
132   <param name="android-package" value="com.ofss.digx.mobile.android.plugins.NucleiPlugin"/>
133   <param name="onload" value="true" />
134 </feature-->
135 <feature name="ARNavigationPlugin">
136   <param name="android-package" value="com.ofss.digx.mobile.android.plugins.ARNavigationPlugin"/>
137 </feature>
```

1.8 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.*

1.9 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.

1.10 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.

1.11 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>.

Please note below things:

1. Make sure that the app that you are using to test the in-app updates, has the lower version code than the update version code.
2. Test with signed builds only.
3. Enable Internal app sharing.

1.12 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.

1.13 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.

New Project

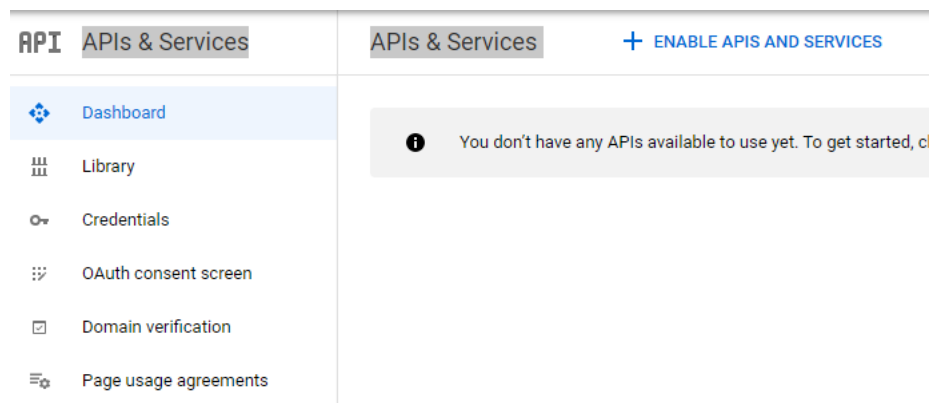
Project name ?

SafetyNet

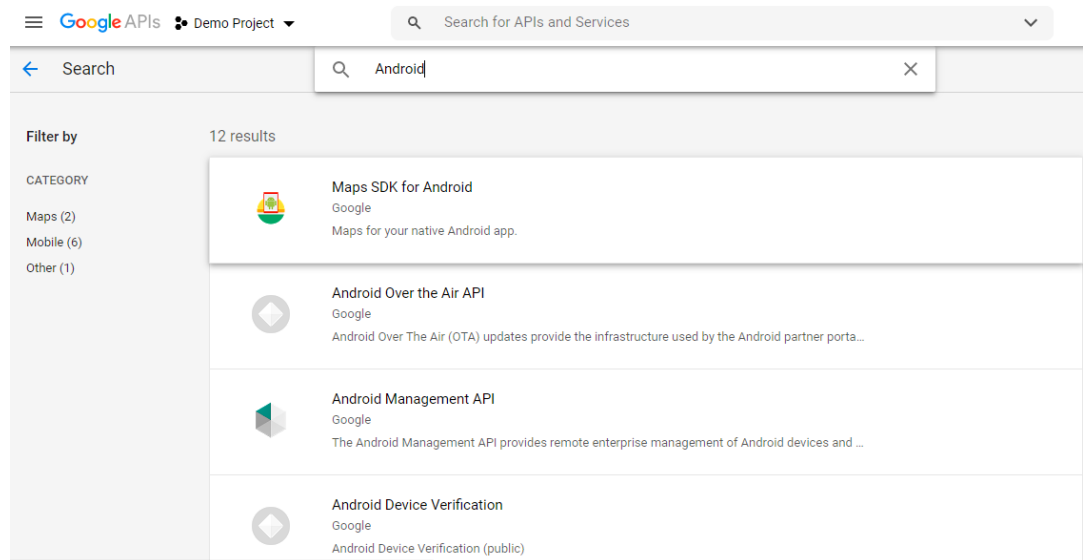
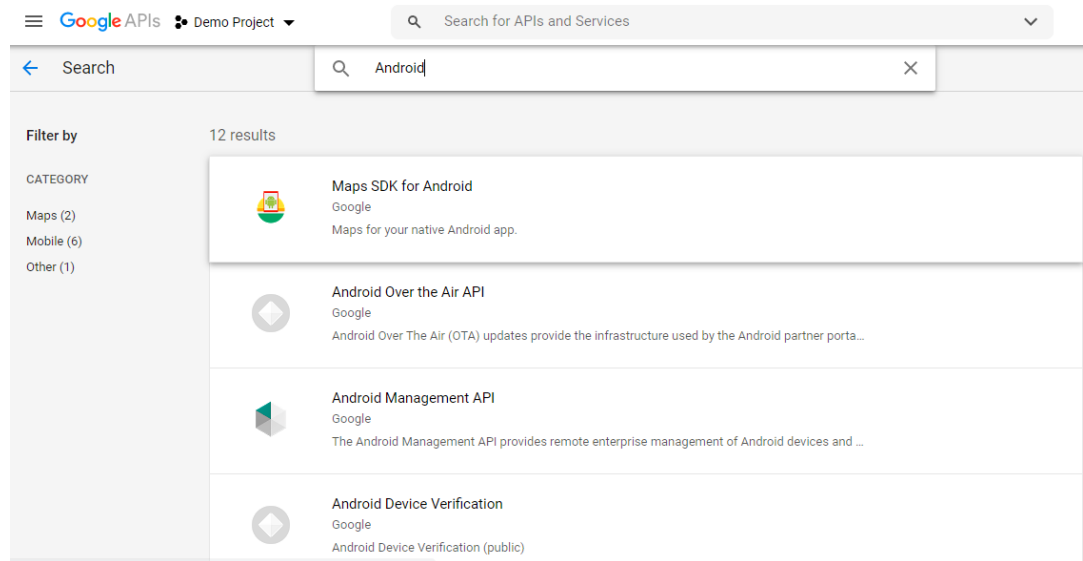
Your project ID will be safetynet-161214 ? Edit

CANCEL CREATE

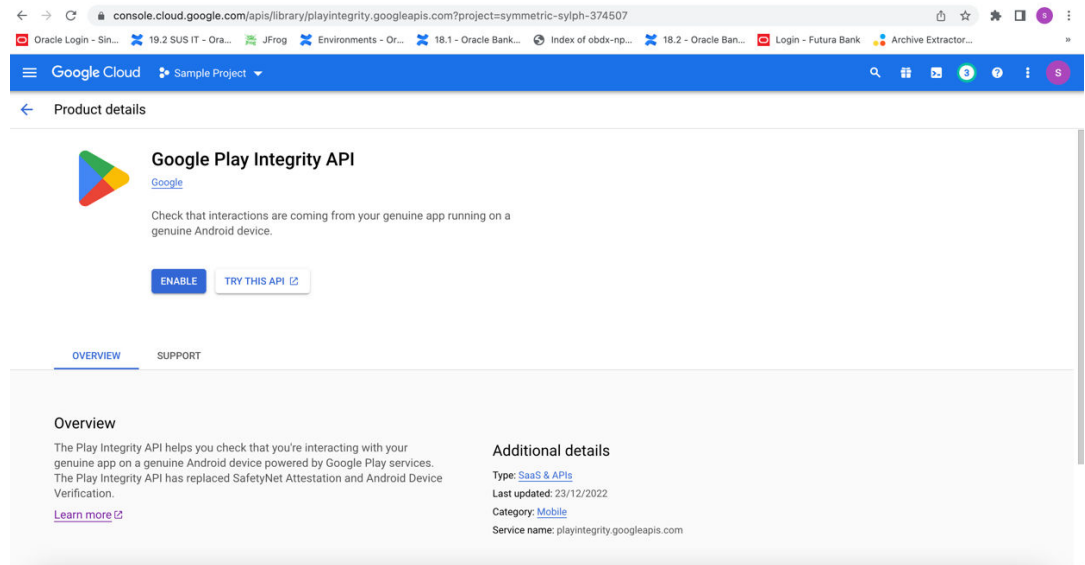
3. Choose **API's & Services** option from side bar.
4. In API's & Services → Dashboard → Choose **Enable APIS AND SERVICES**.



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

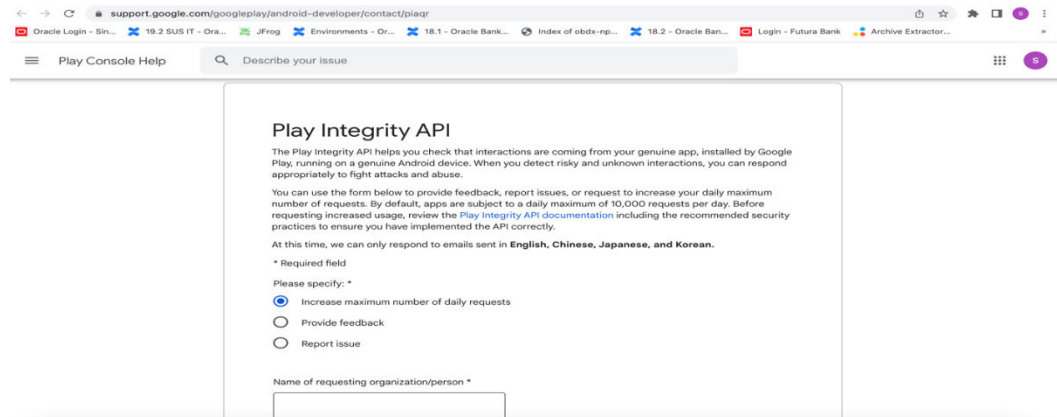


6. Click on Google Play Integrity API and enable it.



The screenshot shows the Google Play Integrity API product details page in the Google Cloud console. The page title is "Google Play Integrity API" and it includes a description: "Check that interactions are coming from your genuine app running on a genuine Android device." There are two buttons: "ENABLE" and "TRY THIS API". Below the description, there are tabs for "OVERVIEW" and "SUPPORT". The "OVERVIEW" tab is selected, showing an overview of the API and additional details such as "Type: SaaS & APIs", "Last updated: 23/12/2022", "Category: Mobile", and "Service name: playintegrity.googleapis.com".

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>.



The screenshot shows the Google Play Integrity API quota request form. The page title is "Play Integrity API" and it includes a description: "The Play Integrity API helps you check that interactions are coming from your genuine app, installed by Google Play, running on a genuine Android device. When you detect risky and unknown interactions, you can respond appropriately to fight attacks and abuse." There is a text box for "Name of requesting organization/person *". Below the text box, there are three radio button options: "Increase maximum number of daily requests" (selected), "Provide feedback", and "Report issue".

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

Oracle Login - Sin... 19.2 SUS IT - Ora... jFrog Environments - Or... 18.1 - Oracle Bank... Index of obdx-np... 18.2 - Oracle Ban... Login - Futura Bank Archive Extractor...

Play Console Help Describe your issue

How are you calling the Play Integrity API? *

My app is calling the API directly

A third party I'm using in the app is calling the API, please specify

How often will you call the API for each user? *

Once per day or less

Once per hour

Once per 15 min

Once per 5 min or more

Is there any PII or SPII used for the nonce (e.g. user id, user name, phone number, Android ID, SSN, etc)? *

Yes, but hashed or encrypted

Yes, in plain-text

No

Play Console Help Describe your issue

Is there any PII or SPII used for the nonce (e.g. user id, user name, phone number, Android ID, SSN, etc)? *

Yes, but hashed or encrypted

Yes, in plain-text

No

How are you validating Play Integrity API responses? *

Server side - by calling Play's server to decrypt and verify

Server side - by decrypting and verifying with self-managed API keys

In my app - by calling Play's server to decrypt and verify

In my app - by decrypting and verifying with self-managed API keys

Other, please specify

How does your app retry in case of Play Integrity API errors? *

No retry

A small number of retry attempts within a short time window

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

Play Console Help

Describe your issue

How will your app act when the Play Integrity API detects risky traffic? *

Please answer with your end goal in mind even if your app is not acting yet. As a reminder, your app should also be able to deal with Play Integrity API errors and the API being unavailable.

Deny access to functionality (for example, users won't be able to log-in). I want unauthorized usage of my app to go down.

Alter or limit specific features (for example, only users on good devices will be allowed on a leaderboard). Overall usage of my app might stay the same.

A mix – deny access for some responses and change features for other responses. I want some unauthorized usage of my app to go down.

No action. I'm only collecting data.

Other, please specify

Quota request - Estimated total queries per day *

10,000 to 1,000,000 (10K to 1M)

1,000,000 to 10,000,000 (1M to 10M)

10,000,000 to 100,000,000 (10M to 100M)

100,000,000 or more (100M+)

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

App integrity | test

play.google.com/console/.../app-integrity/overview

Google Play Console

Search Play Console

App integrity

Play Integrity API Integration not started

App signing Signing by Google Play

Store listing visibility No integrity checks

Play Integrity API Integration not started

Settings Hide

Call the Integrity API at important moments in your app to check that it's your app binary, installed by Google Play, running on a genuine Android device. Your app's backend server can decide what to do next to prevent abuse, unauthorised access and attacks. [Show less](#)

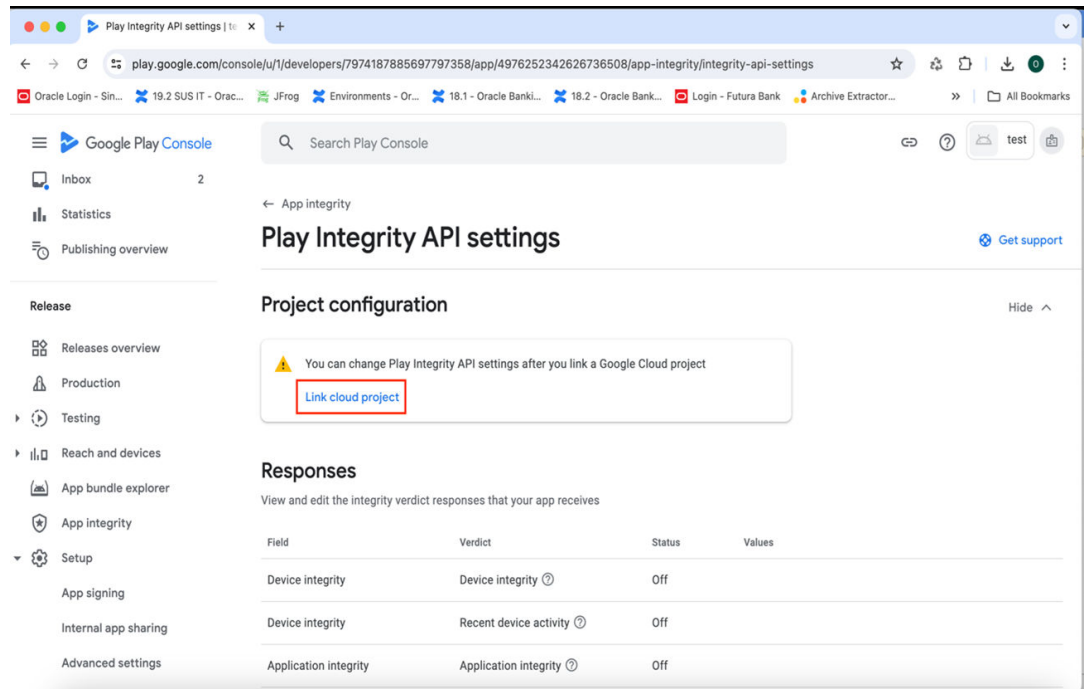
Play Integrity API for Android developers 2 minutes

Play Integrity API setup How to set up your app or game to use the Play Integrity API

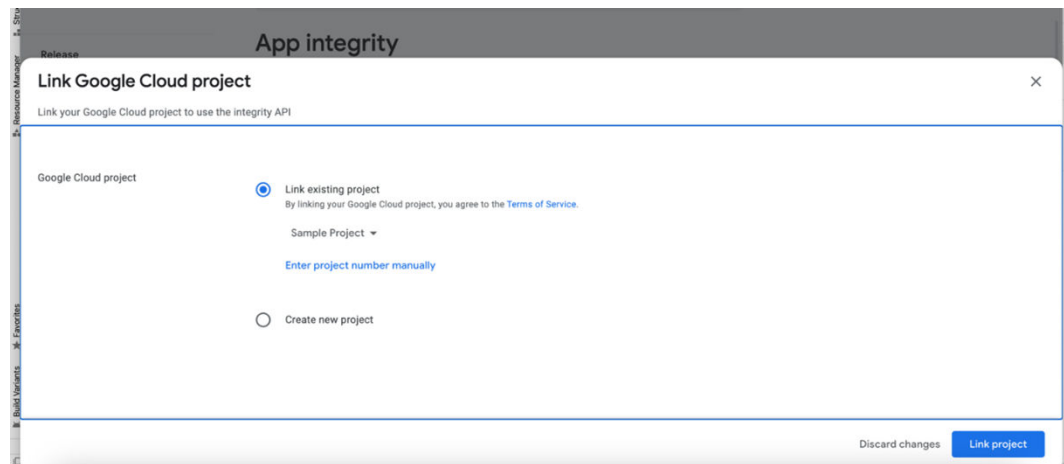
Play Integrity API overview Play Integrity API helps protect your apps and games from risky and fraudulent interactions

https://play.google.com/console/.../app-integrity/integrity-api-settings

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**.

Google Play Console

Search Play Console

Play Integrity API settings

Select actions

Responses

View and edit the integrity verdict responses that your app receives

Field	Verdict	Status	Values
Device integrity	Device integrity ?	On	MEETS_DEVICE_INTEGRITY
Device integrity	Recent device activity ?	Off	
Application integrity	Application integrity ?	On	PLAY_RECOGNIZED, UNRECOGNIZED_VERSION, UNEVALUATED
Account details	App licensing ?	On	LICENSED, UNLICENSED, UNEVALUATED
Environment details	Play Protect status ?	Off	
Environment details	App access risk (beta) ?	Off	

Change responses View JSON sample

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

Change responses

Change the integrity verdict responses that your app receives. Responses are only evaluated if their individual prerequisites are satisfied. For example, most responses require the requesting app to be Play licensed. [Learn more](#)

Device integrity verdicts

- Meets basic device integrity ?
- Meets strong device integrity ?
- Recent device activity
- Device attributes

Environment details verdicts

- Play Protect status

Discard changes Save changes

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

The screenshot shows the Google Play Console interface. On the left is a navigation menu with options like Dashboard, Inbox, Statistics, Publishing overview, Release, Releases overview, Production, Testing, Reach and devices, App bundle explorer, App integrity, Setup, App signing, and Internal app sharing. The main content area is titled 'Play Integrity API settings' and includes a search bar, a 'Get support' link, and sections for 'Testing' and 'Classic requests'. The 'Classic requests' section contains a table with the following data:

Field	Values
Usage tier	Standard
Response encryption	Managed by Google Edit

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';
```

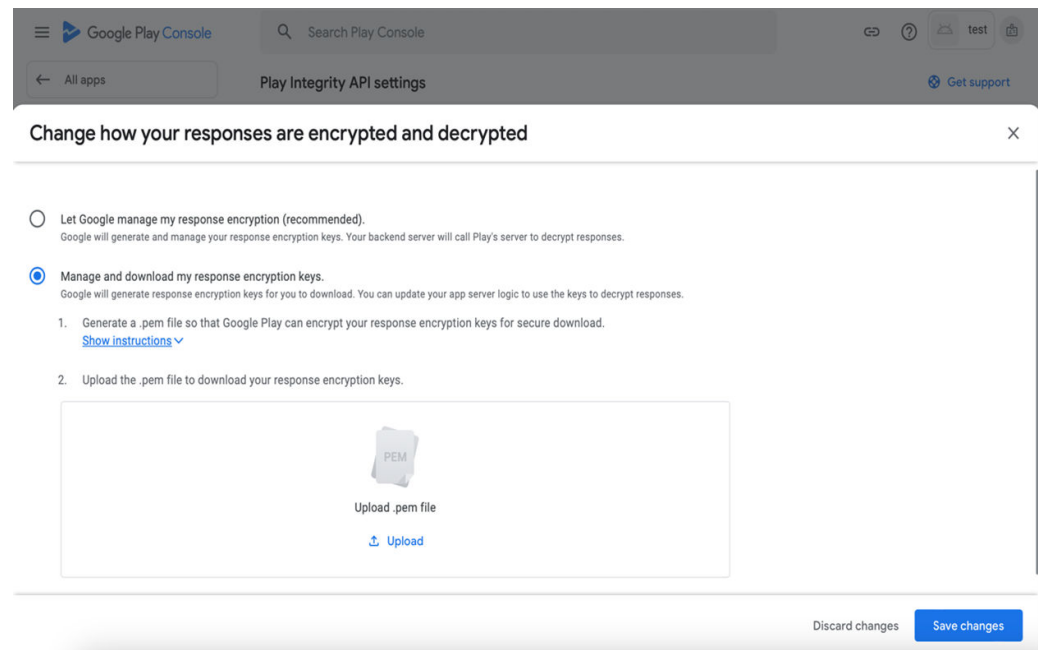
- c. Similarly, Obtain the same keys for authenticator app by using above steps and then maintain those in **DIGX_FW_CONFIG_ALL_B** table corresponding to the following keys respectively:

PLAY_INTEGRITY_ENCRYPTION_KEY_AUTHENTICATOR

and PLAY_INTEGRITY_DECRYPTION_KEY_AUTHENTICATOR

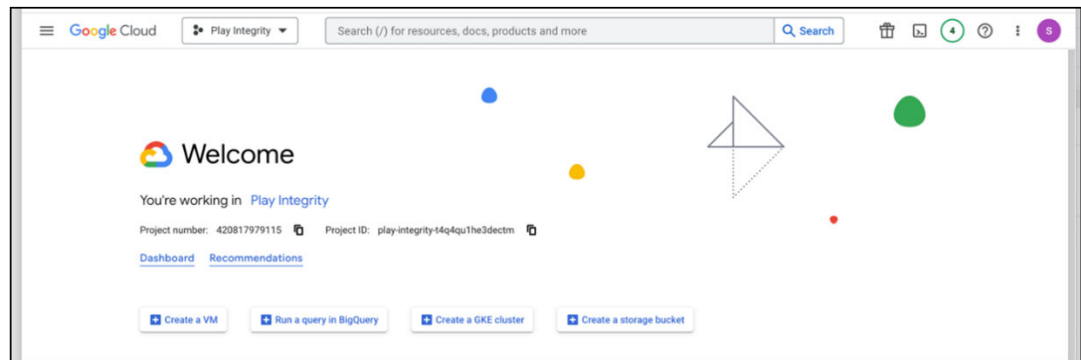
An example query will be:

```
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.



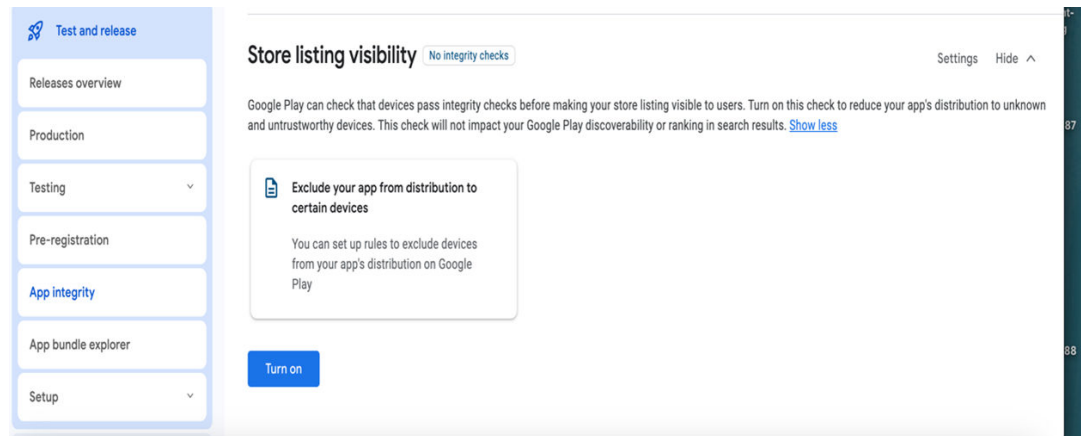
15. Mention the time in seconds to which app can hit the play integrity api. By default it is 300seconds but you can configure as per the requirement. Use below property in RootCheckFlags.java(workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/) long playIntegrityAPICallTime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

The screenshot shows the 'Store listing visibility' settings page in the Google Play Developer console. On the left is a navigation menu with options: Dashboard, Statistics, Publishing overview, Test and release, Monitor and improve, Grow users, and Monetise with Play. The main content area is titled 'Store listing visibility' and features a warning message: 'We will soon take action because your app does not adhere to Google Play Developer Programme Policies. Fix violations as soon as possible.' Below this is a 'Go to Policy status' link. A paragraph of text explains: 'Reduce risk to your app by stopping Play distribution to devices and virtual environments that don't pass system integrity checks, don't meet Android compatibility requirements and may not be licensed to run Google Play services. [Learn more](#)'. The 'Store listing visibility' setting is currently set to 'Strong integrity checks'. The options are: 'No integrity checks', 'Basic integrity checks' (Google Play will check the device meets basic integrity before making your app visible to users), 'Device integrity checks (recommended)' (Google Play will check the device meets device integrity before making your app visible to users), and 'Strong integrity checks' (Google Play will check the device meets strong integrity before making your app visible to users). At the bottom right are 'Discard' and 'Save' buttons.

Note

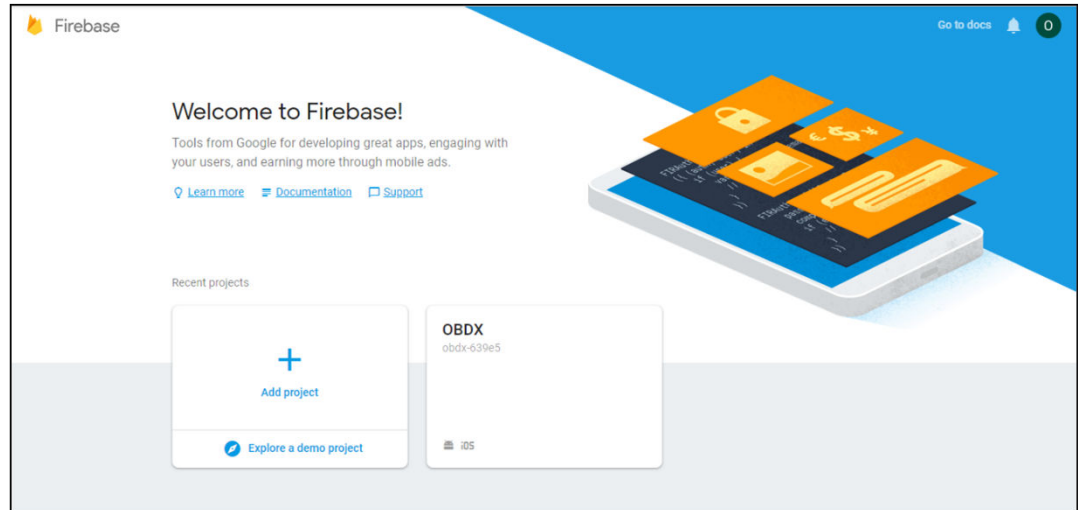
By enabling this setting your app will not be listed on play store of rooted device.

3

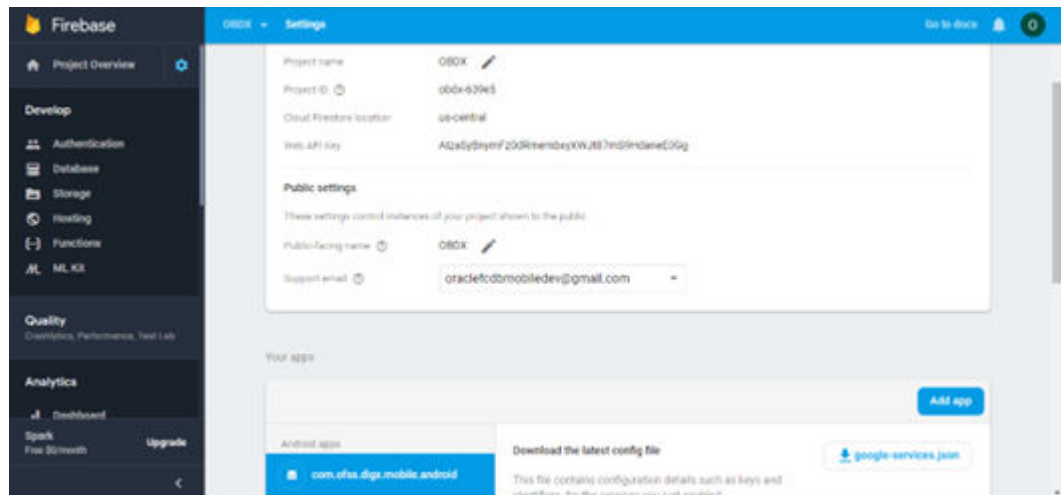
FCM Push Notifications

This topic describes the systematic instruction to **FCM Push Notifications** option.

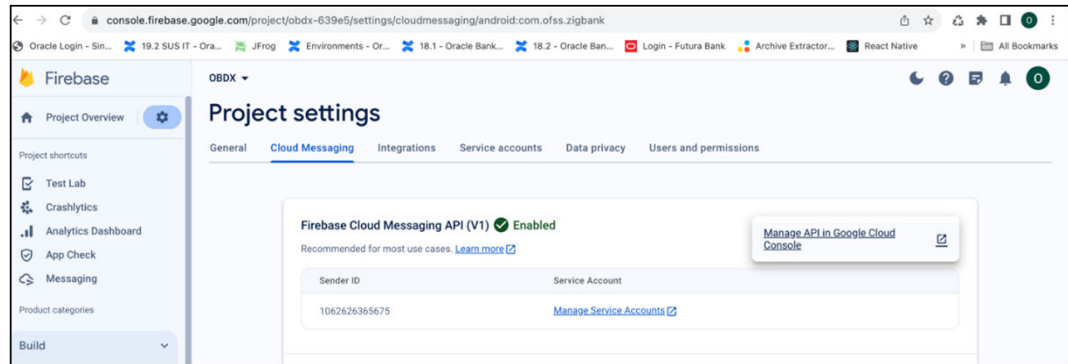
1. Go to URL <https://firebase.google.com/>.
2. Traverse to console and create a project.



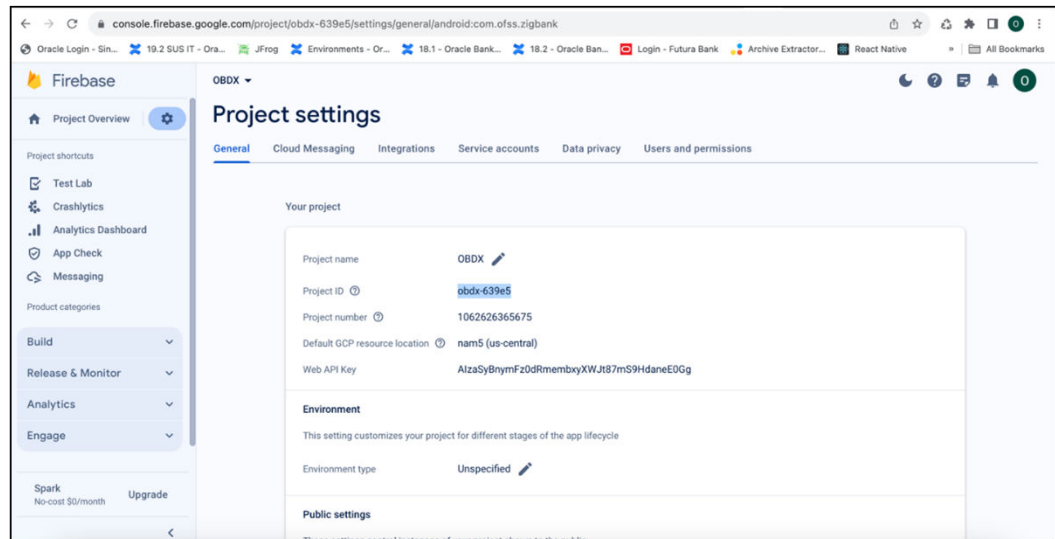
3. Download `google-services.json` from below page and save to (`zigbank\platforms\android\app`) directory.
4. Remember to keep the projects package name and firebase package name same.



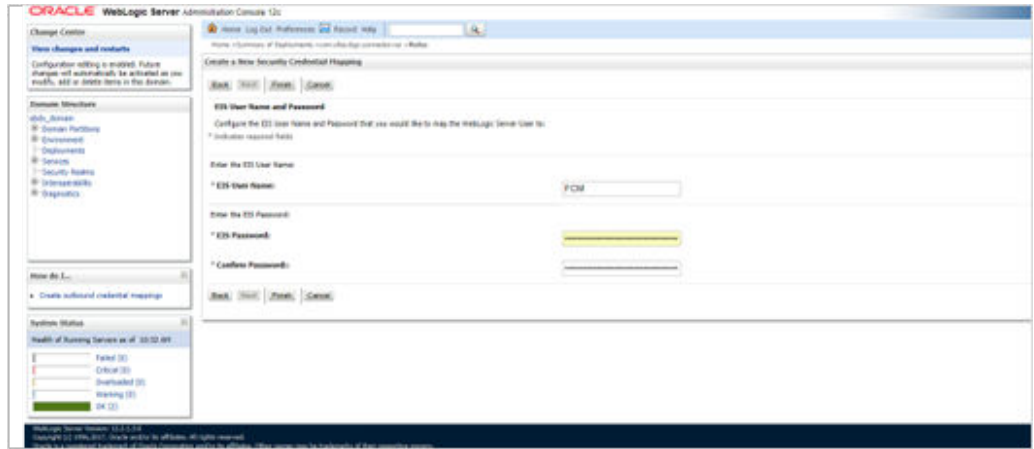
5. Traverse to cloud messaging tab Enable Firebase Cloud Messaging API(V1) by clicking on Manage API in Google Cloud Console.



6. Get the Project ID from Project Setting in Firebase console.



7. Update FCM URL in system configuration (Refer Section 10) by adding YOUR_PROJECT_ID in url which is captured on above step.
8. If proxy address is to be used, provide the same in proxy settings at System Configurations.
9. Generate private key for your service account by using below steps-
 - a. In the Firebase console, open **Settings > Service Accounts**.
 - b. Click **Generate New Private Key**, then confirm by clicking **Generate Key**.
 - c. You can also follow below google doc - <https://firebase.google.com/docs/cloud-messaging/auth-server#provide-credentials-manually>.
 - d. Update the FCM properties which are mentioned in System Configurations section 10.
 - e. If CONNECTOR is selected in Step 2 update password as below:

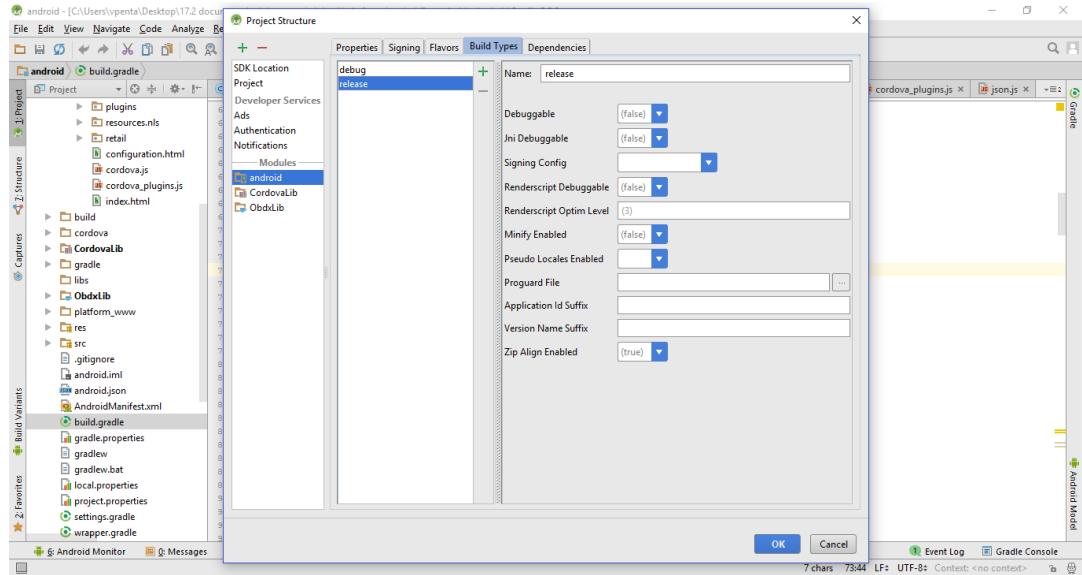


4

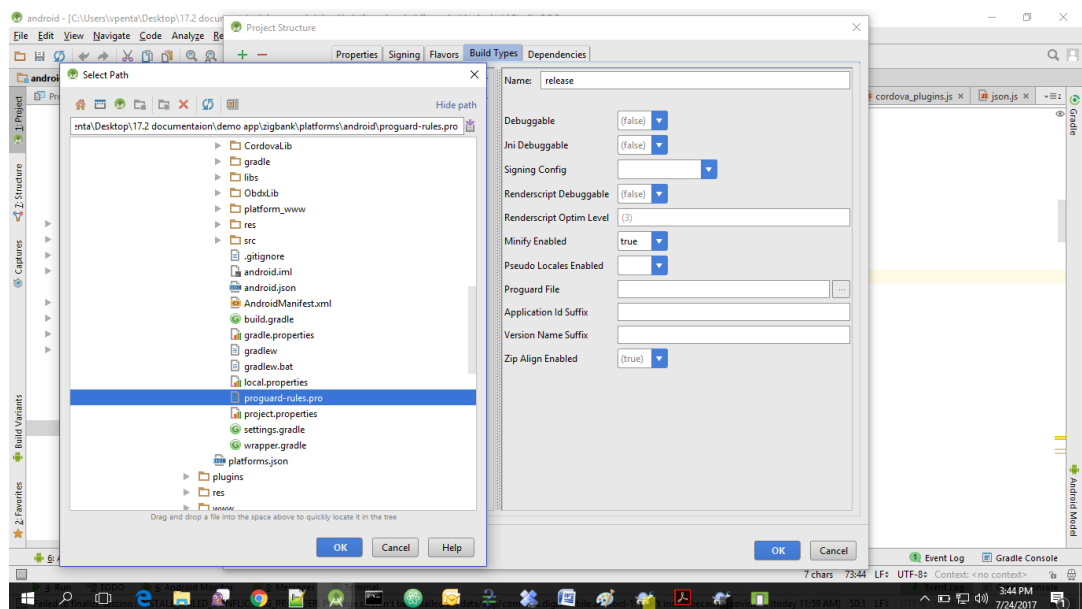
Build Release Artifacts

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>
```
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.oc9gdev.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.

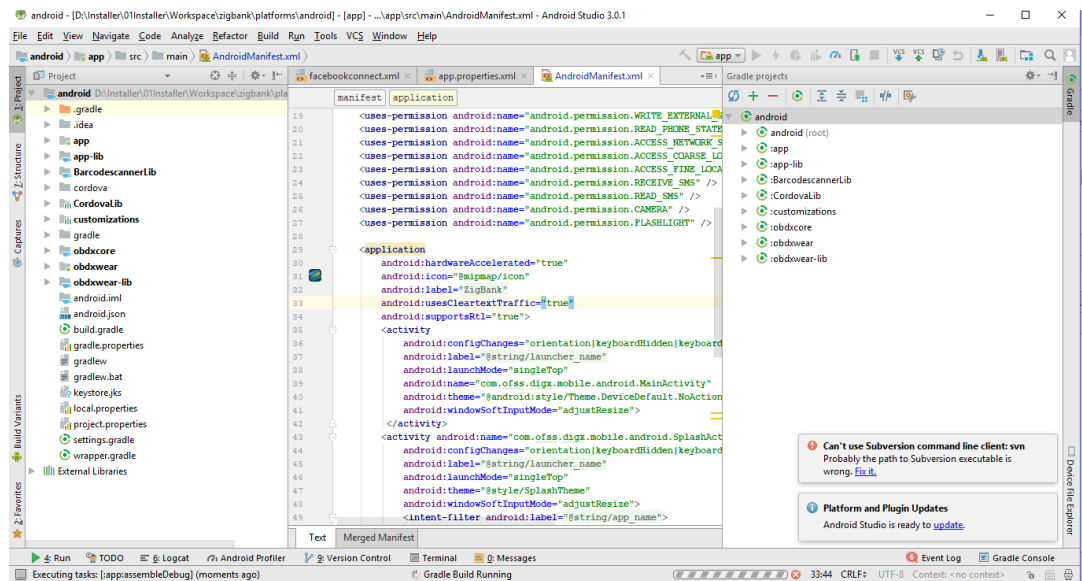
10. Domain Based Setup (This is same for OBDX servicing App and Authenticator App). To use domain based setup, enable below flag in app.properties file -

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

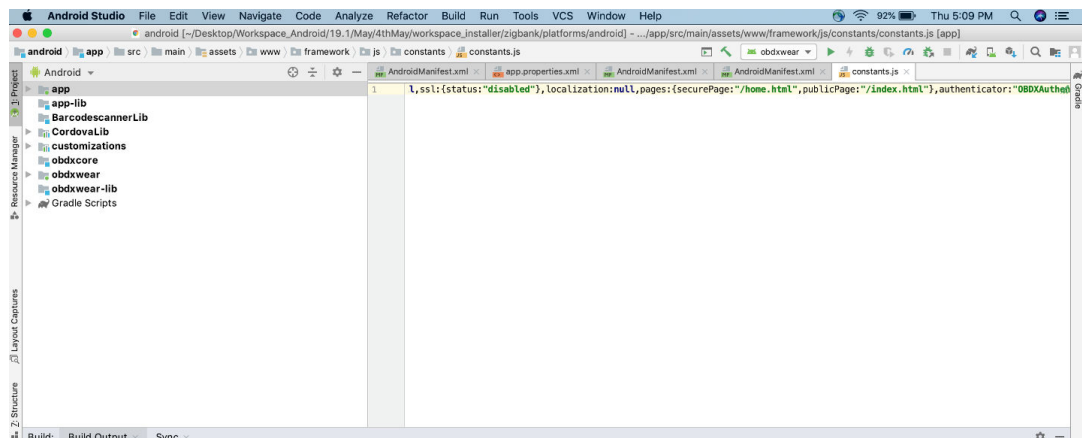
11. Adding chatbot support to mobile application (Optional).

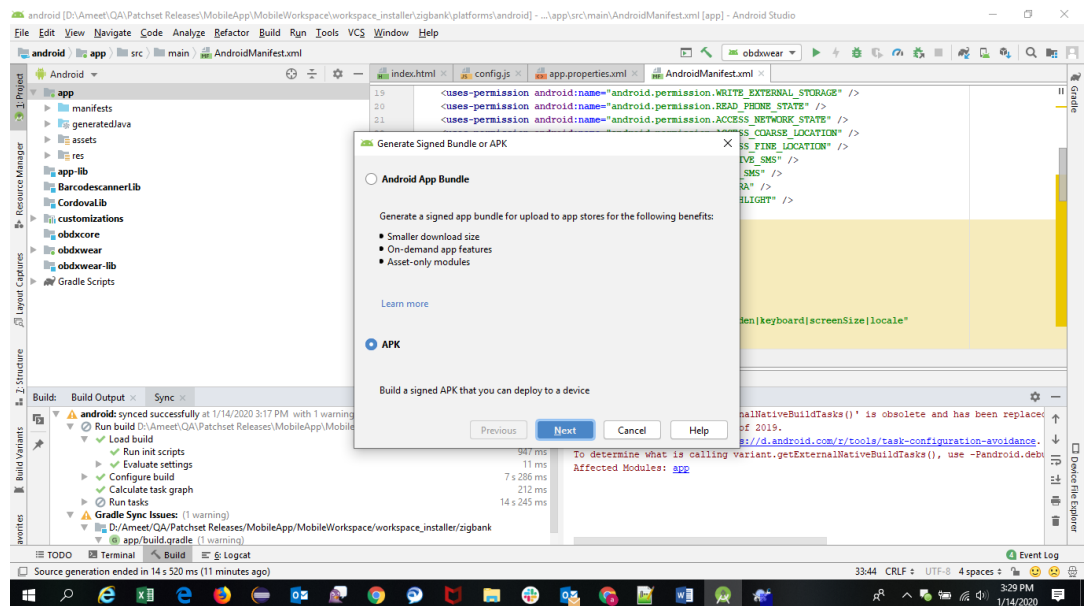
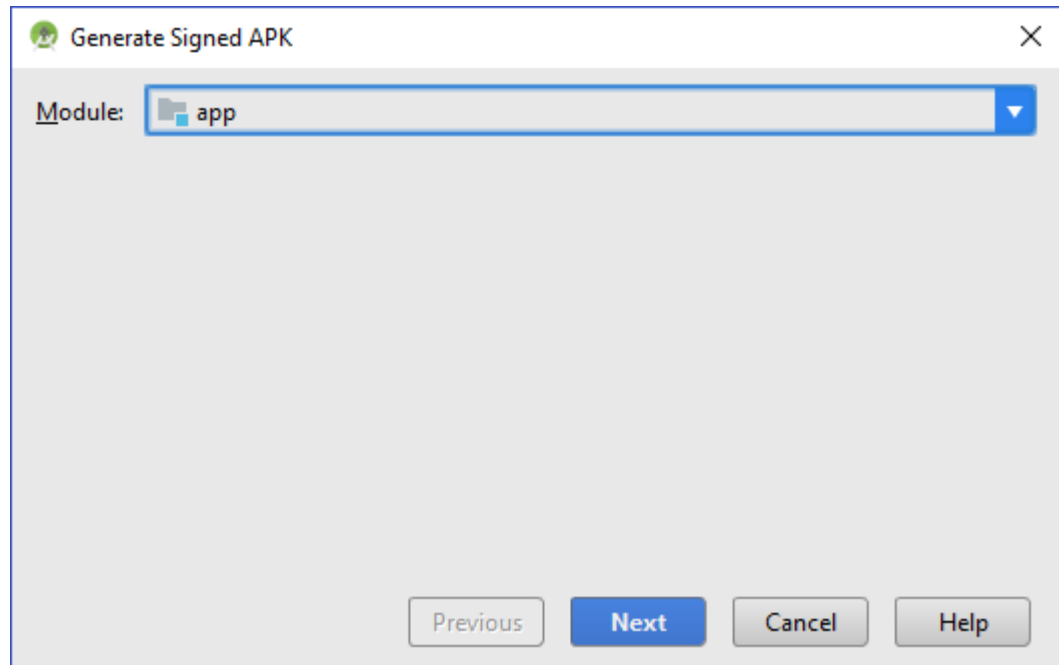
CHATBOT_ID	The tenant ID
CHATBOT_URL	The URL for the ChatApp application in ODA

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

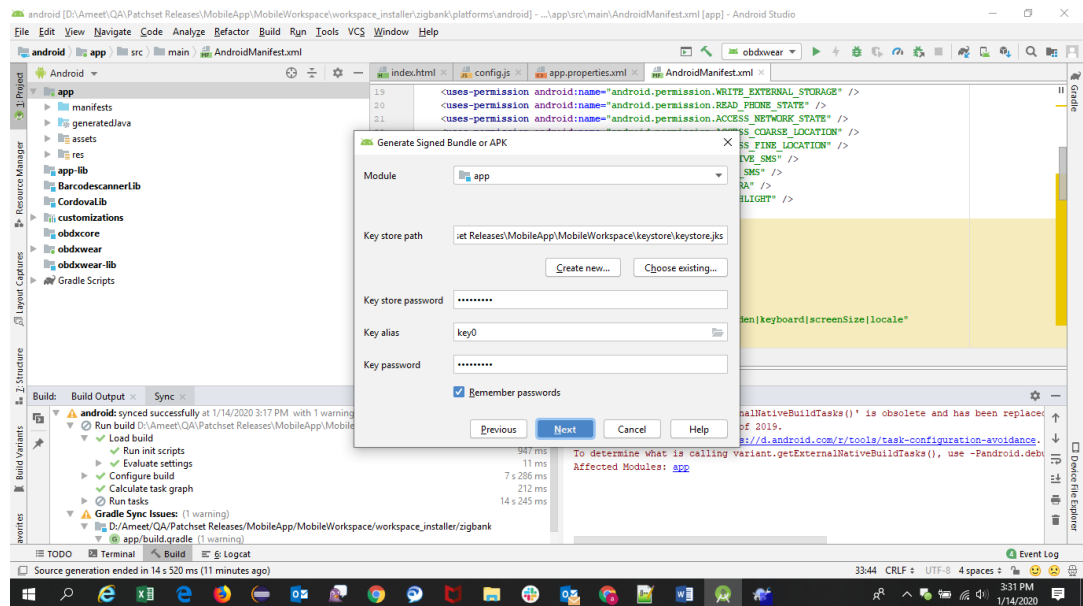
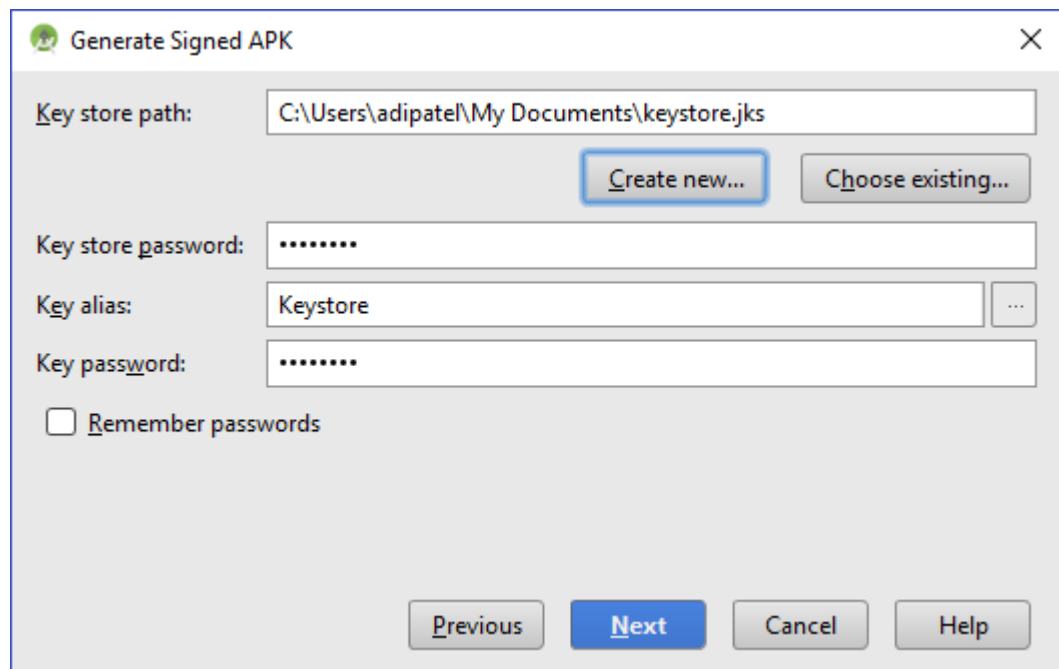


13. For Generating Signed Apk: To Generate release-signed apk as follows: On menu bar click on Build → Generate Signed Apk.

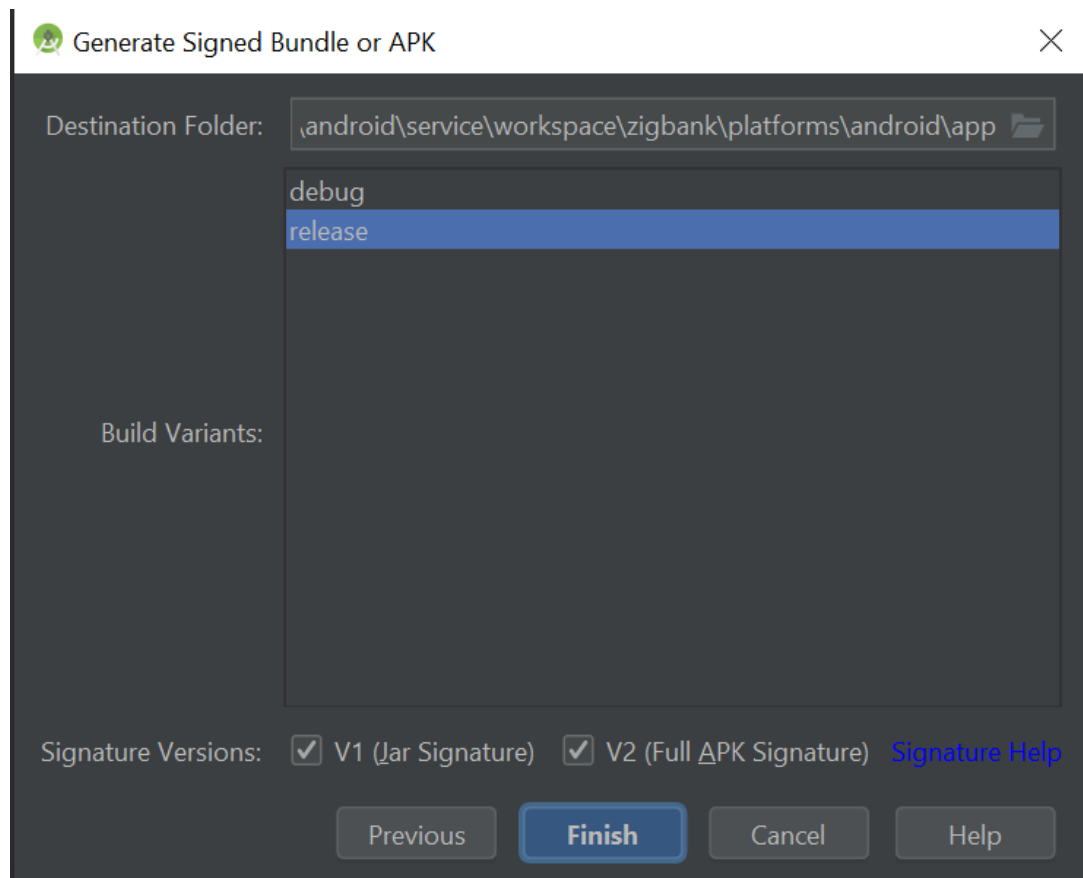




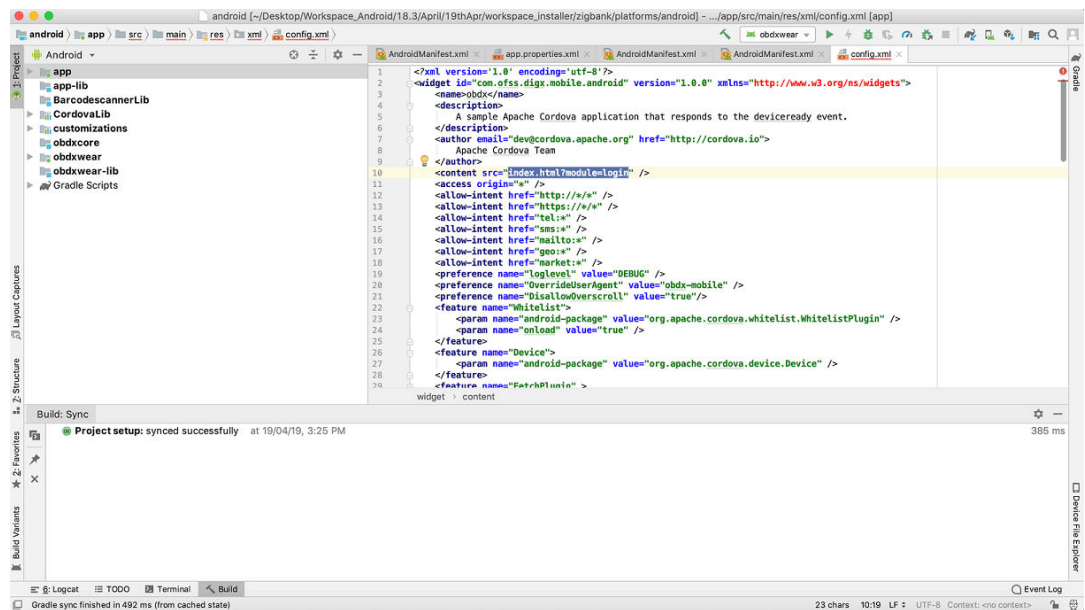
14. If you have an existing keystore.jks file then select choose **Existing** else click **Create New**.



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 APK by 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 from OBDX_Patch_Mobile\authenticator\TOKEN-BASED

5.1.2 Using Un-built UI

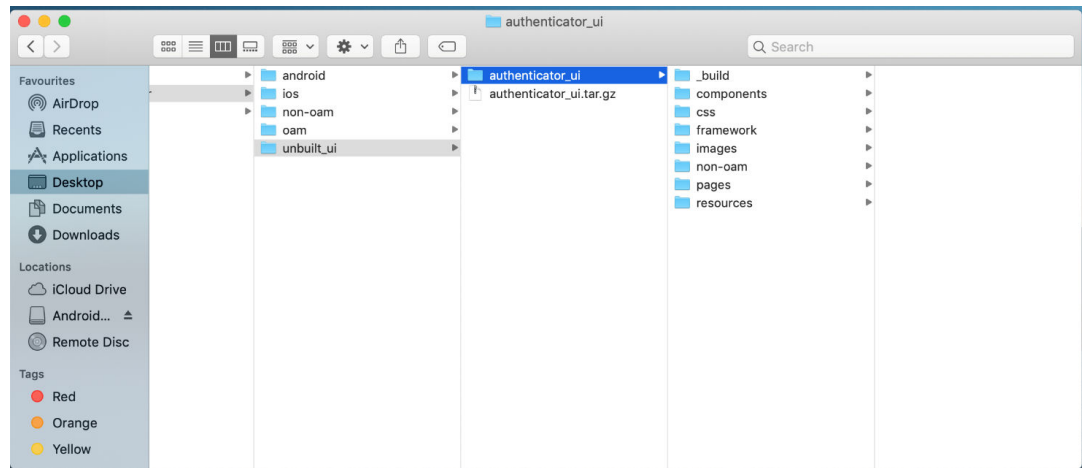
This topic provides information on **Using Un-built UI**.

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.

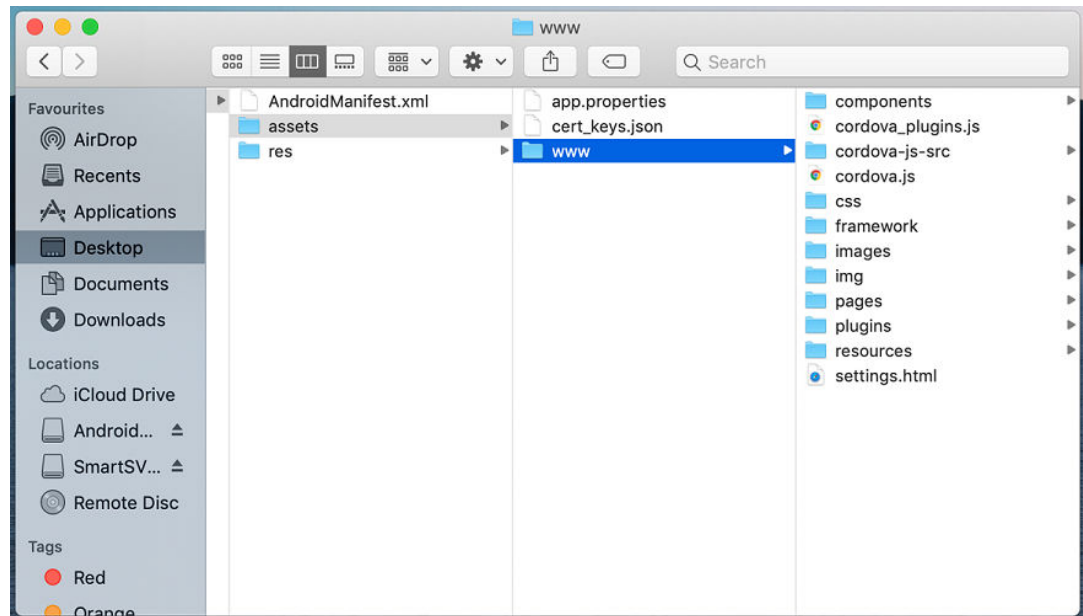
1. Extract authenticator_ui.tar.gz from OBDX_Patch_Mobile\authenticator\unbuilt_ui. The folder structure is as shown :



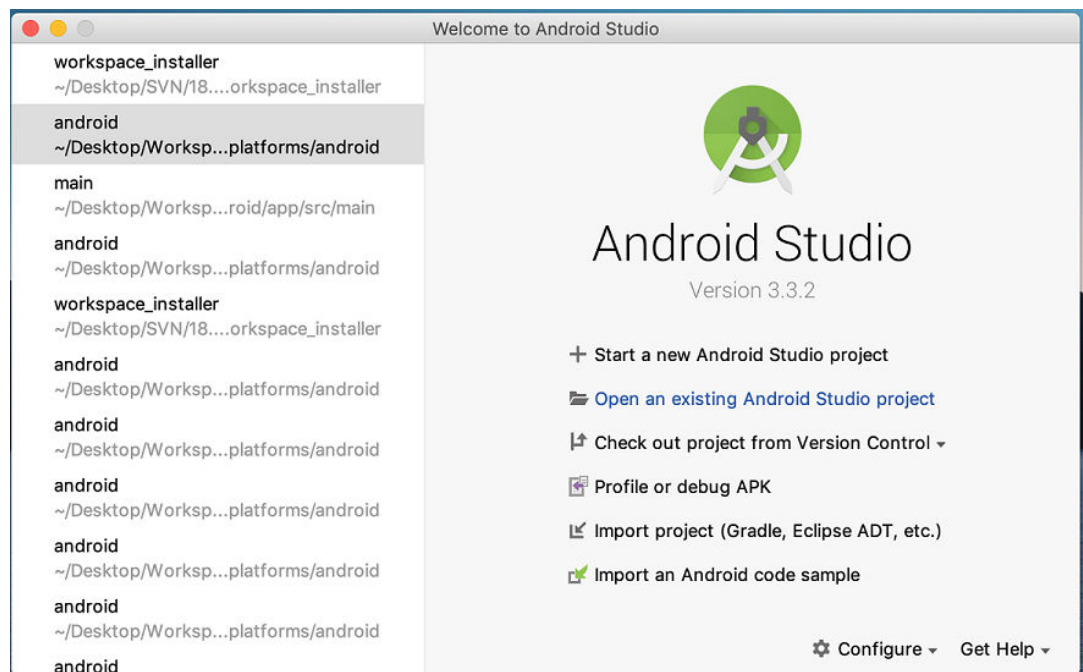
5.2 Authenticator Application Workspace Setup

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

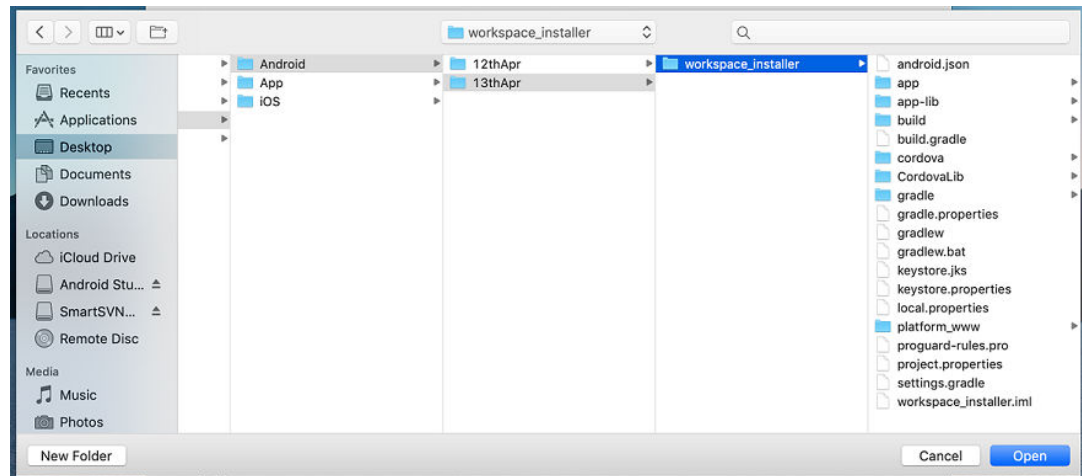
1. Copy UI (Directories – components, css, framework, images, pages, resources) from /dist directory to workspace/installer/app/src/main/assets/www/ In case any popup appears, click **Replace**.



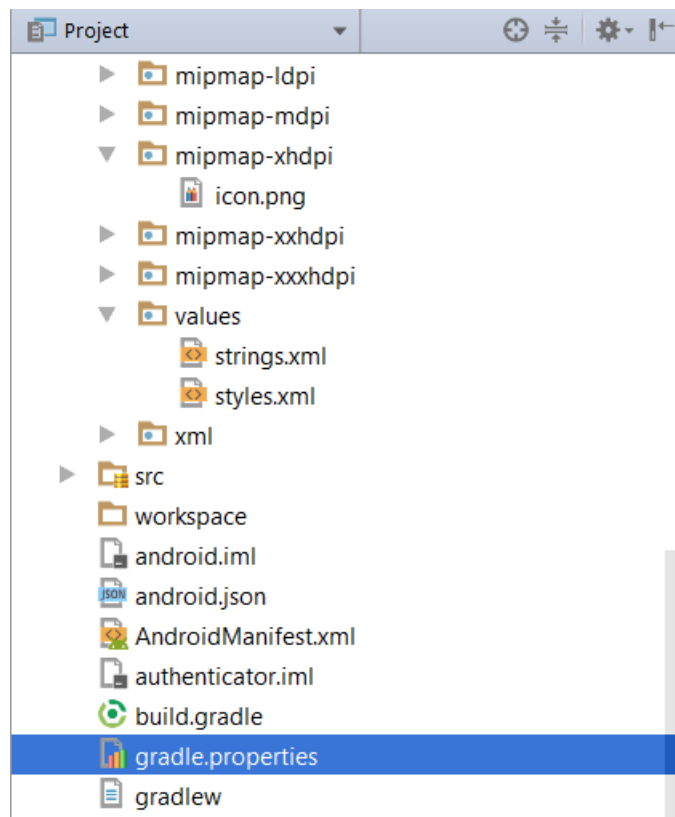
2. Launch Android Studio and open existing project.



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.

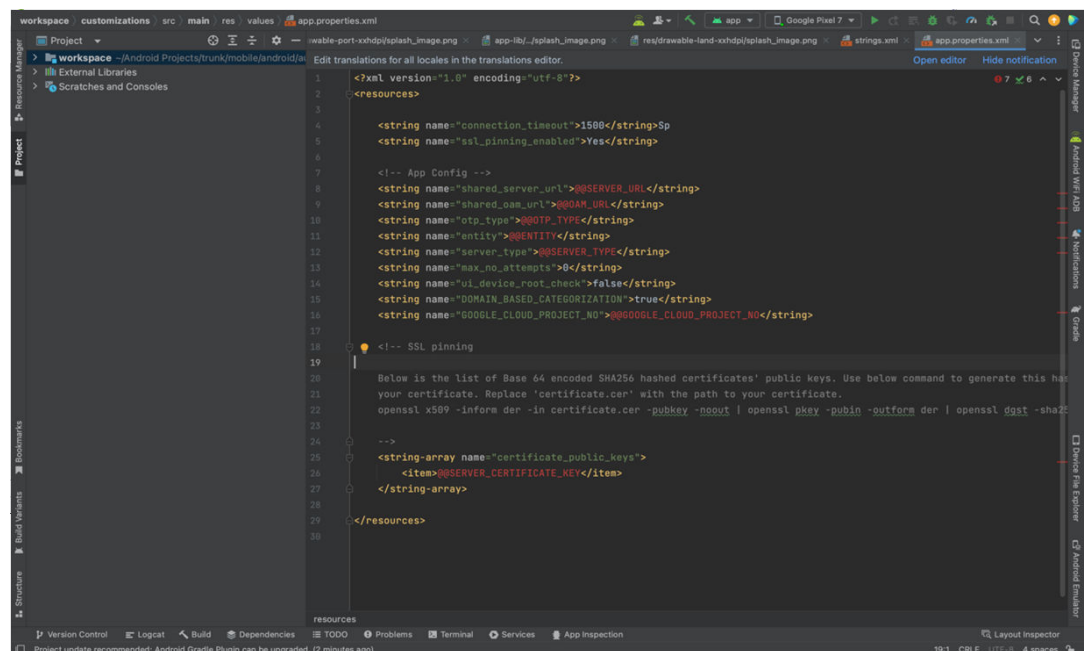
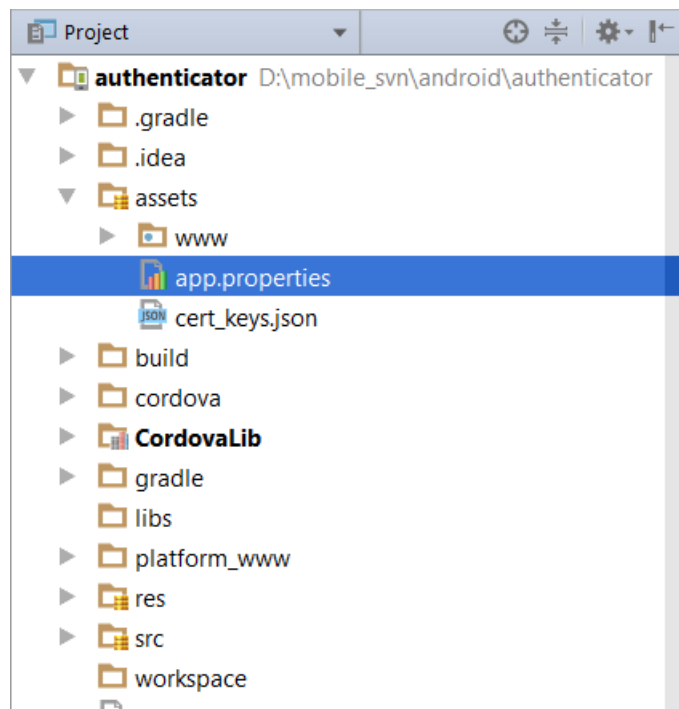


```

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

```

5. Open **“assetslapp.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

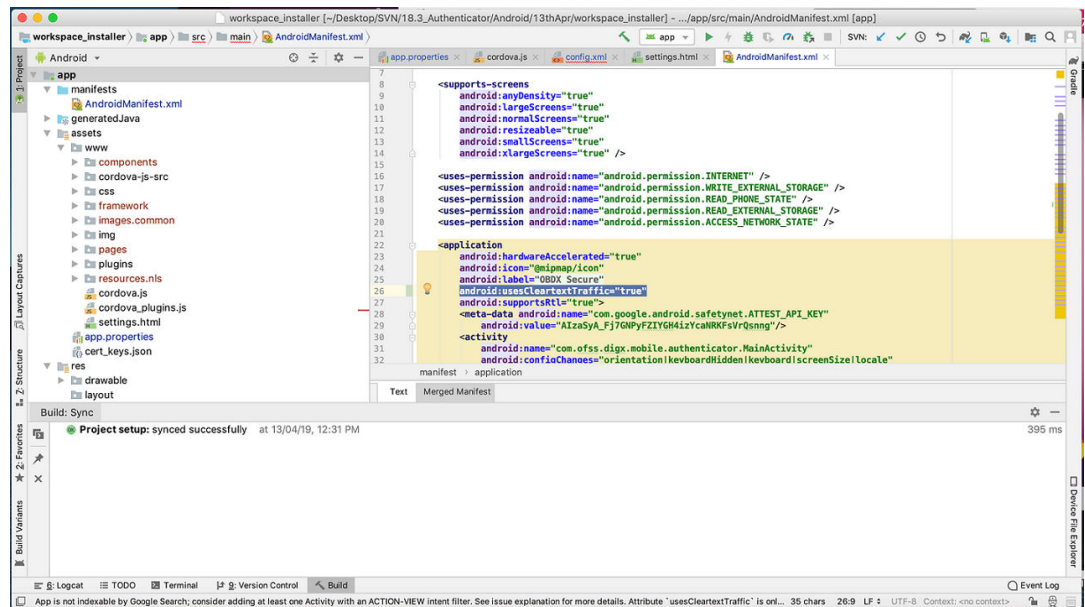
Set MAX No Attempts greater than 0

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

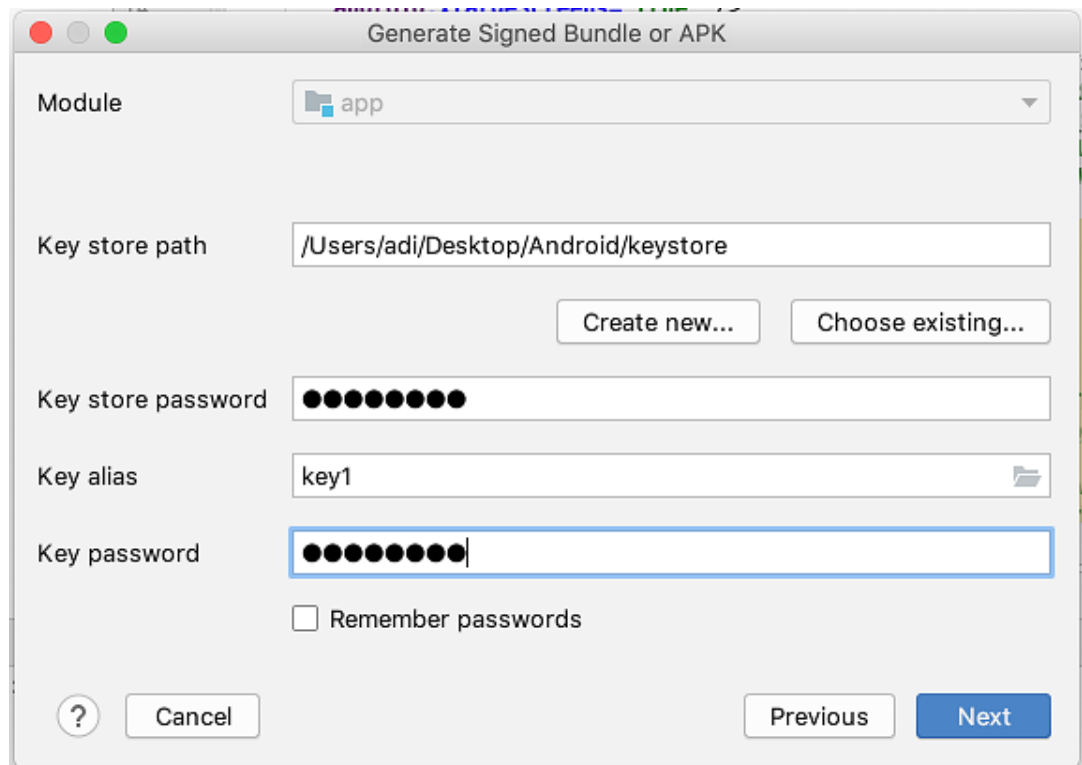
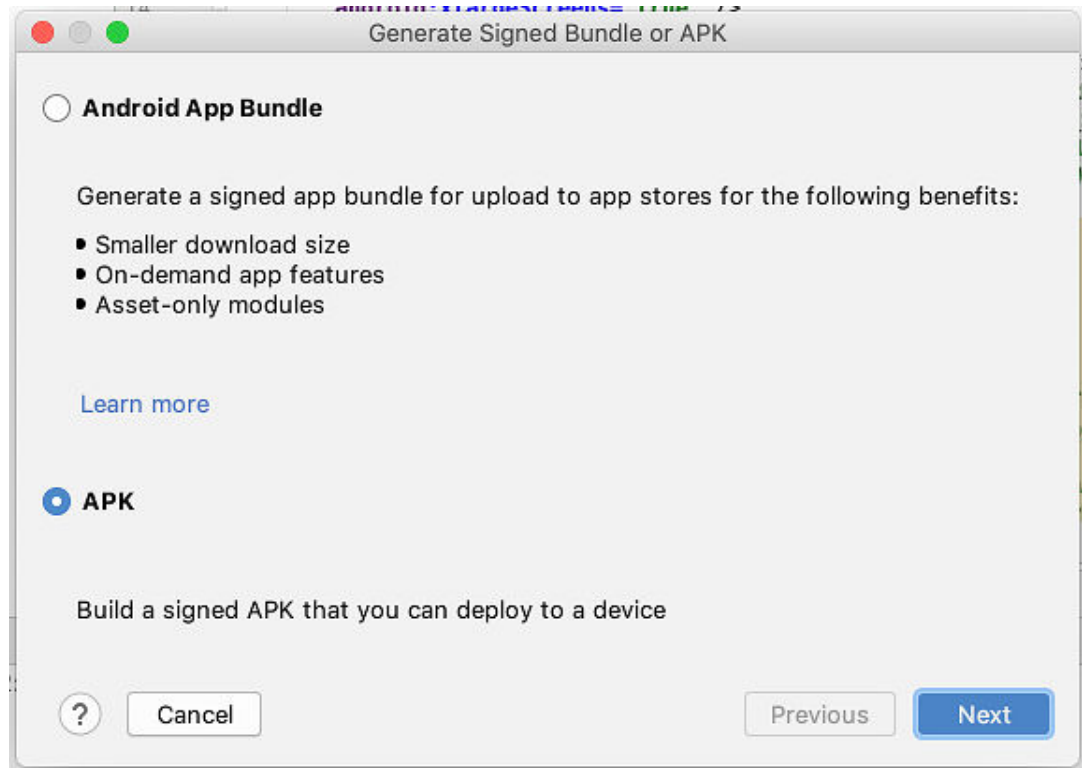
Note

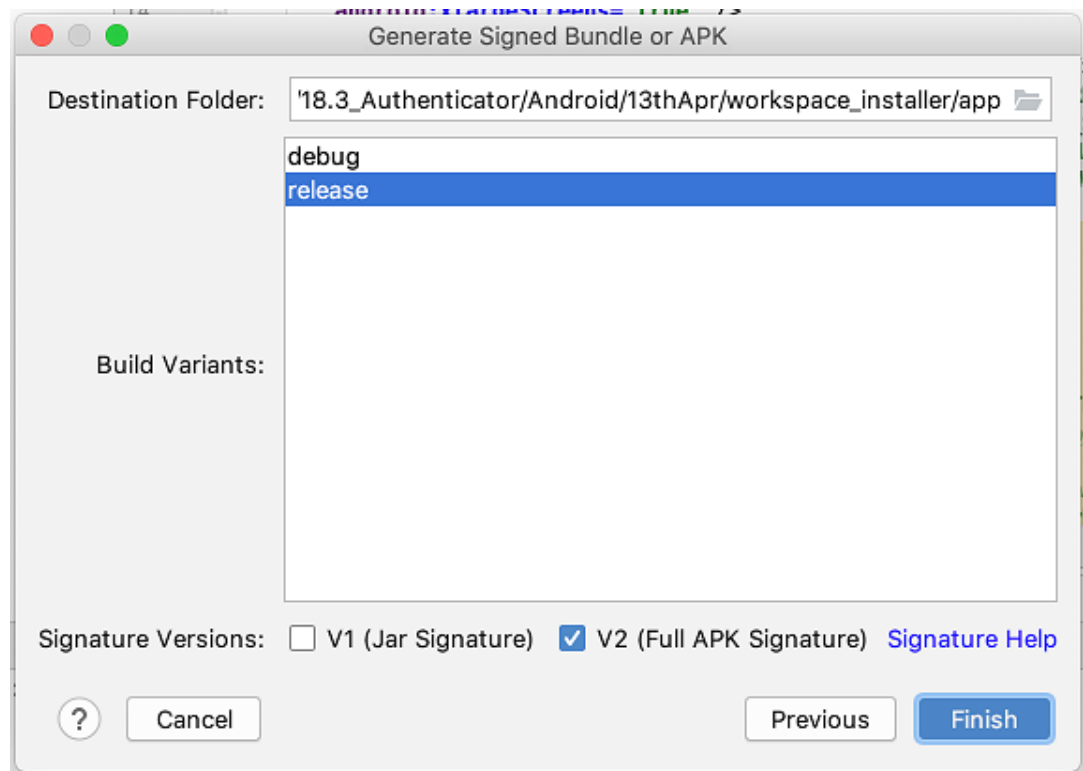
If selected authentication mechanism is not OAM based then remove `shared_oam_url` property.

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



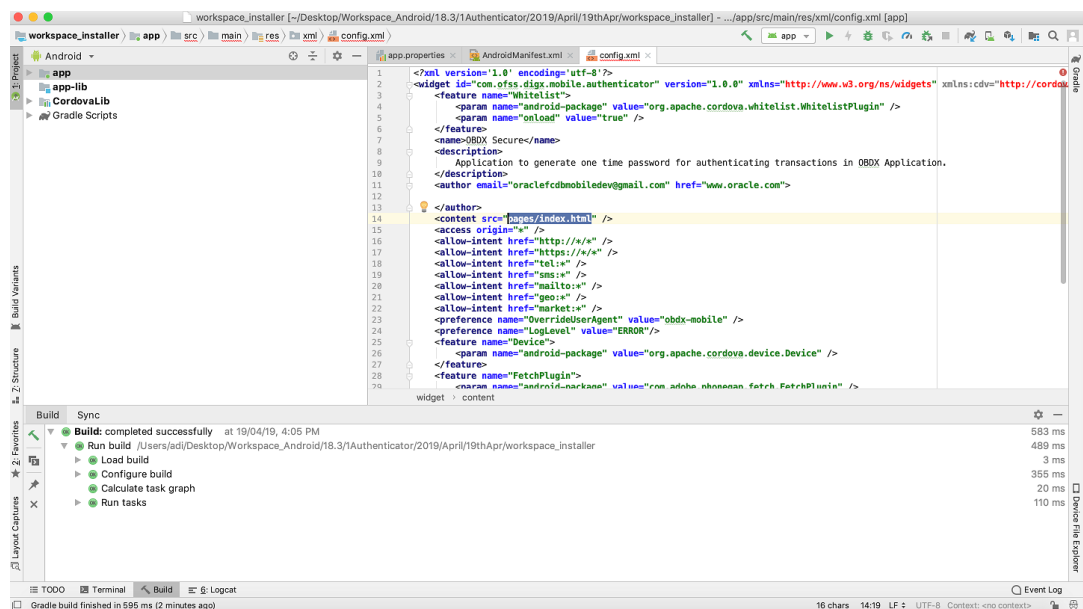
9. **For Generating Signed Apk:** To Generate release-signed apk as follows:
10. On menu bar click on Build → Generate Signed Apk.





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.pr
operties.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>sha256/3rgsgghoqrDegekpkkgk92Fgw1w7exyYCS1okef90o1w=</item>
</string-array>
```


7

Adding Custom Cordova Plugin

This topic provides information on **Adding Custom Cordova Plugin**.

Step 1 -

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

Create java file under your package which will extend CordovaPlugin.

Override execute method with JSONArray as a parameter.

Retrieve JSONObject from JSONArray and get the data which is 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

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.

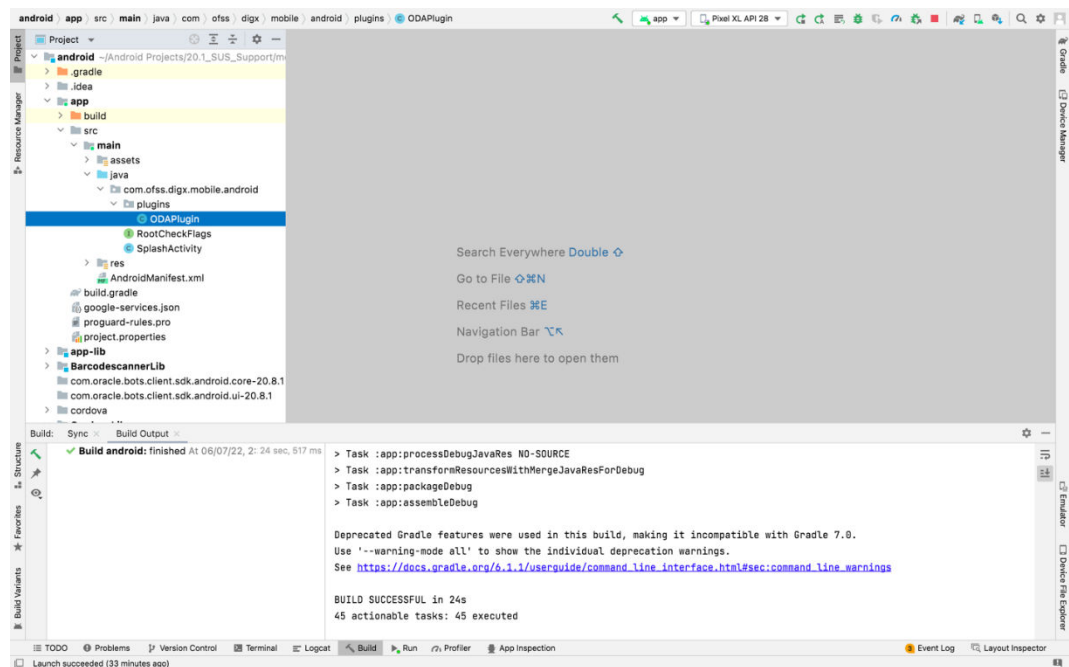
8

ODA Chatbot Inclusion

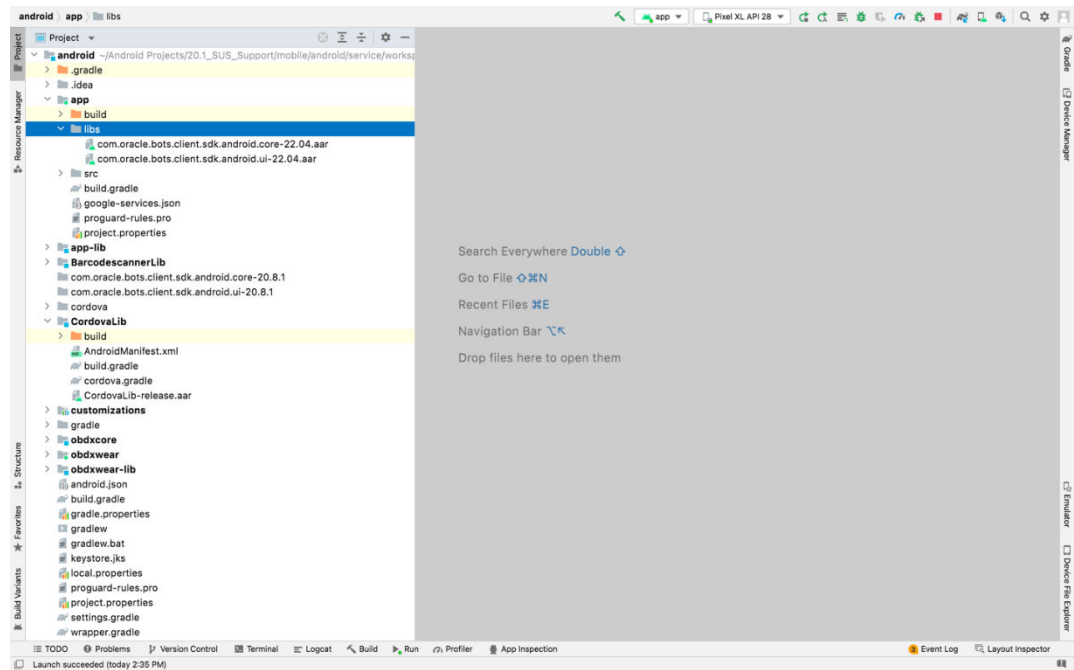
This topic describes the systematic instruction to **ODA Chatbot Inclusion** option.

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

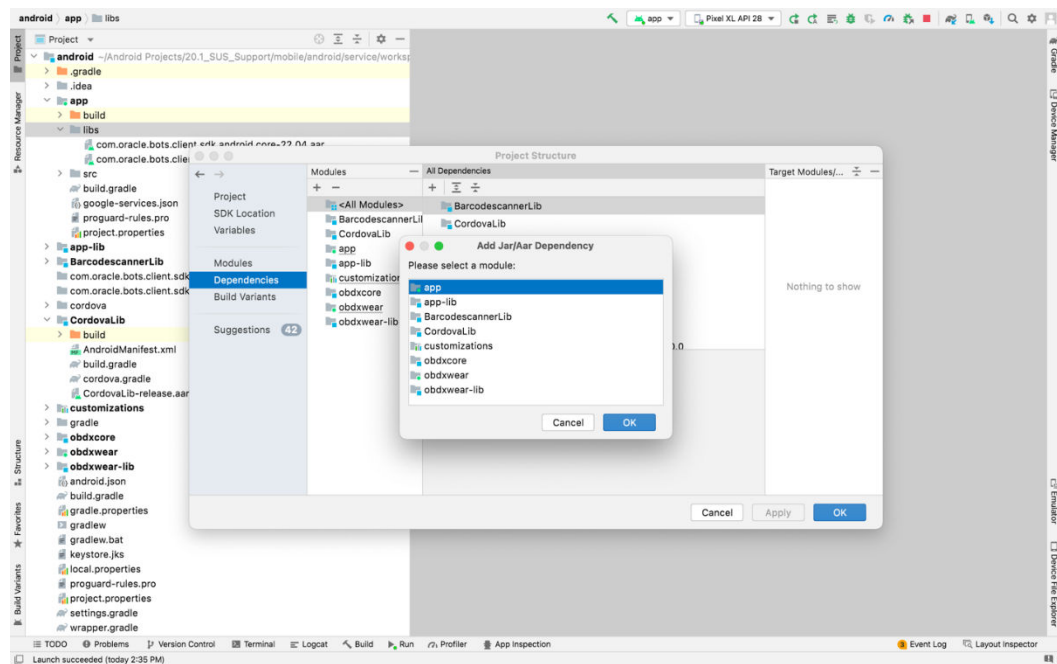
1. Copy ODAPugin.java from workspace_installer/AppExtension/oda to workspace_installer/zigbank/platforms/android/app/src/main/java/com/ofss/digx/mobile/android/plugins/



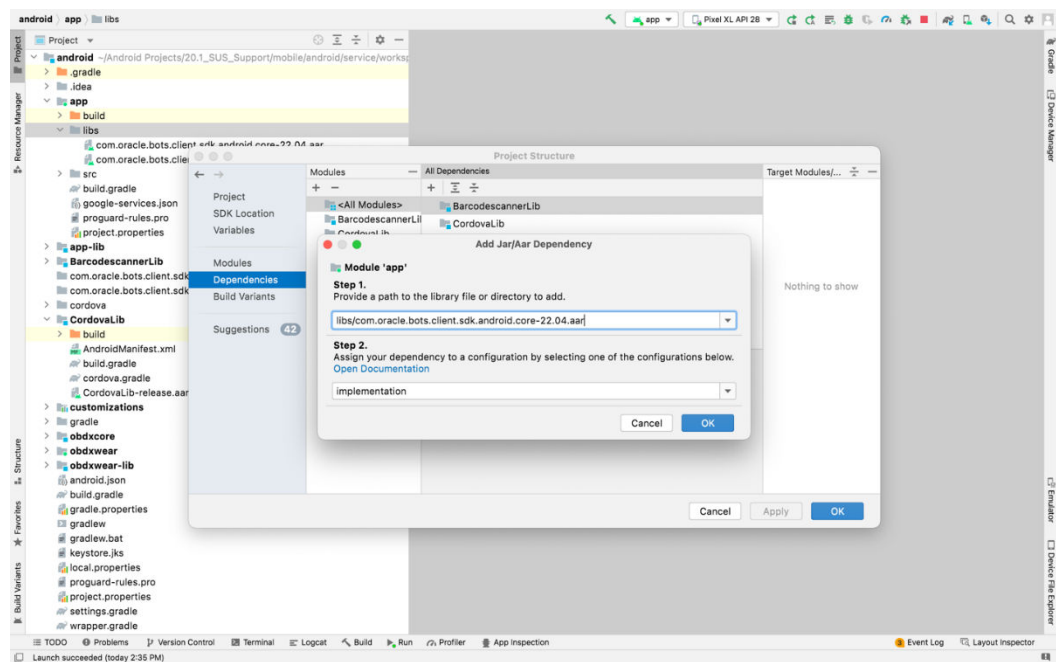
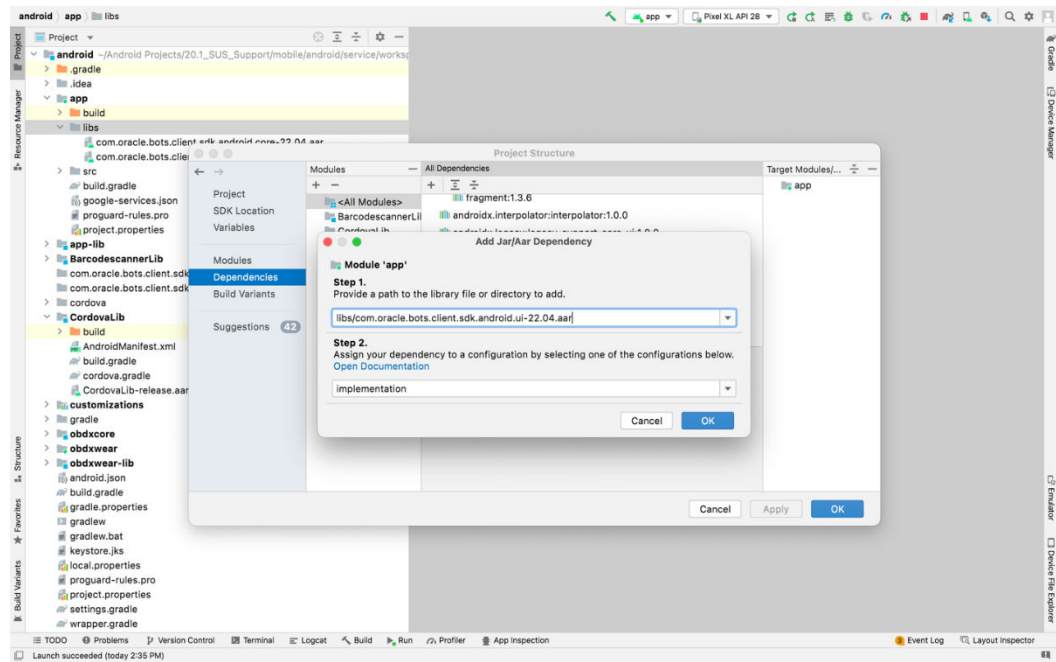
2. Download ODA Android sdk from below link-<https://www.oracle.com/downloads/cloud/amce-downloads.html>.
3. Add libs folder at zigbank\platforms\android\app and copy below files from downloaded sdk folder in it.
 - a. com.oracle.bots.client.sdk.android.core-xx.aar.
 - b. com.oracle.bots.client.sdk.android.ui-xx.aar.



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.

10

System Configuration

This topic provides information on **System Configuration**.

Select the appropriate entity in System Configuration page on admin login.

Table 10-1 System Configuration

Property Name	Entity Config Name	Description
FCM Server URL	Event Management	FCM Server URL for Push Notification.
FCM Key Store	Event Management	Specifies whether to pick server key from database or from connector. Default DB (No change).
FCM Authorization Key	Event Management	FCM Private key for push notification (Service account json file content).
Proxy Settings	Entity / Event Management	Proxy settings to allow APNS connection from OBDX server. The format is: HTTP,<proxy>,<proxy port>, userId, password.
Android Service Application Package Name	Framework/Mobile	Package name of service application.
Android Soft Token Application Package Name	Framework/Mobile	Package name of soft token application.
Play Integrity Encryption Key for Android Service Application	Framework/Mobile	Play integrity encryption key for service app.
Play Integrity Encryption Key for Android Soft Token Application	Framework/Mobile	Play integrity encryption key for soft token app.
Play Integrity Decryption Key for Android Service Application	Framework/Mobile	Play integrity decryption key for service app.
Play Integrity Decryption Key for Android Soft Token Application	Framework/Mobile	Play integrity decryption key for Soft Token app.
Play Integrity License Check For Service Application	Global Settings/ Framework/Mobile	This property allows to enable license checks in Google play integrity to verify whether application is downloaded from Google Store. If this is enabled, application will check installation source and will display error in case the application is downloaded from external and unverified source.
Mobile Application Biometric Token Expiration Time	Global Settings/ Framework	This property allows to set token expiration time which was issued at the time of biometric registration
Snapshot Token Expiration Time	Global Settings / Framework	It is the expiration time of the token generated for snapshot in mobile application.
Wearable Application Token Expiration Time	Global Settings / Framework	Expiration time of the token generated for wearable in mobile application.

Table 10-1 (Cont.) System Configuration

Property Name	Entity Config Name	Description
Chatbot Authentication Type	Global Settings / Security Management System	LOCAL/R_SOFT/T_SOFT/PIN
Allowed Device Count for Biometric Registration	Global Settings / Framework	Set to 1 or 100 based on bank's requirement to allow number of devices or biometric registration
Allowed Window Size for TOTP Token	Global Settings / Framework	Base value is 6, Recommended is 6 or 4. Based on these settings the window size of the generated tokens will vary.

11

Pre-requisites for Production App

This topic provides information on **Pre-requisites for Production App**.

1. Enable all security configs like Play Integrity, SSL.
2. Please share release app only for security testing, as code obfuscation will be applied on release mode only.
3. Enable force app update flag.
4. Server URL must be pointed to production environment.
5. Add SSL certificate public keys for root and intermediate certificates for SSL pinning check.
6. Check all configuration for playintegrity & SSL.

Index

A

Adding Custom Cordova Plugin, [1](#)
App Update Manager, [12](#)
Application Security Configuration, [1](#)
Authenticator Application Workspace Setup, [2](#)
Authenticator UI (Follow any one step below), [1](#)

B

Build Release Artifacts, [1](#)
Building UI Manually, [2](#)

C

Create project using Remote UI, [3](#)

D

Deeplinking - To open reset password, claim money links with the application, [8](#)
Device Registration and Push Registration Functionality, [9](#)
Displaying Rate Option to Redirect to Playstore Page, [11](#)

E

Enabling Force Update, [11](#)

G

Google Play Integrity, [1](#)

I

Importing in Android Studio, [3](#)

L

Location Tracking Metrics, [10](#)

O

ODA Chatbot Inclusion, [1](#)

P

Passkey (Passwordless login), [5](#)
Pre-requisites for Production App, [1](#)
Prerequisites, [1](#)
Push Notification 2FA configuration, [1](#)

S

Scan to Pay from Application Icon, [5](#)
Splash Screen Migration, [11](#)
System Configuration, [1](#)

U

Using built UI, [1](#)
Using Un-built UI, [1](#)

W

Widget Functionality, [4](#)