Mobile Application Builder Guide-iOS Guide
Oracle Banking Digital Experience
Patchset Release 22.2.2.0.0

Part No. F72987-01

December 2023

ORACLE

Mobile Application Builder Guide-iOS Guide

December 2023

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax:+91 22 6718 3001

www.oracle.com/financialservices/

Table of Contents

1. Preface

1.1 Intended Audience

This document is intended for the following audience:

- Customers
- Partners

1.2 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.

1.3 Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.4 Structure

This manual is organized into the following categories:

Preface gives information on the intended audience. It also describes the overall structure of the User Manual.

The subsequent chapters describes following details:

- Introduction
- Preferences & Database
- Configuration / Installation.

1.5 Related Information Sources

For more information on Oracle Banking Digital Experience Patchset Release 22.2.2.0.0, refer to the following documents:

- Oracle Banking Digital Experience Installation Manuals

2. OBDX Servicing Application

This is the main application for mobile banking.

2.1 Prerequisite

- Download and Install node js as it is required to run npm and cordova commands.
- Latest XCode to be download from Mac App Store. This document is w.r.t to XCode 15.0
- OBDX iOS Application is supported only on current iOS version and only one version preceding that.

2.2 Create Project

Ensure **Nodejs Version is >= 12 and latest XCode version available on AppStore is used.**

1. Extract iOS workspace from installer and place in a folder.
2. The workspace contains framework for running on devices and simulator both. The same frameworks within the workspace can we used to run on simulator and device as well. There is no need to copy the frameworks from simulator folder to workspace for this version.
3. Below are the frameworks present inside the workspace. Verify if these are present before running the application on device or simulator.
    a. OBDXFramework.xcframework
    b. CordovaFramework.xcframework
    c. OBDXExtensions.xcframework
    d. OBDXWatchFramework.xcframework
4. To run the application, we also need UI for the application. UI can be hosted on the remote server or hosted on local machine or saved locally in the mobile workspace. The UI pointing to remote server, will be served pointing to the remote server URL. UI hosted on local machine will be served using localhost URL. Refer below sections to include the UI.
5. Also, apple certificates and provisioning profiles are needed to run the application on devices. Refer Section 2.10 for more details.
6. Also, refer section 2.6 for configurations required for the application.

2.3 Create Project Using Remote UI

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

  Proceed to **2.6 Configurations for the application.**

2.4 Create Project Using Local UI by running on local machine or local server.

Use any 1 option below of a/b:

a. 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 host 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. For 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:

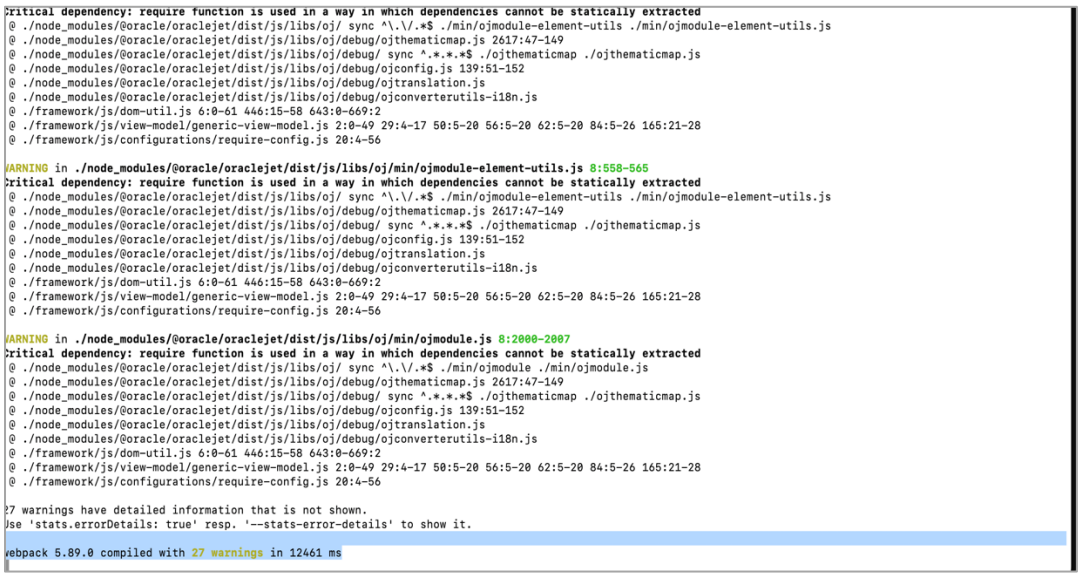
   ```
   devServer: {
       static: path.join(__dirname,
           "../../dist"),
       compress: true,
   port: 4000,
   hot: false,
   client: false,
   headers: {
           "Access-Control-Allow-Origin": "*"
   },
   ```

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 reload logic
   //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.

b. Using built UI (out of box shipped with installer)

Available at --

`OBDX_Installer/installables/ui/deploy` (Main release, OBDX installer), OBDX_Patch_Installer/installables/ui/deploy (Patchsets)

- There will be production enabled dist generated in the built UI.
- Bank can either directly deploy this dist to their server and point the application to that server as mentioned in point a) above OR
- Bank can copy the dist folder in their workspace and follow steps from point 3 in section 2.5.
- If bank wants to do any changes in the code, point a) steps needs to be followed.

NOTE: If banks want to debug UI in production builds, then dist should be created with below configuration enabled in webpack.prod.js

devtool: 'eval',

- This will however increase the files deployed on server and reduce the proformance on production. Refer Webpack documentation https://webpack.js.org/configuration/devtool/ for more details.

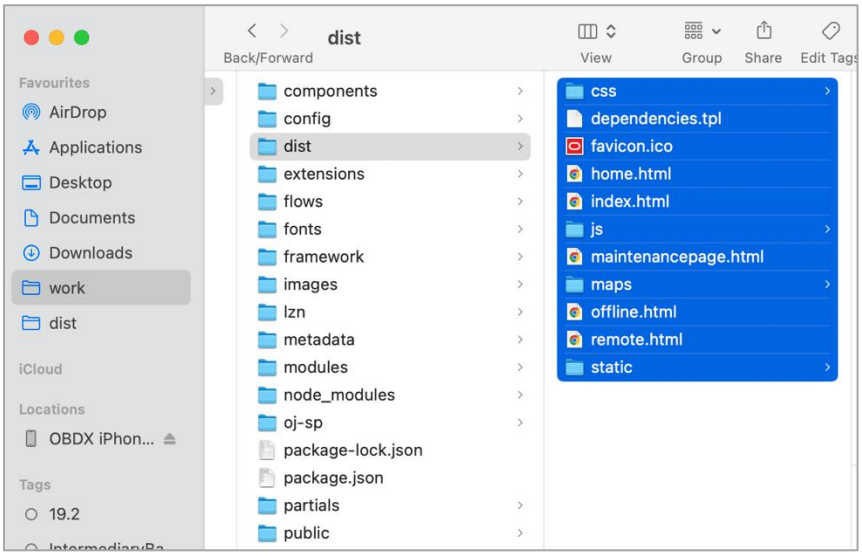
2.5 Create Project Using local UI within the workspace.

1. Extract the unbuilt UI and follow steps up to 5 in the above section 2.4.
2. After step 4, run below command to generate dist folder.

npm run webpack-dev – this will generate development enabled dist

npm run webpack-build- this will generate production enabled dist

3. Once the dist folder is created, copy all files inside dist folder and save it in the Zigbank project->Staging ->www folder.



4. Open Index.html and home.html and add below line inside head section below meta tag.

```
<script src="cordova.js" type="text/javascript"></script>
```

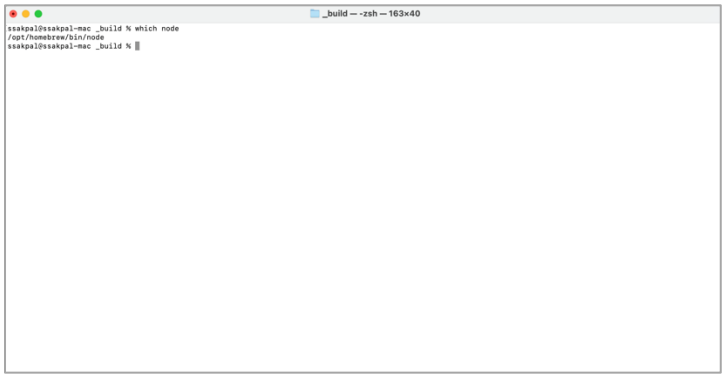
5. Set the server URL in app.plist against key_server_url. This is the URL where backend services are hosted.
6. With this setup, since the files generated in dist folder are minified format we cannot change the code. If any change needs to be done in any UI file, then the changes must be done in the UI folder, built it again to generate dist and copy the files to workspace again. Since this is tedious process, we recommend to setup local server and host UI there for development.

2.6 Configurations for the IOS application:

1. For Running the application on latest M1 Chip Macs with XCode 15.0, please update below path.
   - Open terminal application on Mac and type below command

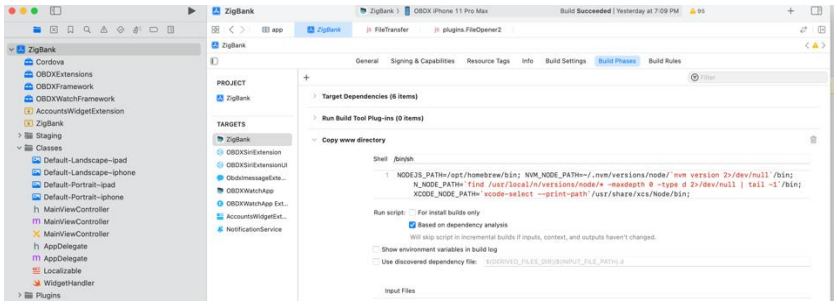
     which node

   - It should print the path where node is installed as shown in the SS below.



- Go to Zigbank Target->Build Phases->Copy www directory and update the NODEJS_PATH=/usr/local/bin; to

  NODEJS_PATH=/opt/homebrew/bin;



NOTE: This is only for M1 Chip macs. For Intel macs, no change is required.

For other configurations refer ‘app.plist’ (ZigBank/Resources) of the workspace.

Note: These are configurations for different features. The description of each in given below format

Type - Data Type of value

Purpose - Its usage

Value – The possible values

Configurable – Yes if bank can be allowed to change. No if the value is not allowed to be changed.

For SERVER_URLs:

Open XCode by clicking ZigBank.xcodeproj at zigbank/platforms/ios/

1. Adding URLs to app.plist (ZigBank/Resources)
   a. NONOAM (DB Authenticator setup)

SERVER_TYPE	NONOAM
KEY_SERVER_URL	https://mumaa012.in.oracle.com:18443/
WEB_URL	https://mumaa012.in.oracle.com:18443/

   b. OBDXTOKEN (Token based mechanism)

SERVER_TYPE	OBDXTOKEN
KEY_SERVER_URL	https://mumaa012.in.oracle.com:18443
WEB_URL	https://mumaa012.in.oracle.com:18443

   c. OAUTH Setup (Refer to installer pre-requisite documents for OAuth configurations)

SERVER_TYPE	OAUTH
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://mum00aon.in.oracle.com:14100/oauth2/rest/token
APP_CLIENT_ID	<Base64 of clientid:secret> of Mobile App client
APP_DOMAIN	OBDXMobileAppDomain
WATCH_CLIENT_ID	<Base64 of clientid:secret> of wearables
WATCH_DOMAIN	OBDXWearDomain
SNAPSHOT_CLIENT_ID	<Base64 of clientid:secret> of snapshot
SNAPSHOT_DOMAIN	OBDXSnapshotDomain
LOGIN_SCOPE	OBDXMobileAppResServer.OBDXLoginScope

d. 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

2. For SIRI:

1. This configuration is Optional.
2. By default, SIRI capability is set to YES. Bank can disable it.

CurrencyCode	Type: String Purpose: Currency code for Siri Payments Value: Currency Value. Ex: INR Configurable: Yes
SiriRequiredFlag	Type: Boolean Purpose: To enable/disable Siri capability Value: YES/NO Configurable: Yes
SIRIDebugEnabled	Type: Boolean Optional. Can be set if debugging is required in development mode. Default the value is NO Purpose: If we need to debug SIRI flow, this can be set to true. Refer section on how to configure device to debug SIRI. Value: YES/NO

	Configurable: Yes

3. Siri-Payload.plist
    a. This is present in ZigBank/Resources folder inside IOS workspace.
    b. It is provided to specify entries in the Siri payload based on transaction types (internal, domestic or international). Entries common to all the transaction types can also be entered.
    c. This is required if bank's SIRI payment payload differs from what is currently present in workspace and if bank needs to add certain mandatory fields for the payload.
    d. By default, SIRI will work with the given payload so no need to change anything in it.

3. To Enable SSL:

1. Refer section 2.7 on how to configure the workspace to enable SSL pinning in the application.
2. By default, SSL pinning is NO in the workspace for UAT and development. Recommended to set to YES for production URLs with a valid authorized certificate on server.

SSLPinningEnabled	Type: Boolean Purpose: To enable SSL Pinning. Value: YES for enabling. NO for disabling. Configurable: Yes
CertificateType	Type: String Purpose: File extension of SSL Pinned certificates Value: This is to set as .cer and the certificate file added in the workspace should also have .cer extension Configurable: Yes
PinnedUrl	Type: Array Purpose: Pinning URL to be entered here. This is the https URL of the server against which the certificate will be verified. Value: https server URL Configurable: Yes
PinnedCertificateName	Type: Array Purpose: For verification of SSL, this certificate will be pinned in the application and verified against the server URL.

	Value: Houses the certificate name (without extension) of the pinning certificate. Old certificate (about to expire) and new one can co-exist. Configurable: Yes
SSLPinningEnabledNoNetworkCall	Provides the option of whether to load the login page if SSL Pinning fails. SSLPinningEnabled also must be set to YES for it to work. If set to YES and SSLPinningEnabled is set to YES then if SSL Pinning fails, then login page does not load. If set to NO and SSLPinningEnabled is set to YES then if SSL Pinning fails, then login page loads. Configurable: Yes

4. To Enable Force Update:

This is optional configuration. Refer section 2.8 on more details on how to configure the workspace for this.

ForceUpdate	Type: Boolean Purpose: To enable updating of app forcibly Value: If set to YES, then the application will check for updates from the Appstore and display a non-dismissing popup. User needs to forcefully update the application. Default value: NO Configurable: Yes
AppStoreID	Type: String Purpose: The force update will be checked against this application ID Value: Enter the ID of the application from AppStore. Configurable: Yes
AppStoreURL	Type: String Purpose: URL to identify app in AppStore for force update Value: It is set to https://itunes.apple.com/in/app/id@@AppStoreID?mt=8 Just replace @@AppStoreID to what is set above for 'AppStoreID' Configurable: Just update as mentioned above. There is no need to change this URL.

5. **WATCH Application parameters**:

   Applicable only if Watch target is added in the workspace.

WatchOATCorp	Type: Boolean Purpose:To enable/disable Own Account Transfer through Apple Watch OBDX application for corporate users only. If set to YES, then OWN Account Transfer option will be available in the watch application. Value: YES to display the option. NO to hide that option Configurable: Yes
WatchSnapshot	Type: Boolean Purpose: To enable/disable snapshot capability in Apple Watch OBDX application. If set to YES, then Snapshot option will be available in the watch application. Value: YES to display the option. NO to hide that option. Configurable: Yes
WatchLocateUs	Type: Boolean Purpose: To enable/disable ATM Location option in Apple Watch OBDX application. If set to YES, then ATM Location option will be available in the watch application. Value: YES to display the option. NO to hide that option Configurable: Yes
WATCHMAXATTEMPTS	Type: Number Purpose: The number of time PIN login is allowed in the watch application Value: Set the value to which Bank wants to restrict the PIN invalid attempts. After these attempts, user will need to register the application PIN again. Configurable: Yes

6. **For DISPLAYING MAINTENANCE PAGE (Optional):**

   1. This is optional configuration. Bank needs to do this if they want to display maintenance page in case of any server error.
   2. By default, a maintenance page html is provided in workspace inside Zigbank->Staging->www. If bank needs to reconfigure the content, they can edit this page html.

3. Also, by default, SHOW_MAINTENANCE_PAGE flag is set to YES and error code is set to 503.

SHOW_MAINTENANCE_PAGE	Type: Boolean Purpose: To display a maintenance page if server is down. Value: true to display. If bank doesn't need any page to be displayed, it can be set to false. Default is true Configurable: Yes
MAINTENANCE_PAGE_STATUS_CODE	Type: Array Purpose: To set the status code for which maintenance page is to be displayed. This will be used only if "SHOW_MAINTENANCE_PAGE" value is true. Value: status code to be checked (E.g. 503, 504 etc) Default value is 503. Bank can set only one value or multiple status codes if required. Configurable: Yes

4. NOTE: If UI is built and copied into workspace as local UI (by using section 2.5), and bank wants to use maintenance page, then additional changes are required as below:
5. No need to do these changes if the UI remotely hosted.
    1. In app.plist, add and additional property "MAINTENANCE_PAGE_URL" as String and set the maintenance page URL which needs to be displayed when server is down.
    2. Open index.html and add below code in script tag as below:

```
<script type="text/javascript" charset="utf-8">

        function init() {
            var
maintenancePageUrl,maintenancePageStatusCode,showMaintenancePage;
            var server_url = "http://ofss-mum-
715.snbomprshared1.gbucdsint02bom.oraclevcn.com:17777"
            var home_html = "?ojr=home";
            var url = server_url + "/" + home_html;

plugins.appPreferences.fetch(MAINTENANCE_PAGE_STATUS_CODE_SUCCE
SS, error, 'MAINTENANCE_PAGE_STATUS_CODE');

plugins.appPreferences.fetch(SHOW_MAINTENANCE_PAGE_SUCCESS, error,
'SHOW_MAINTENANCE_PAGE');

                function MAINTENANCE_PAGE_URL_SUCCESS(value) {
                    maintenancePageUrl = value;
```

```
            showMaintenance();
        }

        function MAINTENANCE_PAGE_STATUS_CODE_SUCCESS(value) {
            maintenancePageStatusCode = value;
        }

        function SHOW_MAINTENANCE_PAGE_SUCCESS(value) {
            var xmlhttp = new XMLHttpRequest();
            xmlhttp.onreadystatechange = function() {
                if (xmlhttp.readyState === 4) {
                    if (maintenancePageStatusCode.includes(xmlhttp.status) &&
value) {

plugins.appPreferences.fetch(MAINTENANCE_PAGE_URL_SUCCESS, error,
'MAINTENANCE_PAGE_URL');

                    }
                }
            }
            xmlhttp.open("GET", url, true);
            xmlhttp.send();
        }

        function error(err) {
            console.log(err);
        }

        showMaintenance = function () {
            var xmlhttpMaintenance = new XMLHttpRequest();

            xmlhttpMaintenance.onreadystatechange = function () {

                if (xmlhttpMaintenance.readyState === 4) {
                    document.getElementById("obdx-dashboard").style.display
="none";

document.getElementById("showMaintenancePageid").innerHTML =
this.responseText;
                }
            };
            xmlhttpMaintenance.open("GET", maintenancePageUrl, true);
            xmlhttpMaintenance.setRequestHeader("Cache-Control", "no-cache,
no-store, max-age=0");
            xmlhttpMaintenance.send();
        }

    }
```

6. `</script>`
7.

3. In the body tag add onload="init();" attribute.
4. Update the server_url in the above script to bank's server url

7. COMMON CONFIGURATIONS:

XcodeBuildVersion	Build version with which the workspace is built with. Configurable: No
SUITENAME	Group identifier for sharing keystore information. Same as given in app groups (mandatory to be given same as App Group name). App Groups are linked with the provisioning profile and its value can be verified from the Zigbank target->Signing Capabilties. This value is important for the secured storage of the information. Configurable: Yes
BankName	Name of bank to be shown on touch id / face id popup Configurable: Yes
DomainDeployment	To be set YES for token-based development. Configurable: NO

8. FOR CHATBOT:

1. This is optional configuration. Bank needs to do this if chat bot is added in their workspace.
2. Adding chatbot support to mobile application (Optional) (refer section **ODA Chatbot Inclusion** for more details)

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

9. For eKYC

1. This is optional. Bank needs to do this if ekyc is enabled.
2. Adding eKYC verification support to mobile application (Optional) (see section eKYC Implementation more details)

LX_CLIENT_ID	The client ID
LX_ADDRESS	This is the live exp server address. By default it is

Adding Bundle Identifiers. (Refer Section 2.10 for generating the appIDs and certificates)

We need to set bundle identifier for each framework within the workspace.

- Bundle identifiers needs to be added in the Info.plist of each the frameworks
    1. For example, the bundle identifier used is abc.def.ghi.jkl. The steps to be followed are,

2. Right click on OBDXFramework.framework(in Xcode's Project Navigator) -> Show in Finder
3. When the finder directory opens the right click OBDXFramework.xcframework -> select ios-arm64 -> Select OBDXFramework.framework
4. Open Info.plist and set Bundle identifier as abc.def.ghi.jkl.OBDXFramework
5. Repeat the steps for the other three frameworks as well, with the following values:
6. Bundle identifier for Cordova.framework : abc.def.ghi.jkl.Cordova
7. Bundle identifier for OBDXExtensions.framework : abc.def.ghi.jkl.OBDXExtensions
8. Bundle identifier for OBDXWatchFramework.framework : abc.def.ghi.jkl. OBDXWatchFramework

- Set the identifier in the Signing Capabilities tab in Xcode for each target.
  1. Open Xcode project in Xcode and select each target and go to Signing and Capabilities and update correct bundle identifier for each target.
  2. Example if main target bundle identifier is "com.ofss.digx.obdx.zigbank" then each target should have below format for bundler indentifiers

     OBDXSiriExtensionSiri – com.ofss.digx.obdx.zigbank.OBDXSiriExtension

     OBDXSiriExtensionUI – com.ofss.digx.obdx.zigbank.OBDXSiriExtensionUI

     ObdxImessageExtension– com.ofss.digx.obdx.zigbank.ObdxImessageExtension

     OBDXWatchApp – com.ofss.digx.obdx.zigbank.watchkitapp

     OBDXWatchAppExtension- com.ofss.digx.obdx.zigbank.watchkitapp.watchkitextension

     AccountsWidgetExtension – com.ofss.digx.obdx.zigbank.AccountsWidget

     NotificationExtension- om.ofss.digx.obdx.zigbank.OBDXNotificationExtension

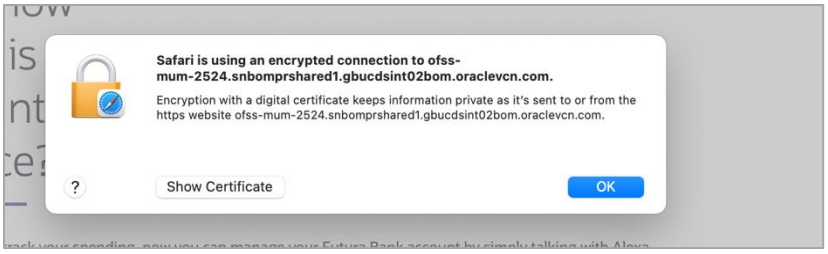
2.7 Enabling SSL pinning in the application

SSL pinning in required to securely connect with a valid bank server. It is recommended to enable this in production. By default, SSL pinning is set to NO in the application for development purpose so that the application can connect to http URLs and no SSL certificates are required to be configured on the server.

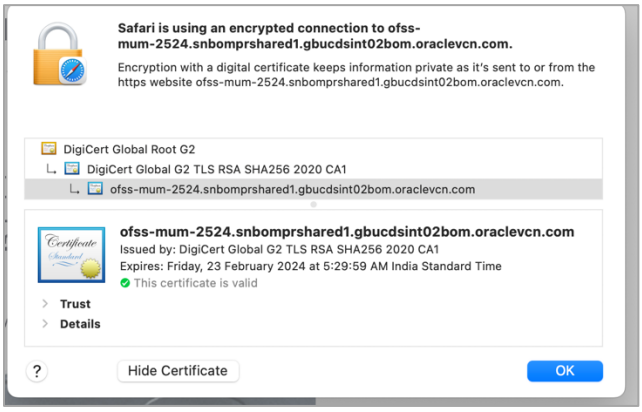
To enable SSL pinning, bank needs to follow the configurations mentioned in the section "Configurations".

Along with those, there are below steps to be followed to include the certificate into workspace:

- Open bank's https website in Safari on Mac machine
- The website will display a lock icon in the address bar next to the URL.
- Click on that lock icon. It will display a window as below:



- Click on Show certificate and below window will be displayed

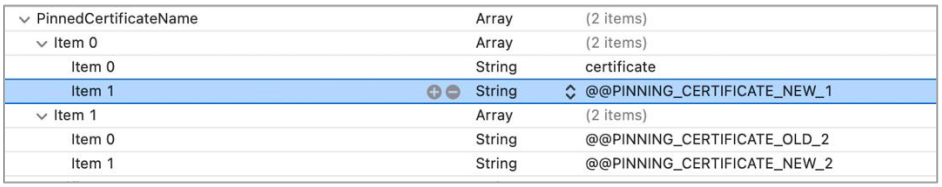


- Press and drag the certificate icon from Safari to any location on your machine.
- Rename it to any name.cer
- Open Zigbank project in Xcode
- Right Click on Resources folder and select “Add Files to Zigbank”
- Select the certificate file which is saved in above step.
- Make sure “Copy items if needed” and target is set to Zigbank as below:

- The certificate will be added in the Resources folder.
- Copy the name and add it in the app.plist against @@PINNING_CERTIFICATE_OLD_1 for PinnedCertificateName as shown below. Refer configuration section for this key information.

Note: Since this is an array, bank can add multiple certificates for @@PINNING_CERTIFICATE_OLD_1, @@PINNING_CERTIFICATE_OLD_2.

Also, since SSL certificate are renewed after the expiry @@PINNING_CERTIFICATE_NEW_1 and @@PINNING_CERTIFICATE_NEW_2 options are provided.

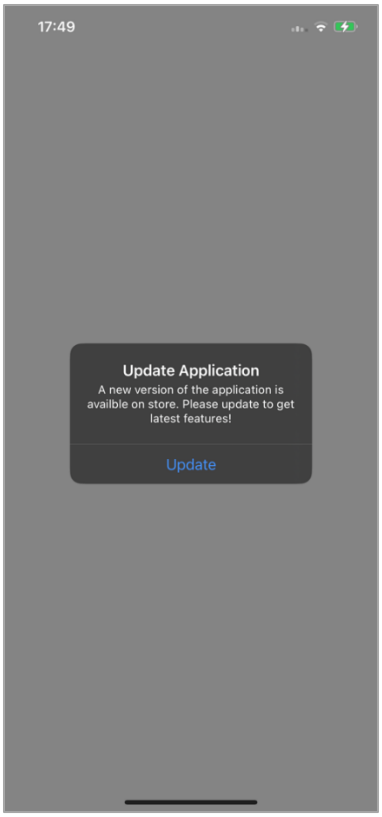


These are the corresponding new certificate names which can be added by the bank when the old certificates are about to expire and release this version of application to Appstore so that the application continues to work with SSL pinning even after old certificate has expired. Same activity bank can continue to do for every year before old certificate expires.

- To add the new certificates in workspace, bank has to follow same steps as mentioned above
- The old unused certificates can be deleted after they are expired.
- After the certificates are configured, next step is to set 'PinnedUrl' key in the app.plist. Refer configuration section for this key information. Add the https URL against which the certificates are to be verified. If there are multiple site certificates added, then bank has to set all those urls in each item as below:

2.8 Enabling Force update

- This is optional configuration.
- When a new version of the application is available on AppStore, user should be notified to upgrade their application. This flag will check the update and display a non-cancellable popup message to the user to update their application.
- For this to happen, enable this flag and other values as mentioned in the configuration section.
- Bank needs to set the version of the application which will be newer to what is present on Appstore.
- Set it in "Bundle version string (short)" in ZigbankInfo.plist and in marketing version in Watch target->Build Settings. (Watch target setting is only required if Watch target is present for bank in its workspace.
- So if Appstore has version 1.0.1, the new application code can have version greater that 1.0.1 which can be either 2.0 or 1.0.2 or 1.1.0 or 1.1.1 etc. Once this new version is released to Appstore, the application already installed on user's device will compare the installed version with what is available on Appstore.
- If the version available on AppStore is greater, then below popup will be displayed.



On clicking the button, user will be redirected to the Appstore page of that application. (Ensure to set correct AppStoreID in the configuration for this redirection)

- The popup header text and message can be configured in “Localizable.strings” file inside Classes folder in the workspace for below keys:

  Header – APP_UPDATE_HEADER
  Message - APP_UPDATE_TEXT
  Button text- APP_UPDATE_BTN_TEXT

- If bank enables this feature in any versions after the first application is launched, then this logic will work from the time the users install the version which has this logic enabled. Ex: If bank has 1.0 in Appstore for which this flag as false and bank releases 1.1 in Appstore with this flag enabled, then user needs to install the 1.1. application manually. Since 1.0 didn’t have that flag enabled, it will not check for any updates.

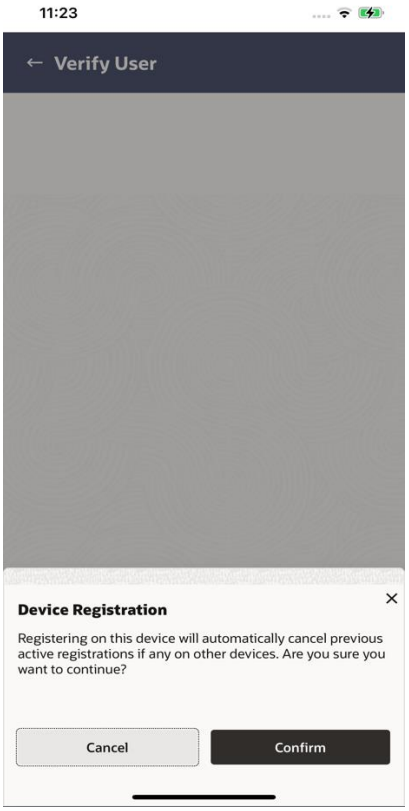
  However, every future release made to Appstore will check for any updates and display the force update popup.

2.9 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/FACE on the de-registered devices.

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

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



If user cancel, the process is exited.

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

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

ALLOWED_DEVICE_COUNT to any value between than 1 and 100.

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

ALLOWED_PUSH_DEVICE_COUNT any value between 1 and -1

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

2.10 Generating Certificates for Development, Production and Push Notifications

This is required for running the application on device for debugging, testing as well for releasing the application to Appstore.

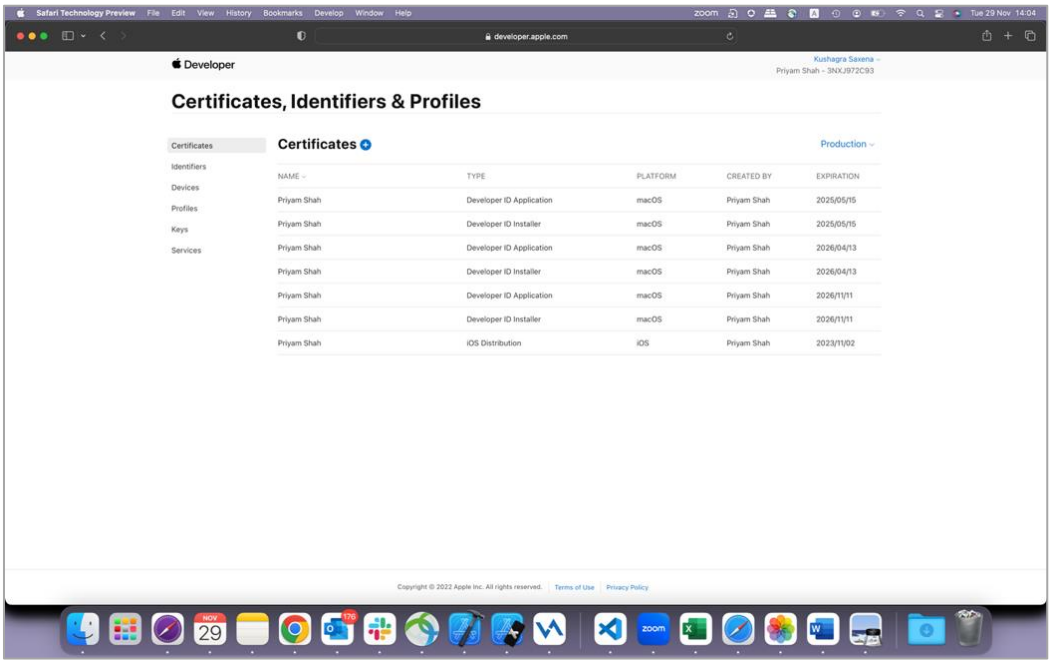
Bank can refer to Apple's documentation on how to create certificates and provisioning profiles.

Create all certificates (by uploading CSR from keychain utility), provisioning profiles and push certificates as shown below by login in developer console. For development add device UUIDs and add same to provisioning profiles. Add capabilities as shown below and ensure the bundle identifier matches the one of the application in XCode.

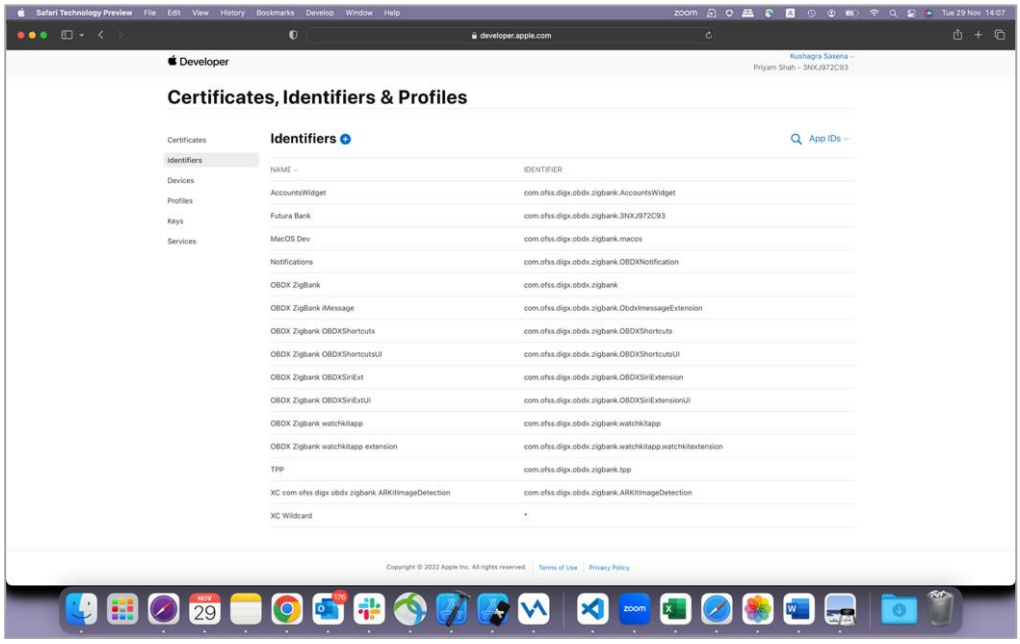
There are parts to it:

1. Certificate Creation
2. AppID creation
3. Profile creation
4. Enabling Push services and push certificate creation
5. Adding devices for testing on actual devices

Certificate Creation: Below is the screen on apple developer portal where bank needs to create distribution and Development certificates.



AppID creation: Below is the screen where bank needs to create appIDs for each target



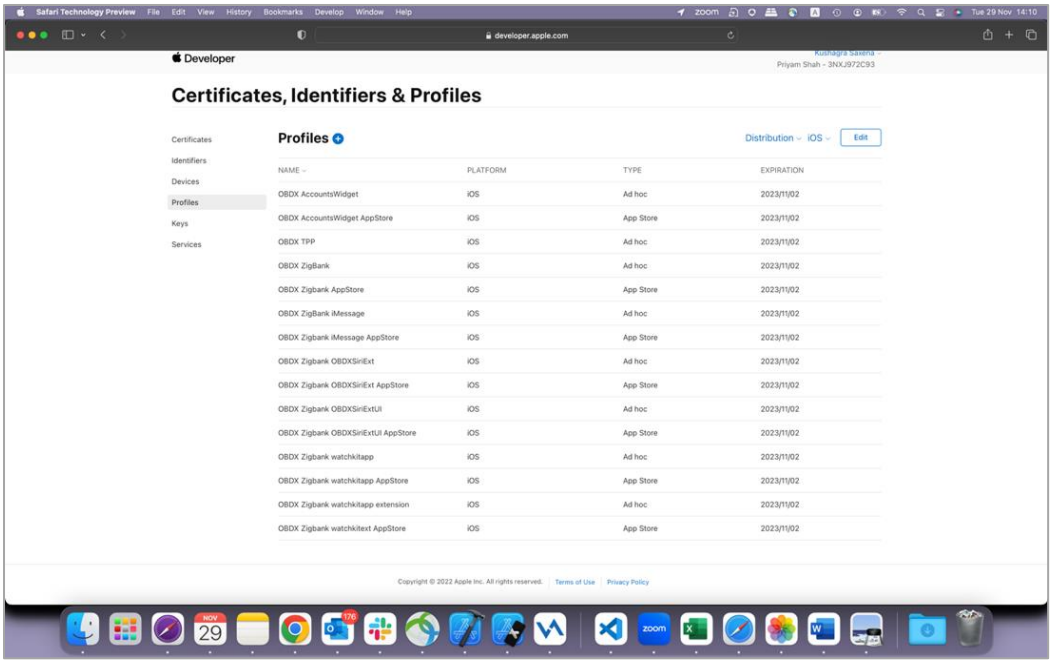
Ensure AppGroups capability is added to all profiles and for appIDs.

Ensure SiriKit, App Groups, Push Notifications, Associated domain capabilities are added in Zigbank appIDs.

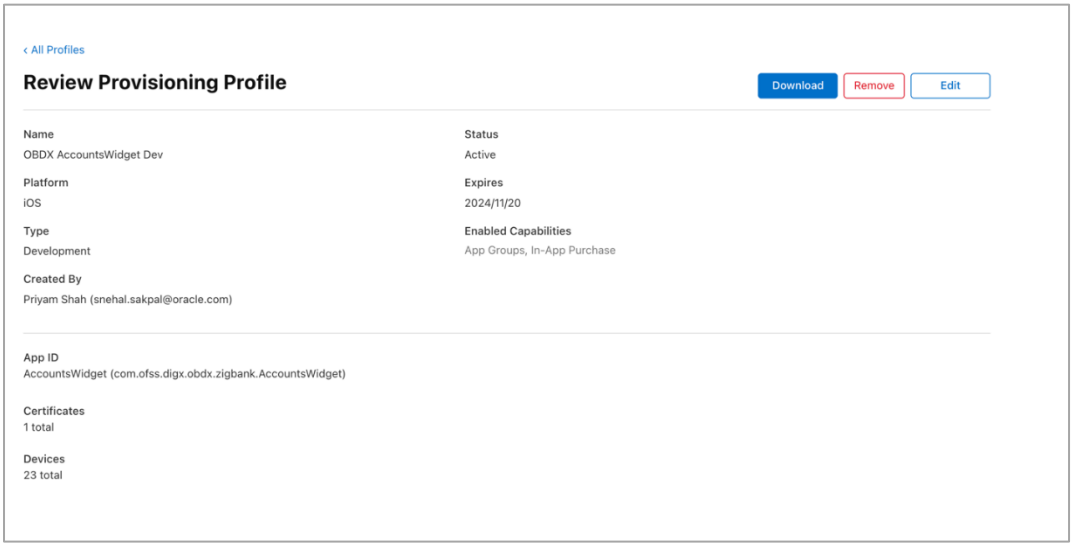
Below are the appIDs which we need for OBDX application in similar format as below:

OBDX ZigBank iMessage	com.ofss.digx.obdx.zigbank.ObdxImessageExtension
OBDX Zigbank OBDXSiriExt	com.ofss.digx.obdx.zigbank.OBDXSiriExtension
OBDX Zigbank OBDXSiriExtUI	com.ofss.digx.obdx.zigbank.OBDXSiriExtensionUI
OBDX Zigbank watchkitapp	com.ofss.digx.obdx.zigbank.watchkitapp
OBDX Zigbank watchkitapp extension	com.ofss.digx.obdx.zigbank.watchkitapp.watchkitextension
OBDX Notification Extension	com.ofss.digx.obdx.zigbank.OBDXNotificationExtension
OBDX ZigBank	com.ofss.digx.obdx.zigbank
AccountsWidget	com.ofss.digx.obdx.zigbank.AccountsWidget

Profile Creation: Bank needs to create development, distribution and Appstore profiles for each target.



Select appropriate AppIDs to relevant profile and appropriate certificates. E.g. AccountWidget target will have development certificate with appId created for AccountWidget if development type profile is created. Likewise for other targets.

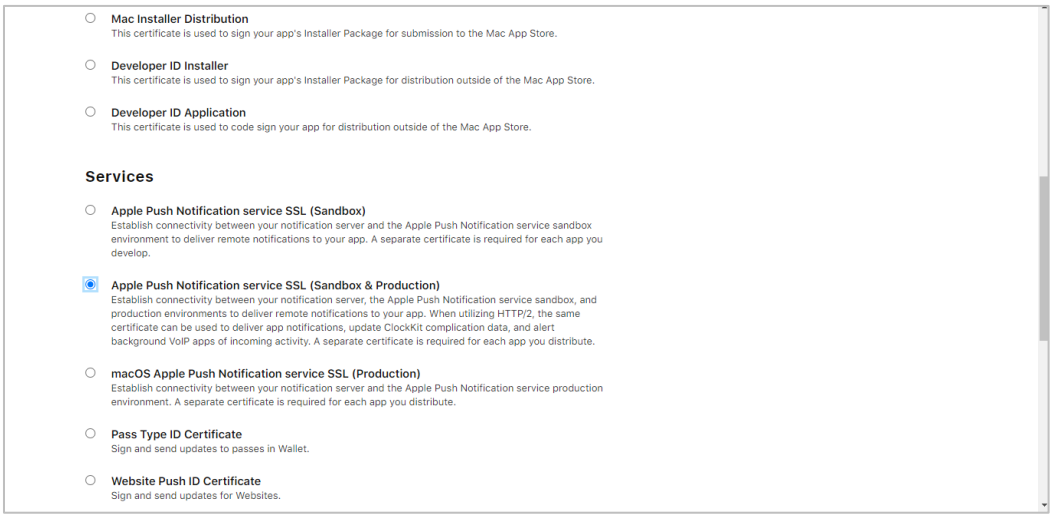


2.11 Setup for Push Notification in the application

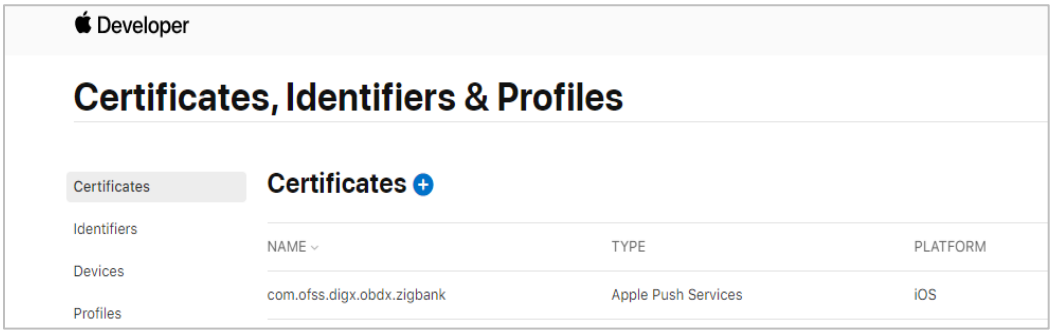
There are two ways: p12 and .p8

For p12 bank needs to create APNS certificate, it's password and store that certificate on server and update database configurations. For p8, bank needs to setup Key and update database with the details. All details are mentioned below:

1. Certificate needs to be created for push notification as below:

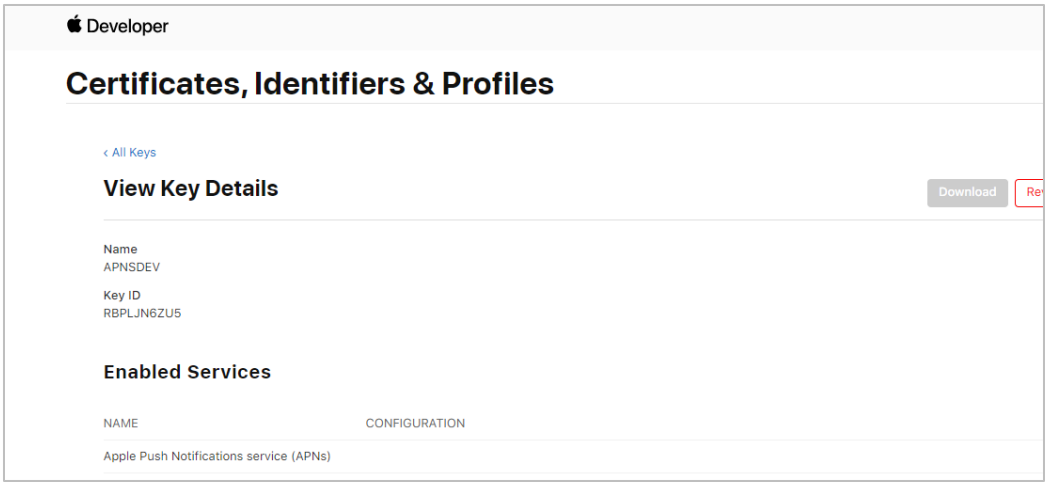


Note the certificate/bundle name



Download this certificate and store on server E.g. resources/mobile/ios-cert.p12. We need this path to be set in database.

1. Note the Team ID from top right corner below the account name.
2. Note the bundle identifier for Zigbank (Refer section 2.10 AppIDs)
3. Navigate to the "Keys" section and create APNS key.
4. Note APNS key and download the .p8 file. We need the contents of this file to update in database



5. Below are the configurations to be done at database level:

Common configuration

Sr. No.	Table	PROP_ID	CATEGORY_ID	PROP_VALUE	Purpose
1	DIGX_FW_CONFIG_ALL_B	APNSKeyStore	DispatchDetails	DATABASE or CONNECTOR	Specifies whether to pick certificate password(value of APNS prop) from database or from connector. Default DB (No change)
2	DIGX_FW_CONFIG_ALL_B	APNSCertKeyStore	DispatchDetails	DATABASE	

3	DIGX_FW_CONFIG_ALL_B	proxy	DispatchDetails	<protocol,proxy_address>	Provides proxy address, if any, to be provided while connecting to APNS server. Delete row if proxy not required. Example: HTTP,148.50.60.8,80
4	DIGX_FW_CONFIG_ALL_B	CERT_TYPE	DispatchDetails	For dev push certs add row with value 'dev' Possible values for Dev: 1. dev 2. dev-cert For Production: 3. prod 4. prod-cert	For prod push certificates this row is not required as it takes prod by default
5	DIGX_FW_CONFIG_VAR_B	APNS_BUNDLE		Eg. com.ofss.digx.obdx.zigbank	Bundle Name (Update for all entities)
7	DIGX_FW_CONFIG_VAR_B	APNS_TEAM ID		Eg. 3NX1974C93	Team ID of Apple developer account (Update for all entities)

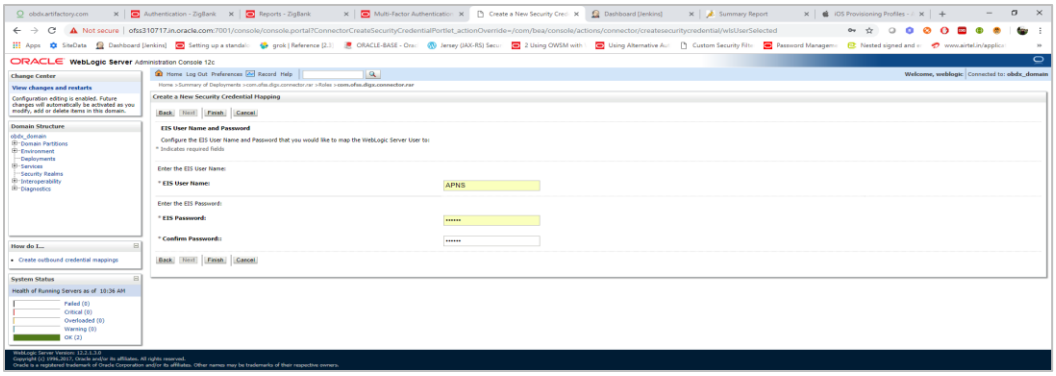
If bank uses, P12 below are the configurations to be set:

Sr. No.	Table	PROP_ID	CATEGORY_ID	PROP_VALUE	Purpose
1	DIGX_FW_CONFIG_ALL_B	CERT_TYPE	DispatchDetails	dev-cert prod-cert	To specify to take the certificate file from the path mentioned in "ios_cert_path" prop_id
2	DIGX_FW_CONFIG_ALL_B	ios_cert_path	DispatchDetails	resources/mobile/ios-cert.p12	This is the path of the p12 certificate stored on server
3	DIGX_FW_CONFIG_VAR_B	APNS	DispatchDetails	Certificate password	Provide APNS certificate password created in section 2.11

If bank has configured .p8 for push notification:

Sr. No.	Table	PROP_ID	CATEGORY_ID	PROP_VALUE	Purpose
1	DIGX_FW_CONFIG_VAR_B	APNS	DispatchDetails	<Key ID>	Provides key of .p8 certificate
2	DIGX_FW_CONFIG_ALL_B	CERT_TYPE	DispatchDetails	For dev push certs add row with value 'dev'	For prod push certificates this row is not required
3	DIGX_FW_CONFIG_VAR_B	APNSCert		Eg – -----BEGIN PRIVATE KEY----- abcd -----END PRIVATE KEY-----	Open the .p8 file and copy contents to column (Update for all entities)

6. If CONNECTOR is selected in " APNSKeyStore"" update key as below



2.12 Push Notification Actionable Alerts Configuration

To enable deep linking with actionable alerts make the following changes on the server end to the push notifications payload:

1. Send the “category" as “pac”.
2. Send the required deep-linking URL in “SUMMARY_TEXT”.

2.13 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.

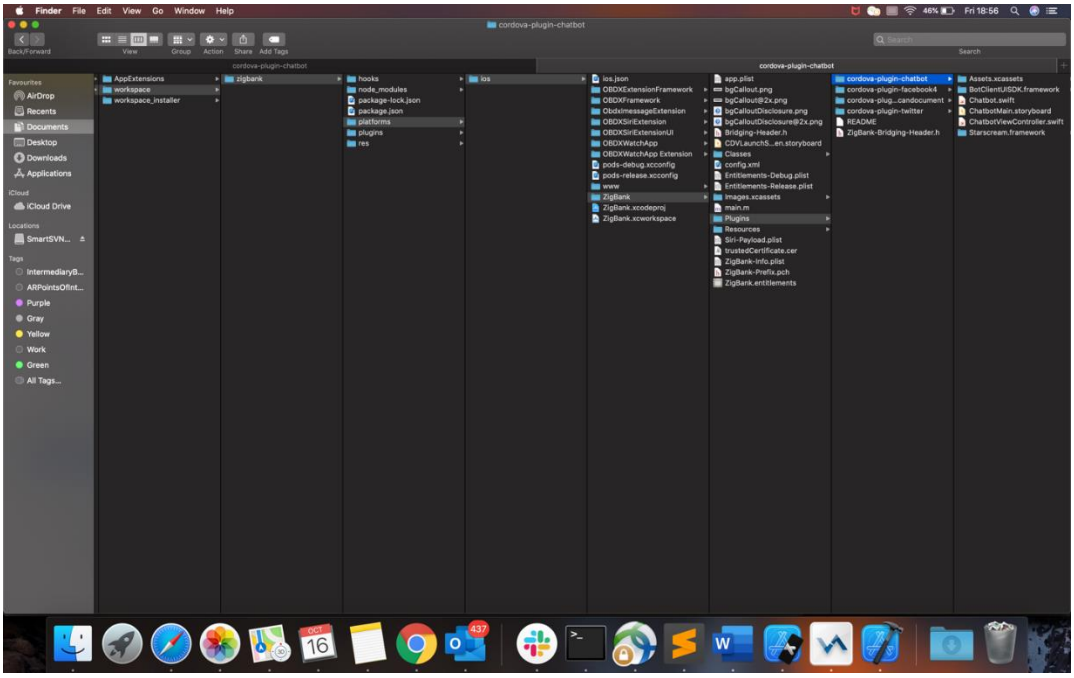
   So, ideally 5 minutes (existing value in base UI code) is an ideal time. Any changes made in this value should satisfy below pre-condition
5. There is an OTP expiration time set in “digx_fw_config_ALL_b” table.
6. Also, there is business policy check set to 10 minutes for validation of the generated 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.

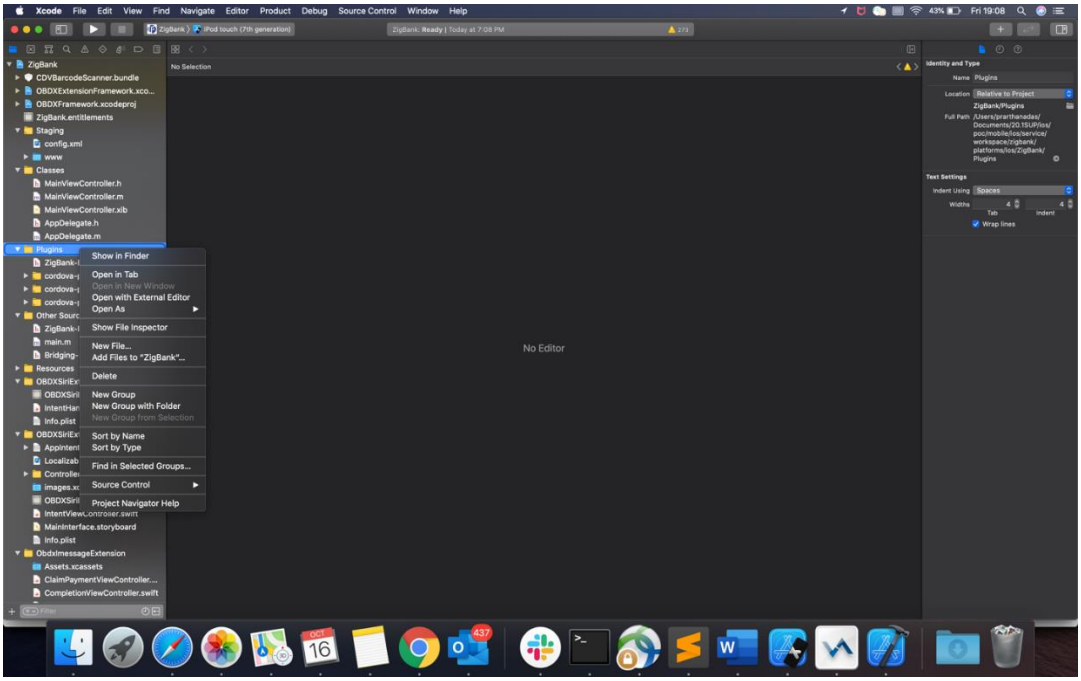
2.14 ODA Chatbot Inclusion

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

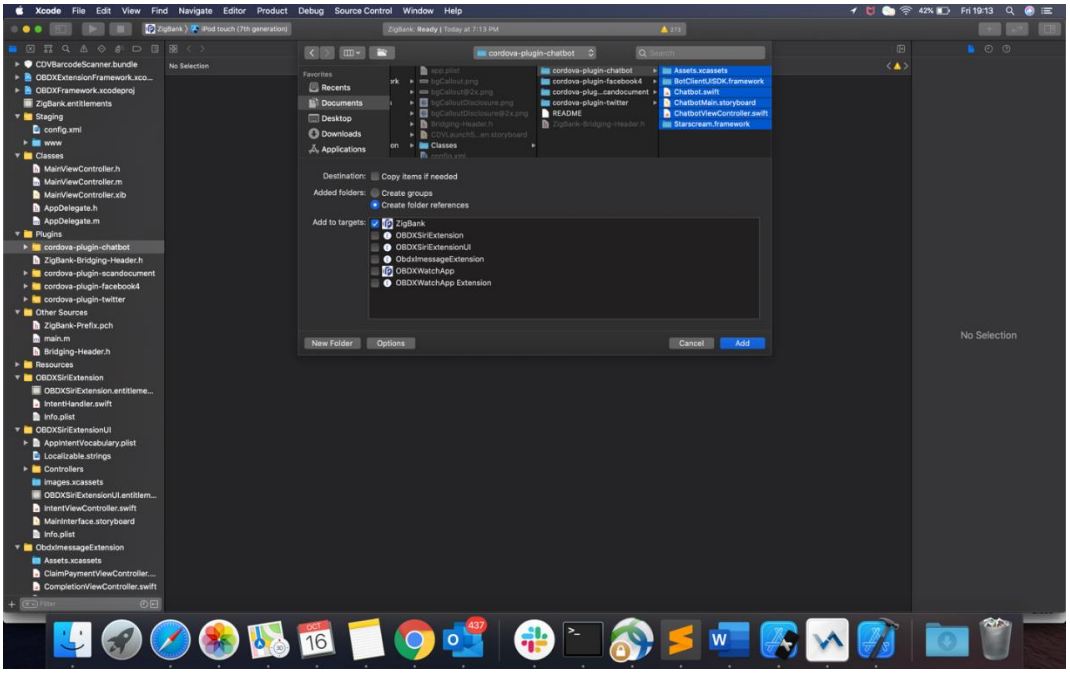
1. Copy the folder "cordova-plugin-chatbot" from the SVN path : workspace_installer/AppExtensions/ODAChatbot The frameworks can be found at ODA Client SDK for iOS x.y.z - Latest in https://www.oracle.com/downloads/cloud/amce-downloads.html#license-lightbox. After downloading and unzipping the latest version the frameworks for an actual device and simulator can be found inside the folders named "FrameworksActualDevice" and "FrameworksSimulator" respectively. Frameworks to be chosen as per the target and pasted inside "cordova-plugin-chatbot".
2. Paste the folder "cordova-plugin-chatbot", copied previously in the path : workspace_installer/Zigbank/plugins A screenshot of the destination in Finder is attached herewith.



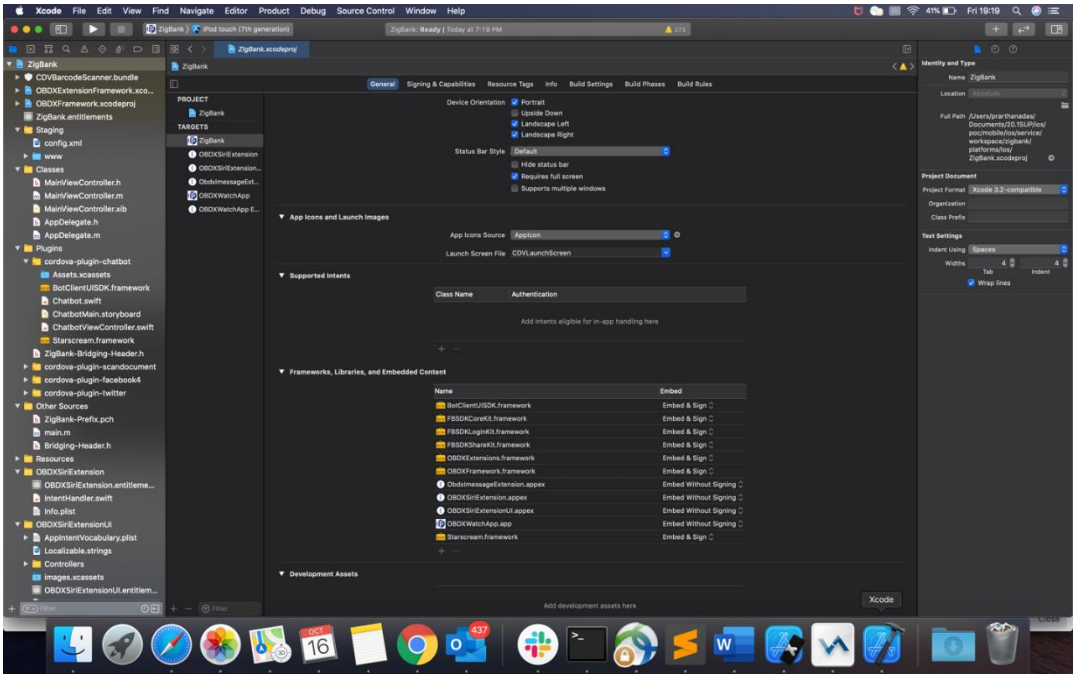
3. Open the Zigbank.xcodeproj file, right click on "Plugins" folder and select "New Group" option. Name the group as "cordova-plugin-chatbot".



4. Right click on the newly created group and select "Add files to "Zigbank"" option, and add all the contents of "cordova-plugin-chatbot" folder, pasted previously.



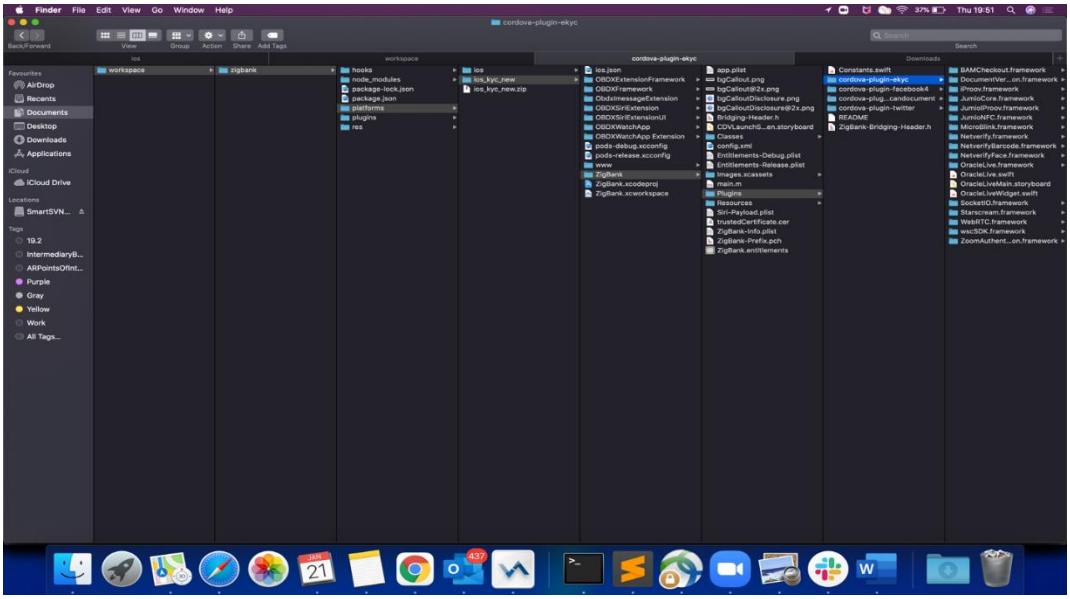
5. After addition of the files, go to "General" tab for "Zigbank" target and under the "Frameworks, Libraries and Embedded Content" section change the embed type of the frameworks "Starscream.framework" and "BotClientUISDK.framework" to "Embed and Sign". Failing to do so will make the app crash after installation.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
ZigBank
CDVBarcodeScanner.bundle
OBDXExtensionFramework.xco...
OBDXFramework.xcodeproj
ZigBank.entitlements
Staging
config.xml
www
Classes
MainViewController.h
MainViewController.m
MainViewController.xib
AppDelegate.h
AppDelegate.m
Plugins
cordova-plugin-chatbot
Assets.xcassets
BotClientUISDK.framework
Chatbot.swift
ChatbotMain.storyboard
ChatbotViewController.swift
Starscream.framework
ZigBank-Bridging-Header.h
cordova-plugin-scandocument
cordova-plugin-facebook4
cordova-plugin-twitter
Other Sources
ZigBank-Prefix.pch
main.m
Bridging-Header.h
Resources
OBDXSiriExtension
IntentHandler.swift
Info.plist
OBDXSiriExtensionUI
Localizable.strings
Controllers
images.xcassets
PROJECT
TARGETS
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
Device Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Default
Hide status bar
Requires full screen
Supports multiple windows
App Icons and Launch Images
App Icons Source
AppIcon
Launch Screen File
CDVLaunchScreen
Supported Intents
Class Name
Authentication
Add intents eligible for in-app handling here
Frameworks, Libraries, and Embedded Content
Name
Embed
BotClientUISDK.framework
FBSDKCoreKit.framework
FBSDKLoginKit.framework
FBSDKShareKit.framework
OBDXExtensions.framework
OBDXFramework.framework
ObdsImessageExtension.appex
OBDXSiriExtension.appex
OBDXSiriExtensionUI.appex
OBDXWatchApp.app
Starscream.framework
Embed & Sign
Embed Without Signing
Development Assets
Add development assets here
Xcode

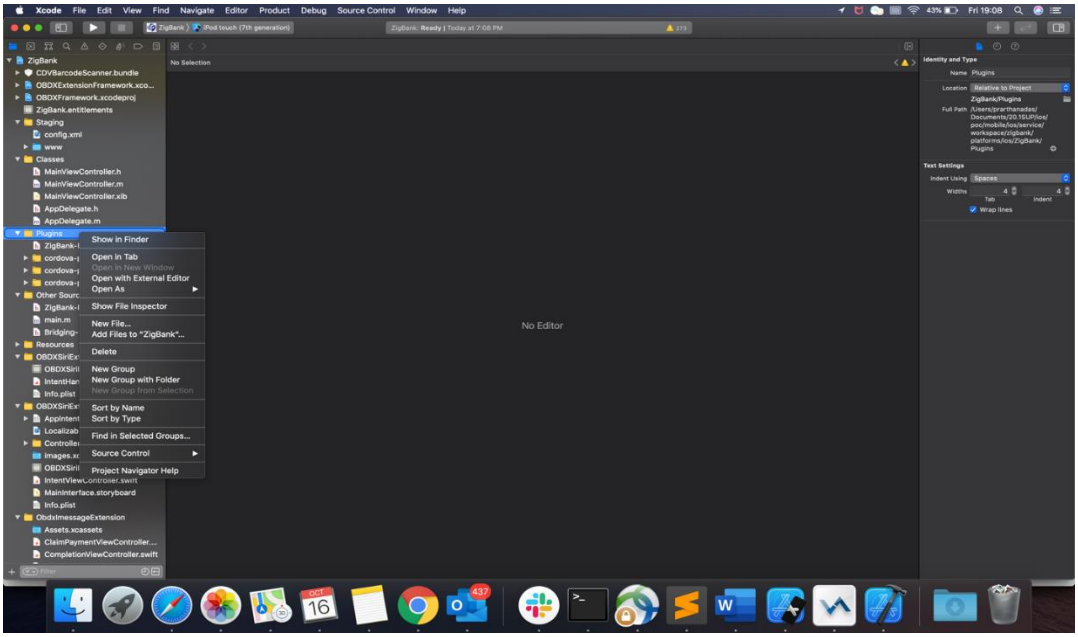
2.15 eKYC Implementation

To enable eKYC please follow the steps mentioned below:

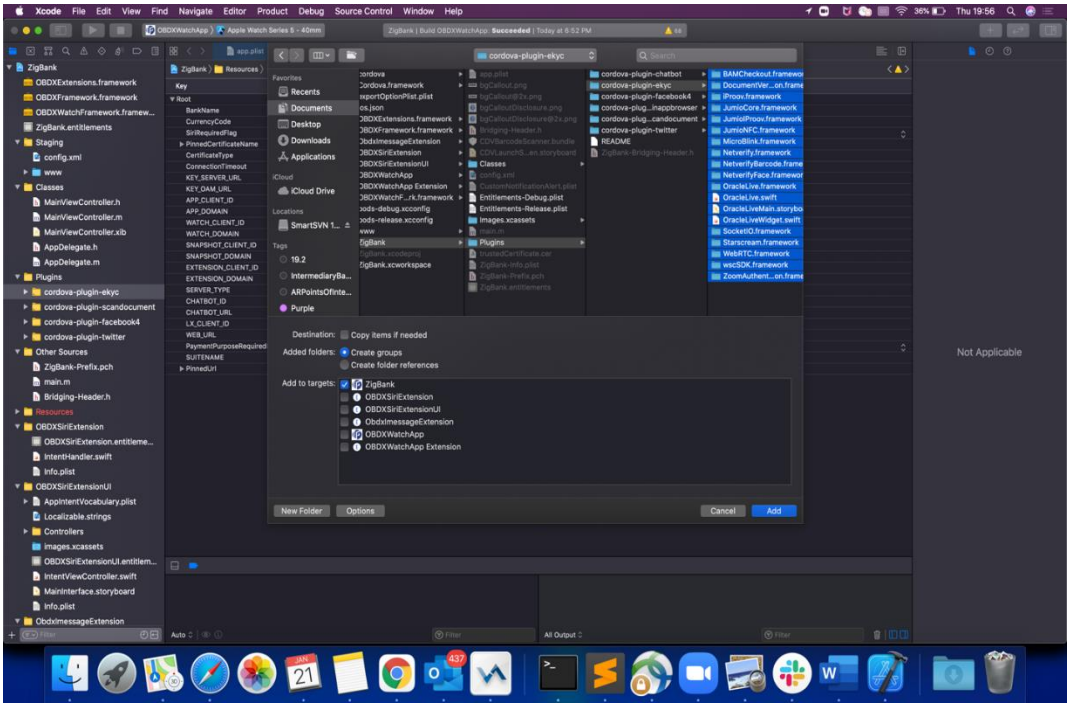
1. Download the iOS ID Verification SDK from oracle.live.api-ios-id-verification.zip from Oracle Live Experience. All the frameworks inside “release” folder of “oracle.live.api-ios-id-verification” are needed viz.
   - OracleLive.framework
   - WebRTC.framework
   - wscSDK.framework
2. Go to https://mobile-sdk.jumio.com/com/jumio/ios/jumio-mobile-sdk/ and navigate to the latest version to download the Jumio frameworks. Unzip the downloaded folder the following frameworks are of use to us:
   - BAMCheckout.framework
   - DocumentVerification.framework
   - iProov.framework
   - JumioCore.framework
   - JumioProov.framework
   - JumioNFC.framework
   - Microblink.framework
   - Netverify.framework
   - NetverifyBarcode.framework
   - NetverifyFace.framework
   - SocketIO.framework
   - Starscream.framework
   - ZoomAuthentication.framework
3. Paste the frameworks downloaded in the previous steps in the folder "cordova-plugin-ekyc" from the SVN path : workspace_installer/AppExtensions/eKYC
4. Paste the folder "cordova-plugin-ekyc", copied previously, in the path : workspace_installer/Zigbank/plugins A screenshot of the destination in Finder is attached herewith.



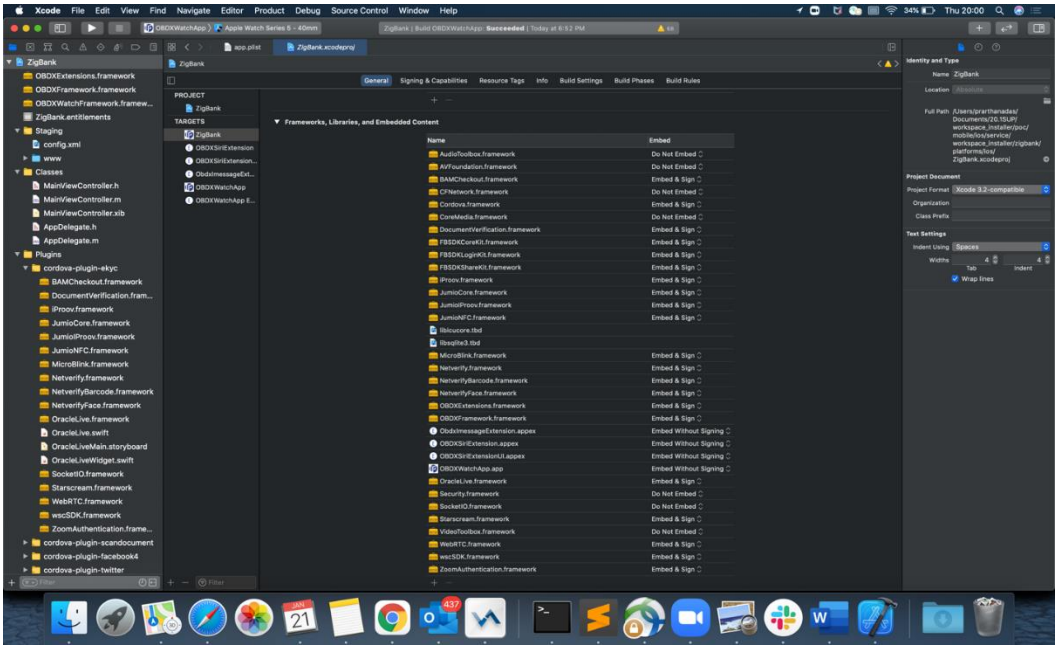
5. Open the Zigbank.xcodeproj file, right click on "Plugins" folder and select "New Group" option. Name the group as "cordova-plugin-ekyc".



6. Right click on the newly created group and select "Add files to "Zigbank"" option, and add all the contents of "cordova-plugin-ekyc" folder, pasted previously.



7. After addition of the files, go to "General" tab for "Zigbank" target and under the "Frameworks, Libraries and Embedded Content" section change the embed type of all the frameworks to "Embed and Sign". Failing to do so will make the app crash after installation.



Home

2.16 Widget Functionality

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

(Refer functional doc - User Manual Oracle Banking Digital Experience Quick Snapshot.docx)

1. All Accounts Widgets – Widget, showing top 3 account balance.
2. Account Details Widget - Widget, showing account balance of default account and last 3 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, transfer money, credit account overview and investment overview.
4. Scan to Pay Widget – Widget which allows to scan to pay.

Prerequisite:

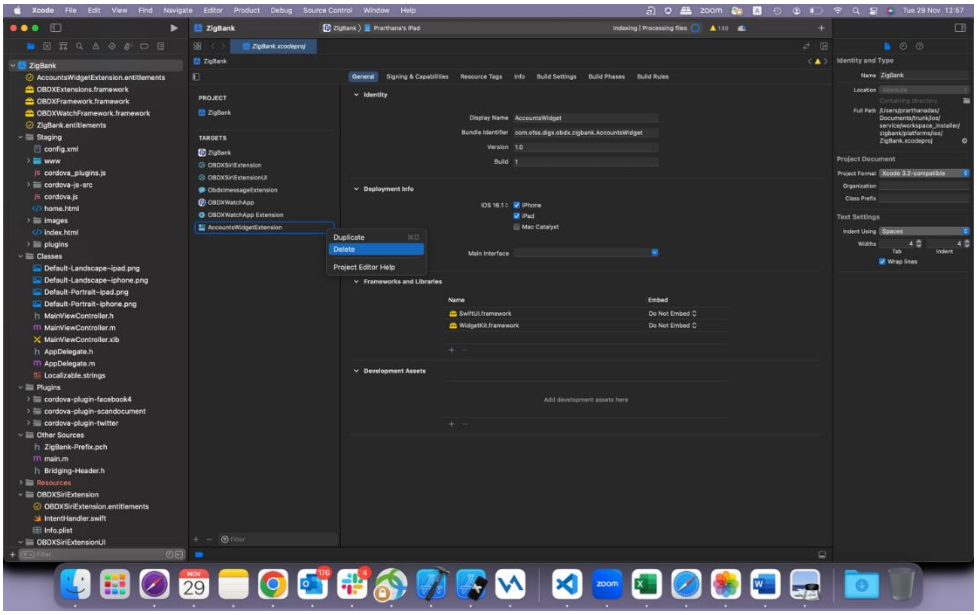
Quick Snapshot feature needs to be enabled in the application from the login screen. (Refer function doc - User Manual Oracle Banking Digital Experience Quick Snapshot.docx)

If bank doesn't need this feature, then can do below steps:

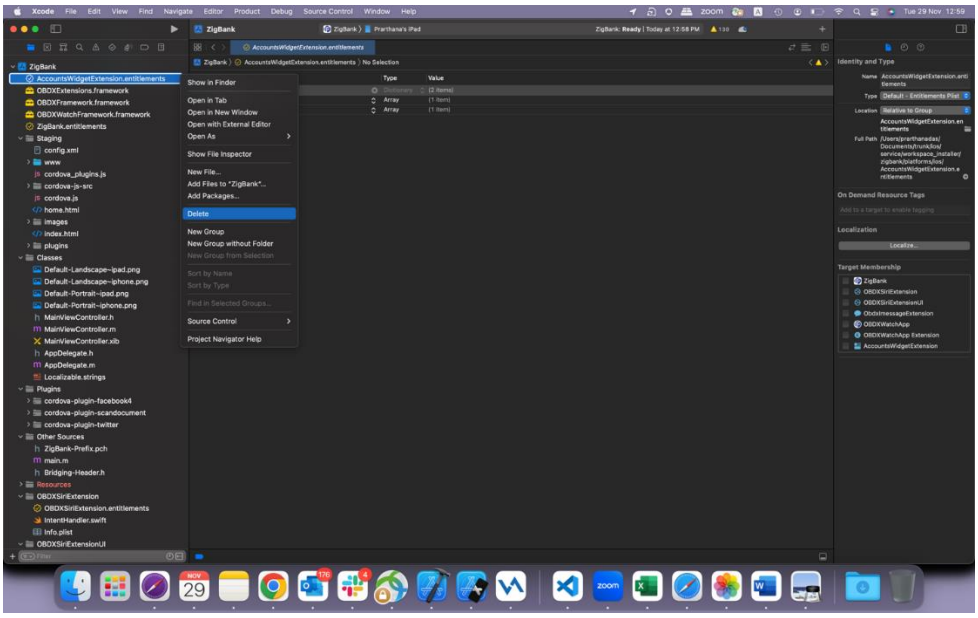
Removal of Widget functionality from workspace:

To remove the widget functionality from workspace please carry out the following steps:

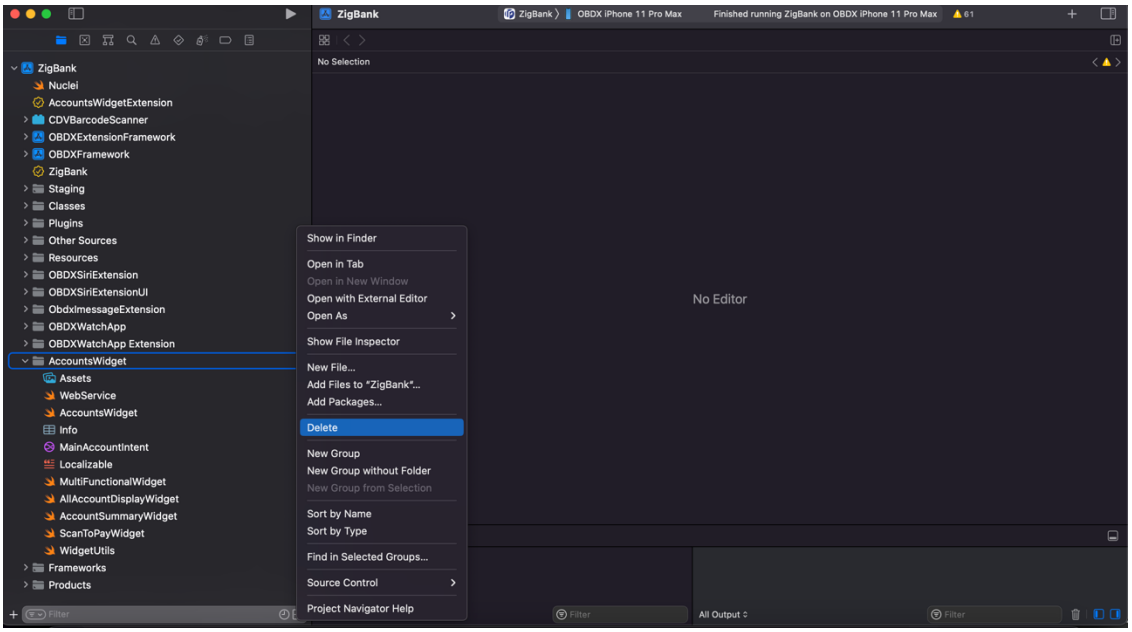
1. Please delete "AccountsWidgetExtension" from the "Targets" section.



2. Please delete "AccountsWidgetExtension.entitlements" file and "AccountsWidget" folder from Project navigator.



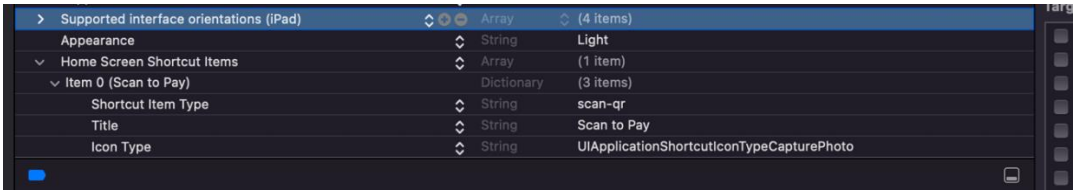
3. Delete “AccountsWidget” Folder as shown below



2.17 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.

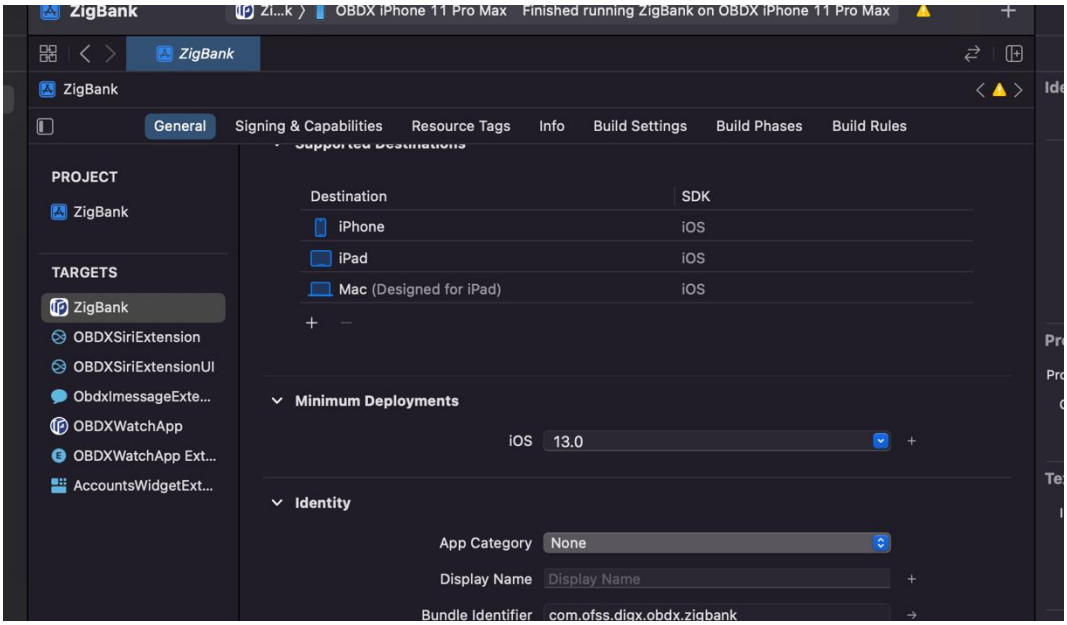
1. This option is not RTM controlled hence to remove this option if bank doesn't need it, then open Zigbank project in Xcode, open ZigBank-Info.plist. Delete entry for key – "Home Screen Shortcut Items."



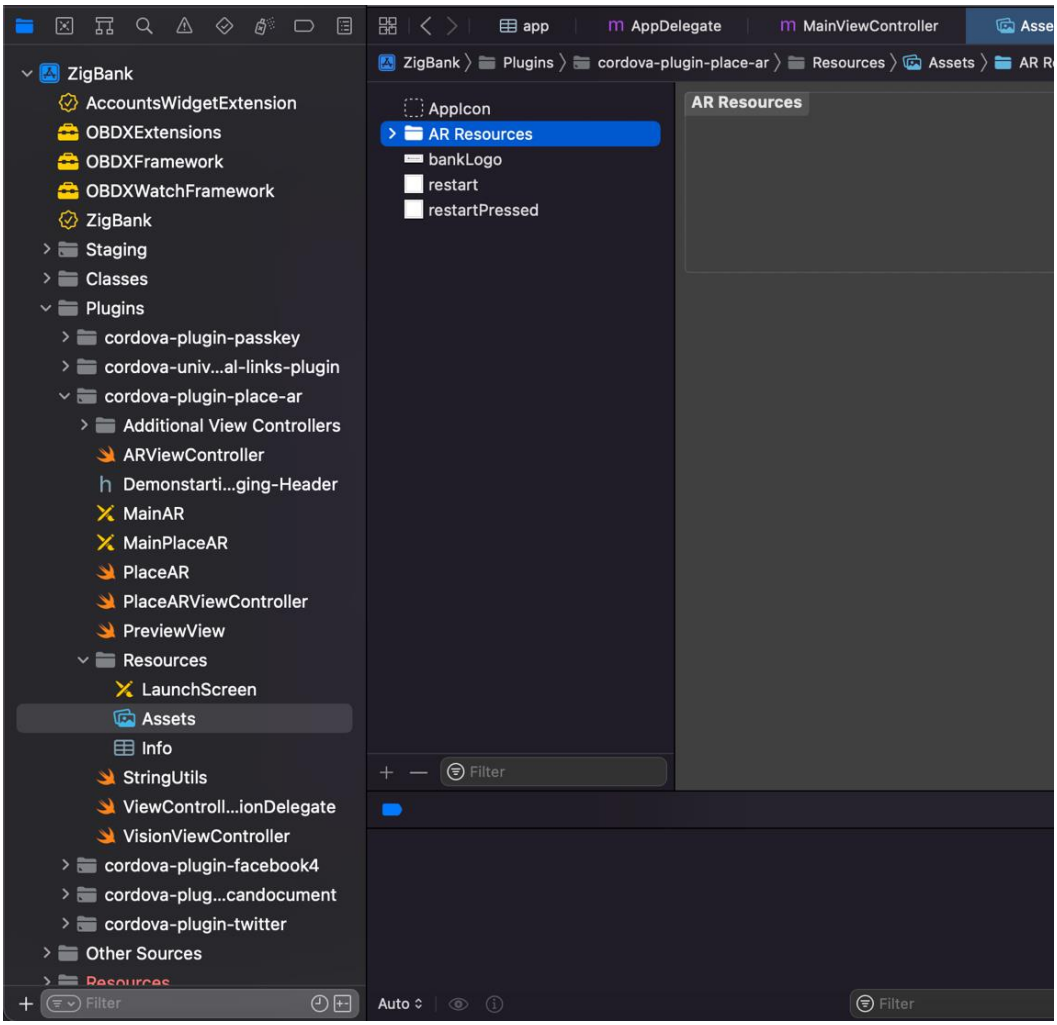
2.18 Scan Card using Augmented Reality

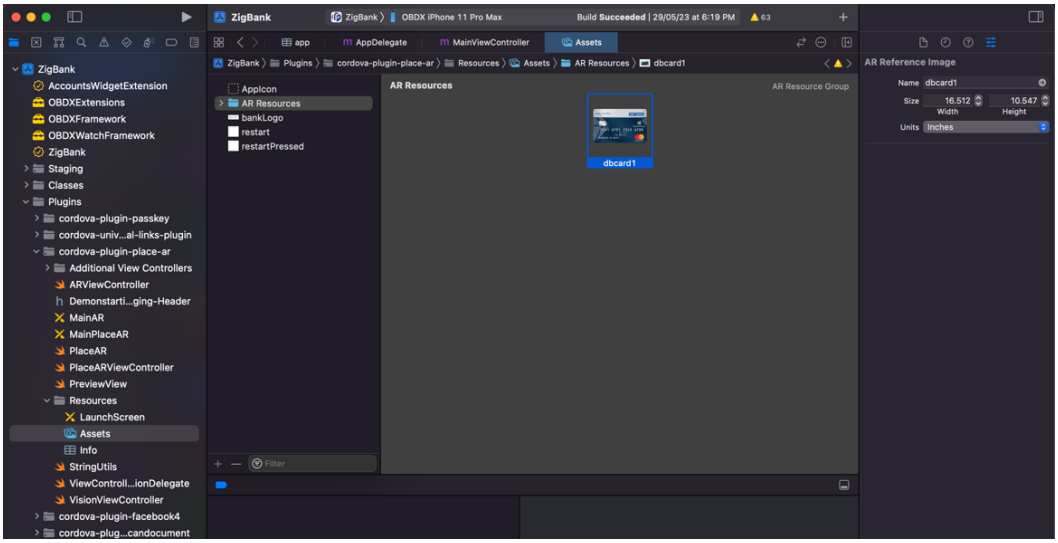
Users can scan card and view account details and transactions of the account associated with the card.

1. To use this feature, minimum deployment target should be iOS 13 as this feature is supported IOS 13 and above.
2. Open Zigbank project. Select target, and go top General Tab. Change minimum IOS version to 13.0



3. Bank needs to add sample card images in the Zigbank project so that OS recognizes it as a valid bank card. Keep bank's available card images ready in jpeg/png format.
4. Open Plugins->"cordova-plugin-place-ar" -> "Resources" -> Assets. Drag and drop sample bank card images inside "AR Resources" folder. Click on the image and open Attributes pane on right panel. Set width of the image in inches to the real world size of the card.


ZigBank
AccountsWidgetExtension
OBDXExtensions
OBDXFramework
OBDXWatchFramework
ZigBank
Staging
Classes
Plugins
cordova-plugin-passkey
cordova-univ...al-links-plugin
cordova-plugin-place-ar
Additional View Controllers
ARViewController
Demonstarti...ging-Header
MainAR
MainPlaceAR
PlaceAR
PlaceARViewController
PreviewView
Resources
LaunchScreen
Assets
Info
StringUtils
ViewControll...ionDelegate
VisionViewController
cordova-plugin-facebook4
cordova-plug...candocument
cordova-plugin-twitter
Other Sources
app
AppDelegate
MainViewController
ZigBank 〉 Plugins 〉 cordova-plugin-place-ar 〉 Resources 〉 Assets
AppIcon
AR Resources
bankLogo
restart
restartPressed
AR Resources
Filter
Auto


ZigBank
ZigBank 〉 OBDX iPhone 11 Pro Max
Build Succeeded | 29/05/23 at 6:19 PM
app
AppDelegate
MainViewController
Assets
ZigBank 〉 Plugins 〉 cordova-plugin-place-ar 〉 Resources 〉 Assets 〉 AR Resources 〉 dbcard1
AppIcon
AR Resources
bankLogo
restart
restartPressed
AR Resources
AR Resource Group
dbcard1
AR Reference Image
Name dbcard1
Size 16.512 Width 10.547 Height
Units Inches

5. “Scan your card” option is not RTM controlled so to remove this option, remove the entry from UI – components – docked-menu – pre-login.json. Remove entry for “scan-your-card”. There is no need to increase the minimum deployment target to 13.0 if you do not need this feature to be enabled for the users.

2.19 Passkey (Passwordless login)

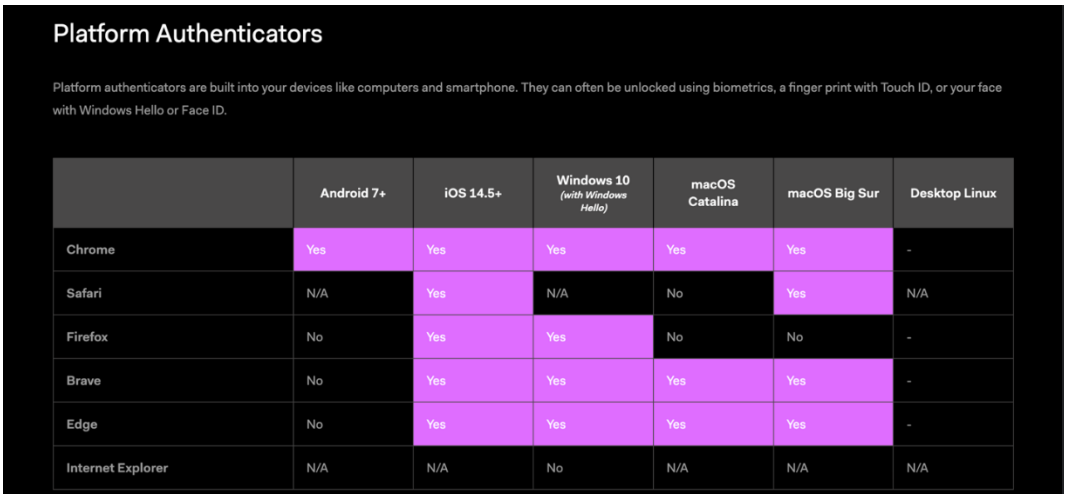
Passkey is passwordless login is introduced in latest IOS ecosystem (iOS devices, Safari) and Android devices and on latest browsers.which allows users to login securely without entering username and password.

IOS passkeys are shared in iCloud keychain. Hence iCloud Keychain should be enabled by the users on their apple ID on iOS device.

System requirements: Attached is the compatibility chart for passkeys to work:

Browsers should also be latest versions supporting WebAuthn protocols.

NOTE: Only one of two login types can exist in the application, Either passkey or alternate (PIN/Pattern/FaceId/touchID). So, setup this option only for first time application launch in the market. If the application is already launched and users have enabled alternate login, then enabling passkey option will hide the alternate login options from the application.



Platform Authenticators

Platform authenticators are built into your devices like computers and smartphone. They can often be unlocked using biometrics, a finger print with Touch ID, or your face with Windows Hello or Face ID.

	Android 7+	iOS 14.5+	Windows 10 *(with Windows Hello)*	macOS Catalina	macOS Big Sur	Desktop Linux
Chrome	Yes	Yes	Yes	Yes	Yes	-
Safari	N/A	Yes	N/A	No	Yes	N/A
Firefox	No	Yes	Yes	No	No	-
Brave	No	Yes	Yes	Yes	Yes	-
Edge	No	Yes	Yes	Yes	Yes	-
Internet Explorer	N/A	N/A	No	N/A	N/A	N/A

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) and 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

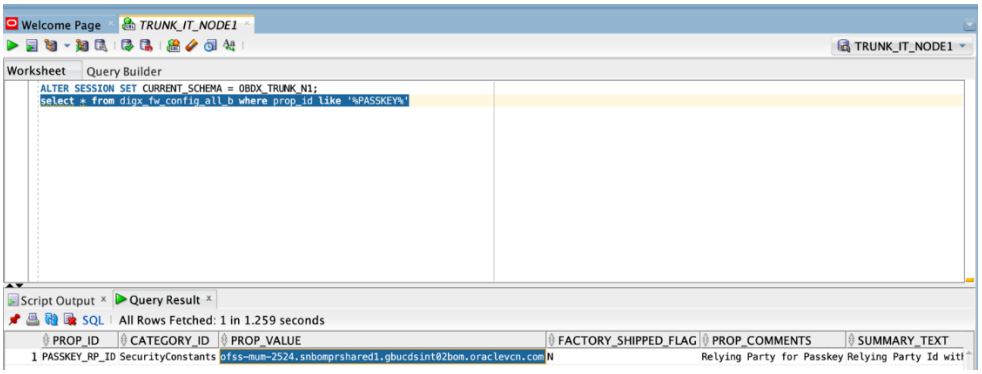
   thirdPartyAPIs -> passkey -> required -> true

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='PASSKEY_RP_ID'



2. Note – Relying partId is the domain name of the website to which credentials will be associated. (Eg google.com, example.com etc)
3. Hosting ASA file (Apple-app-site-association) on server.
   a. ASA- Apple App Site Association fille which IOS installs on the device when application is installed. This ASA file is hosted on our server for testing and then apple stores that file to its APPLE CDN when application is released on Appstore.
   b. This file is fetched by Apple after a duration of 5 days. So, any new update in the file takes 5 days to gets reflected in the application. In development mode though, every application installation, the ASA file is re-fetched on device.

c. If Bank doesn't want to set this up, do not follow below steps to setup ASA file. Also, open Zigbank project in XCode, Select Zigbank target -> Signing Capabilities -> Delete Associated domain.

There are two parts for the AASA file setup – Server side and application side.

Server-Side AASA Setup:

1. Create a json file and save it with name "apple-app-site-association" (without any extension not even ". json" extension is to be added). Copy the content from below. Update "appID", "appIDs", and "apps" value in below JSON to that of bank's appID and bundleID.

```
{
  "webcredentials":{
    "apps":[
      "3NXJ972C93.com.ofss.digx.obdx.zigbank"
    ]
  }
}
```

2. This file needs to be on https server with valid SSL certificate.
3. Update properties in digx-admin.war --> com.ofss.digx.app.sms.service.jar --> resources/IphoneApplink.properties

Below are the sample values for a single application supporting deeplink. Need tp update banks' teamID and bundle ID.

```
numberofapps=1
appid0=3NXJ972C93.com.ofss.digx.obdx.zigbank <Add bank's teamID.bundleID>
paths0=*
```

4. Need to change host and port in Obdx.conf

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

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

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

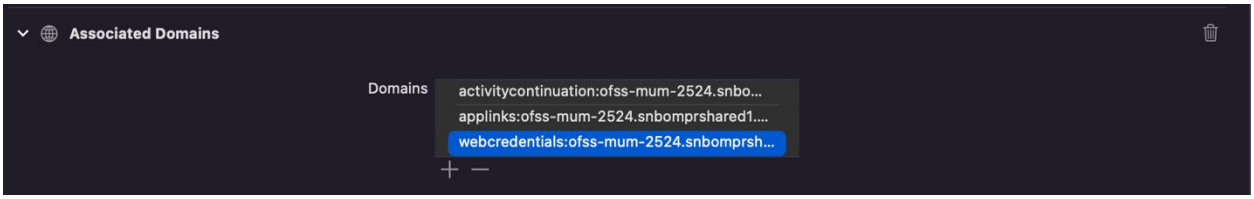
   https://<host>/.well-known/apple-app-site-association

Application side setup:

1. Open developer portal and enable Associated domain for your appID

2. Open Zigbank.workspace- Select Zigbank target. Go to Signing and Capabilities – In associated domain section, update the URL with bank's host for webcredentials key.

   Example. Replace ofss-mum-2524.snbomprshared1.gbucdsint02bom.oraclevcn.com?mode=developer with banks host where the ASA file is hosted. Port and "https" should not be added here.



Pease note – in webcredentials value "?mode=developer"
is only for development mode and testing on testflight. Hence for development with this mode, we can test only with developer profile.

Once app is ready for distribution to appStore, ?mode=developer should be removed while archiving for app store release.

How to test on device in development/testing phase.

a. Developer mode should be enabled on IOS device.

b. iCloud keychain should be enabled for appleID configured on the device. Settings – profile – iCloud – Passwords and Keychains – Sync this iPhone.

c. Go to Settings -> Passwords -> Password Option -> Check Auto fill option is enabled.

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

1. Deeplinking in IOS works with https url and a valid ASA configuration. Deeplinking keeps the application flow within the application when user clicks on bank's reset-password or claim-money link on email or message.

2. ASA- Apple App Site Association file which OS installs on the device when application is installed. This ASA file is hosted on our server for testing and then apple stores that file to its APPLE CDN when application is released on Appstore.
3. This file is fetched by Apple after a duration of 5 days. So, any new update in the file takes 5 days to gets reflected in the application. In development mode though, every application installation, the ASA file is re-fetched on device.
4. This is optional setup. If bank wants deep linking, then below steps to setup AASA file.
5. If Bank doesn't want to set this up, do not follow below steps to setup AASA file. Also, open Zigbank project in XCode, Select Zigbank target -> Signing Capabilities -> Delete Associated domain

AASA SETUP:

There are two parts for the setup – Server side and application side.

Server Side AASA Setup:

1. Create a json file and save it with name "apple-app-site-association" (without any extension not even .json extension is to be added). Copy the content from below. Update "appID", "appIDs", and "apps" value in below JSON to that of bank's appID and bundleID.

```
{
  "applinks":{
    "apps":[


    ],
    "details":[
      {
        "appID":"3NXJ972C93.com.ofss.digx.obdx.zigbank",
        "appIDs":[
          "3NXJ972C93.com.ofss.digx.obdx.zigbank"
        ],
        "components":[
          {
            "comment":"Match",
            "/":"*"
          }
        ],
```

```
        "paths":[
          "*"
        ]
      }
    ]
  },
  "activitycontinuation":{
    "apps":[
      "3NXJ972C93.com.ofss.digx.obdx.zigbank"
    ]
  }
}
```

2. 3NXJ972C93.com.ofss.digx.obdx.zigbank In this 3NXJ972C93 is the team Id and com.ofss.digx.obdx.zigbank is bundle identifier. So the format is <TEAM_ID>.<bundleIdentifier>
3. Team ID is present in developer account in membership details
4. This file needs to be hosted on https server with valid SSL certificate. There should be no rediection to this file.
5. Update properties in digx-admin.war --> com.ofss.digx.app.sms.service.jar --> resources/IphoneApplink.properties

   Below are the sample values for a single application supporting deeplink. Bank should update banks' teamID and bundle ID.

   numberofapps=1

   appid0=3NXJ972C93.com.ofss.digx.obdx.zigbank <Add bank's teamID.bundleID>

   paths0=*

6. Need to change host and port in Obdx.conf

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

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

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

   https://<host>/.well-known/apple-app-site-association

Application side setup:

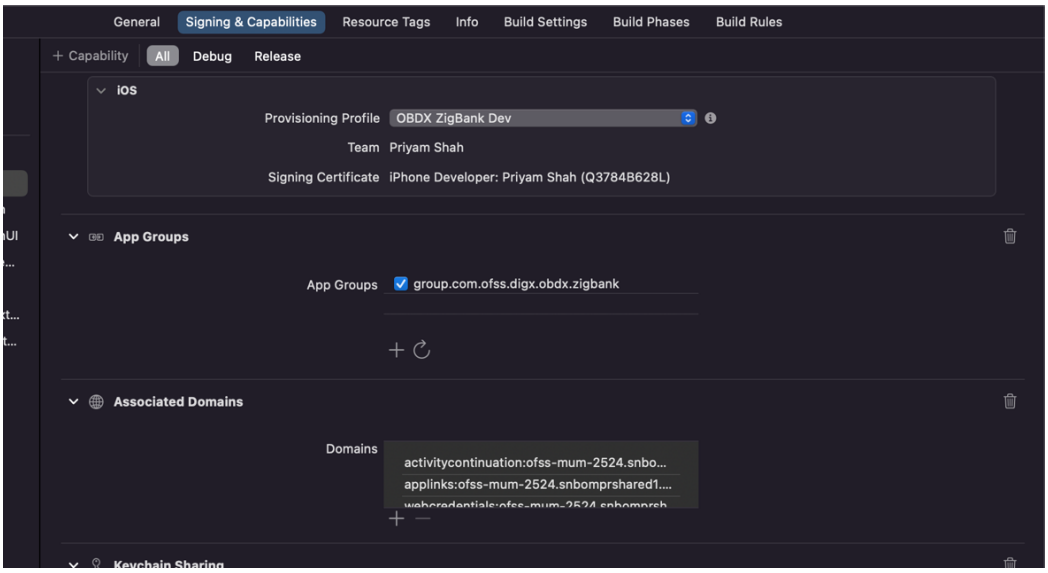
1. Open developer portal and enable Associated domain for your appID

2. Open Zigbank.workspace->Select Zigbank target. Go to Signing and Capabilities – In associated domain section, update the URL with bank's host for activitycontinuation and applinks

   Example. Replace ofss-mum-2524.snbomprshared1.gbucdsint02bom.oraclevcn.com?mode=developer with banks host where the ASA file is hosted. No port and https to be added here.

   Pease note – in applinks and activitycontinuation "?mode=developer"
   is only for development mode and testing on TestFlight. Hence for development with this mode, we can test only with developer profile.

   Once app is ready for distribution to Appstore and TestFlight, ?mode=developer should be removed.



3. Update the key_server_url to https URL in the Zigbank project app.plist

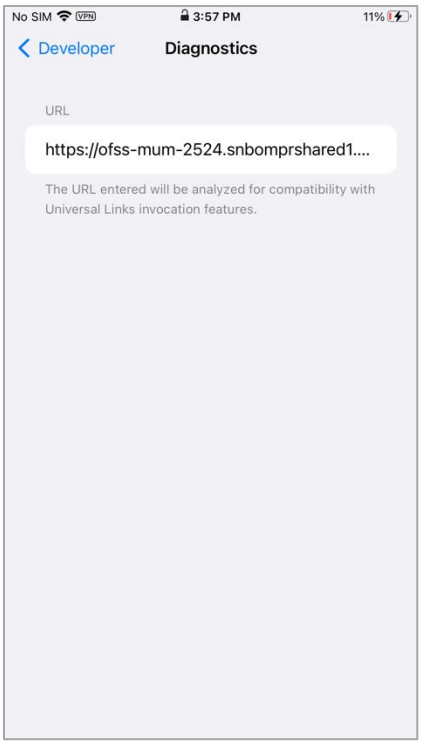
Device Side setup for development and testing:

1. To test on device, Developer mode should be enabled. Additionally, go to Phone Settings –

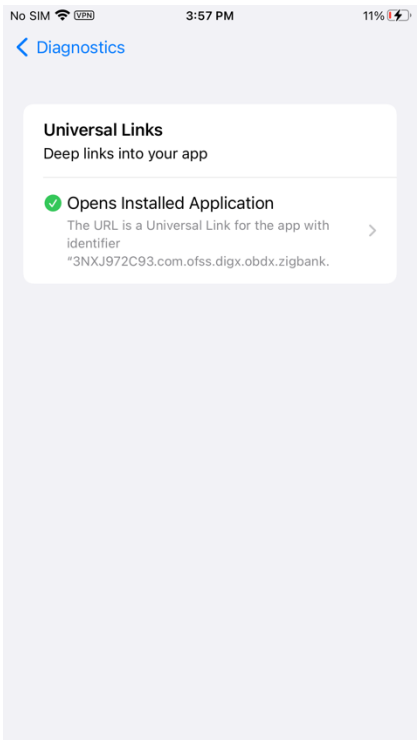
   Developer mode- > Enable “Associated domain Development”.

2. With all above setup, install the application on the device. Please not while installing the device must be connected to network in which the ASA file is accessible.
3. Under Settings-Developer Option – Go to Diagnostics - > Add your server URL like below and check if device can identify this link as deeplink. If all setup is correct and ASA file is successfully installed on device, this will display a valid URL as below:

   Example: In screenshot below, we have added our server URL which is also the URL where ASA file is hosted.
   https://ofss-mum-2524.snbomprshared1.gbucdsint02bom.oraclevcn.com/



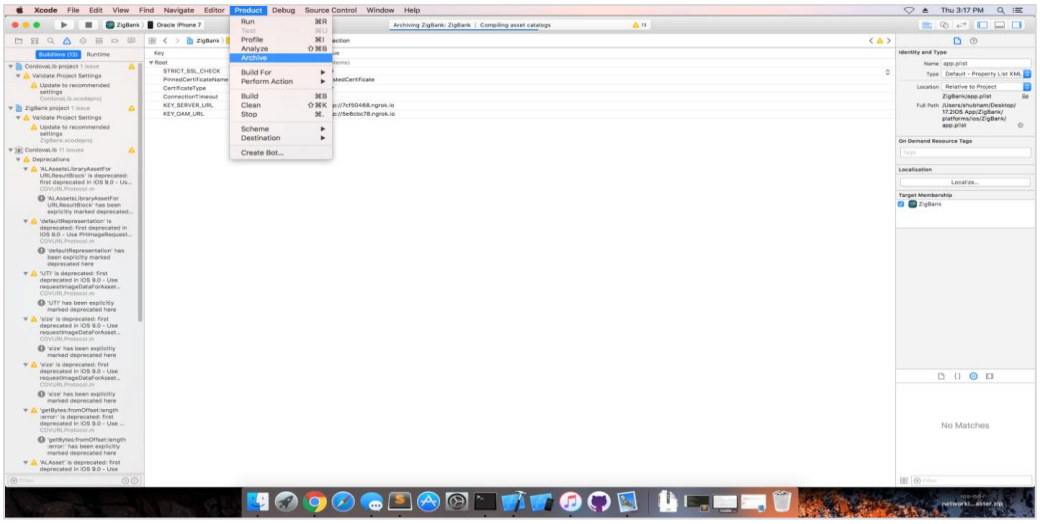
If we see below message, then deeplink can be tested on this device.



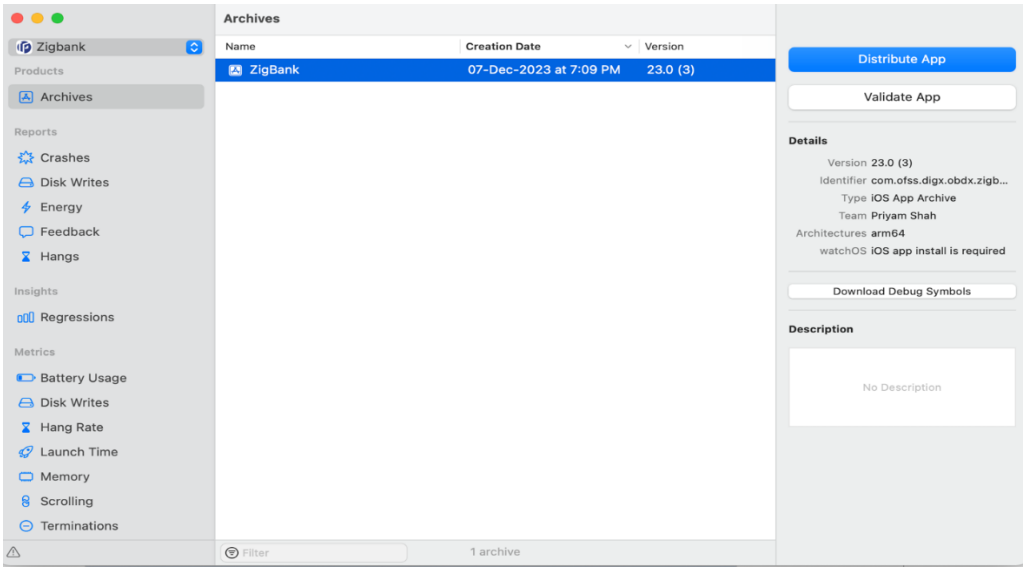
4. Send the link for reset-password/claim money in mail or copy the link and save the link in phone's notepad. The link should be a https URL where the ASA is hosted and should not contain port.
5. Long press on the link and you must see "Open In Zigbank App" option. Clicking the option page opens in the application.

3. Archive and Export

a. In the Menu bar click on **Product -> Archive (Select Generic iOS Device)**



b. After archiving has successfully completed. Following popup will appear.



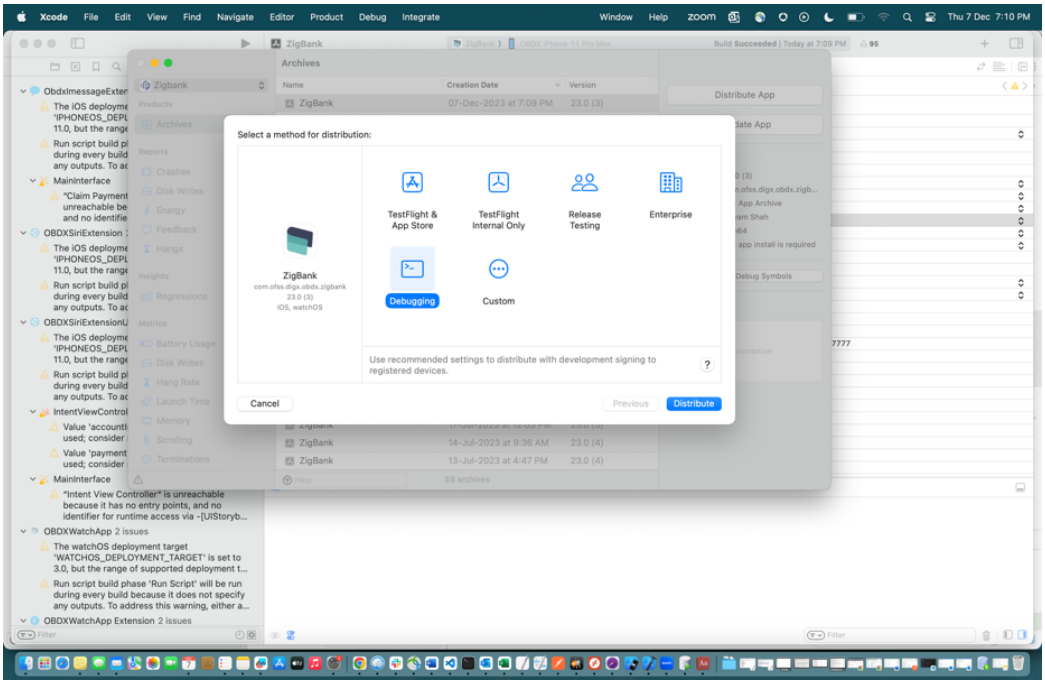
c. Click on Distribute App in the right pane of the popup -> select **the Method of Distribution -> Select Distribute.** Review the contents and click on **Export -> Export** and generate the .ipa

There are multiple options for exporting, select according to what is needed.

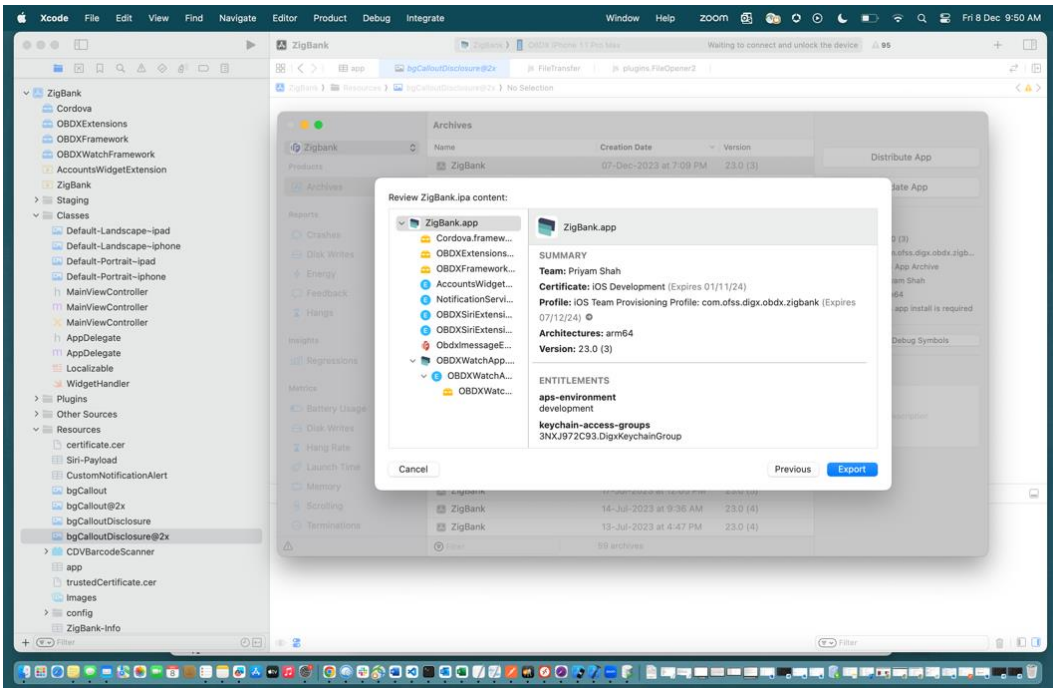
Debugging – this will create an ipa with development profile for internal testing.

Release Testing – This will create an ipa with Ad Hoc distribution profile for adHoc testing.

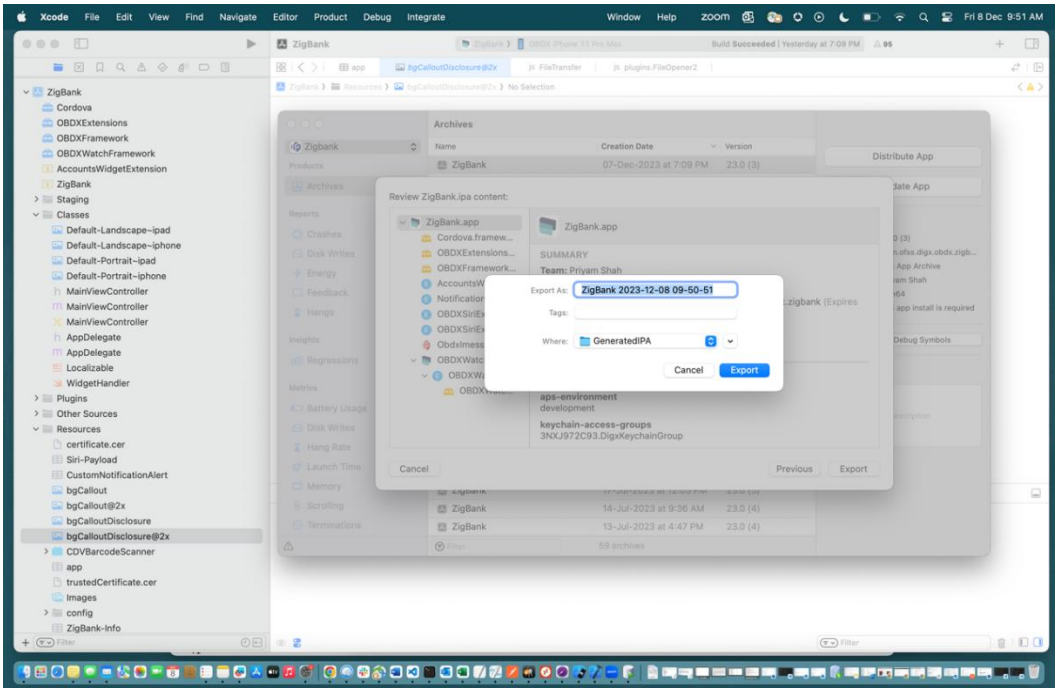
TestFlight Internal Only , TestFlight & AppStore- As the name suggests, this is for testflight and AppStore release.



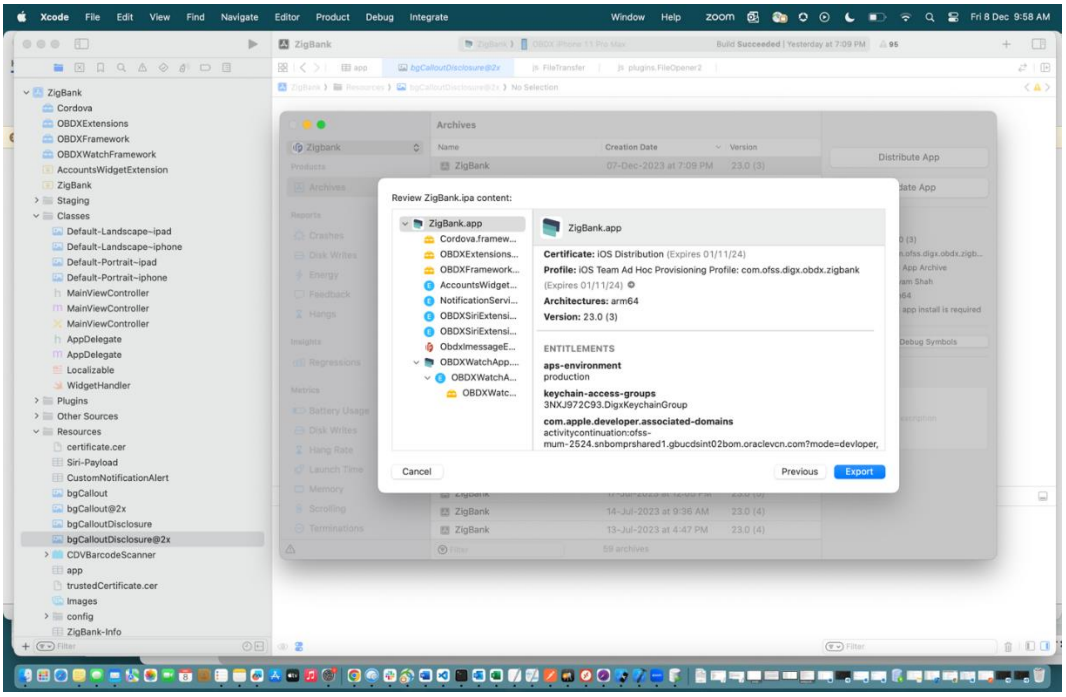
This is window which appear after selecting Debugging option. Note the Certificate and provisioning profile is for development.



Click on Export and it will ask to save the ipa. Select the location and click on Export. This ipa will be development ipa which can be installed on devices which are added in the profiles on developer account. Refer Section for more details ()



Below is the window which appears after selecting “Release Testing”. Note here the Certifiate and Profile is of Adhoc Distribution.



Follow the above steps to Export and save the ipa. This ipa will be adhoc distribution ipa.

NOTE: The frameworks built are compatible with devices and simulators both. So, the application can be run on the simulator directly without copying any frameworks.

Home

4. OBDX Authenticator Application (Futura Secure)

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”

    Prerequisite:

    - Download and Install node js as it is required to run npm and cordova commands.
    - Latest XCode to be download from Mac App Store. This document is w.r.t to XCode 15.0
    - Authenticator Application is supported only on current iOS version and only one version preceding that.

4.1 Authenticator UI (Follow any one step below)

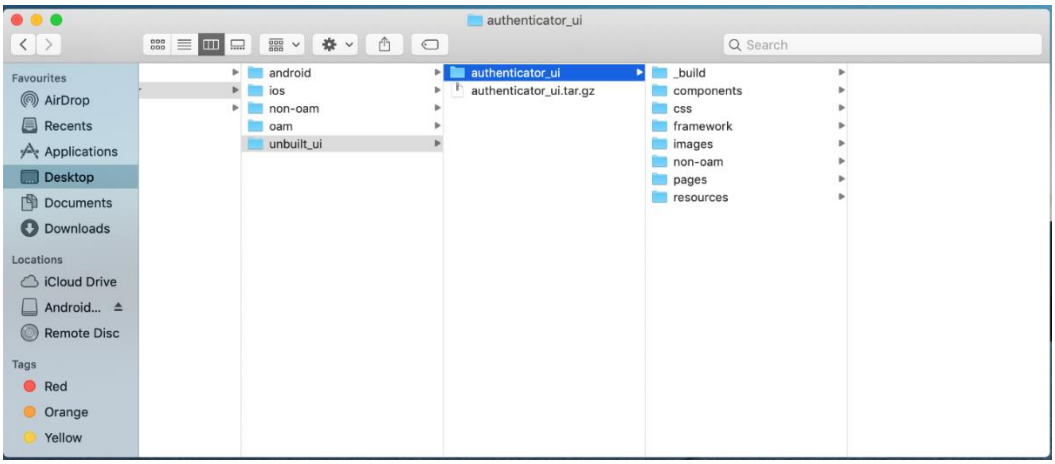
4.1.1 Using built UI

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

4.1.2 Building UI manually

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

The folder structure is as shown:



a. Token Based Authentication Mechanism

   a. Copy the "*token-based/login*" folder and replace it at the "components/modules/" [in ui folder] location. This will replace the existing the login folder.

   b. Open the terminal at "_build" level.

   c. Run the following commands:

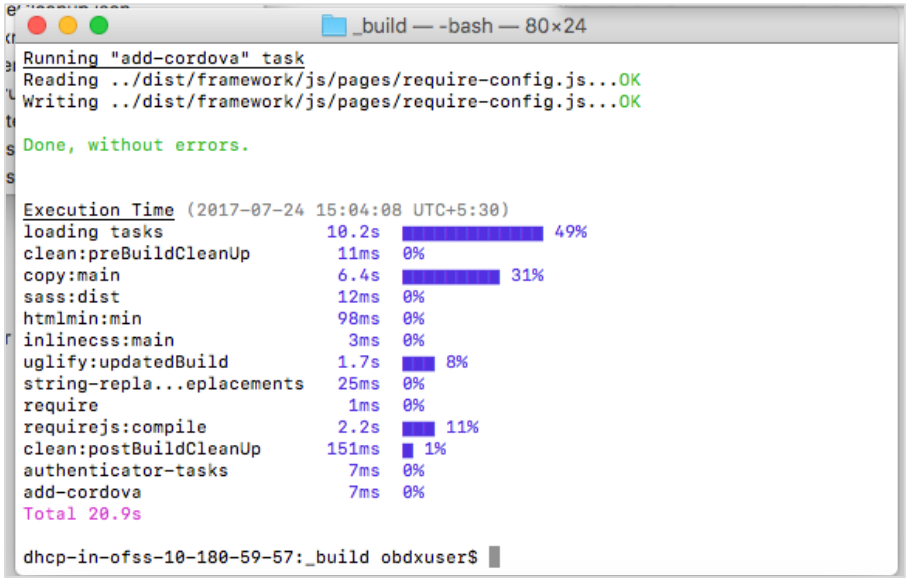
```
sudo npm install -g grunt-cli

sudo npm install

node render-requirejs/render-requirejs.js

grunt authenticator --verbose
```

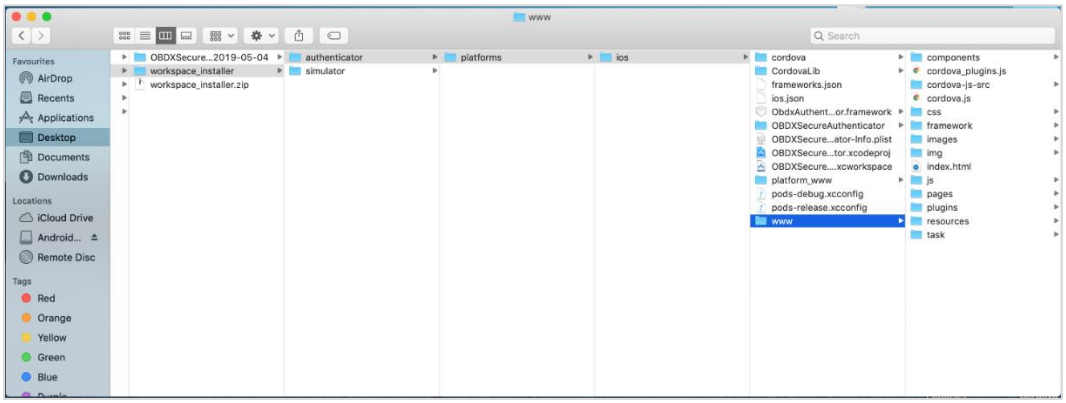
   d. After running above commands and getting result as "*Done, without errors.*" A new folder will be created at "_build" folder level with name as "dist".



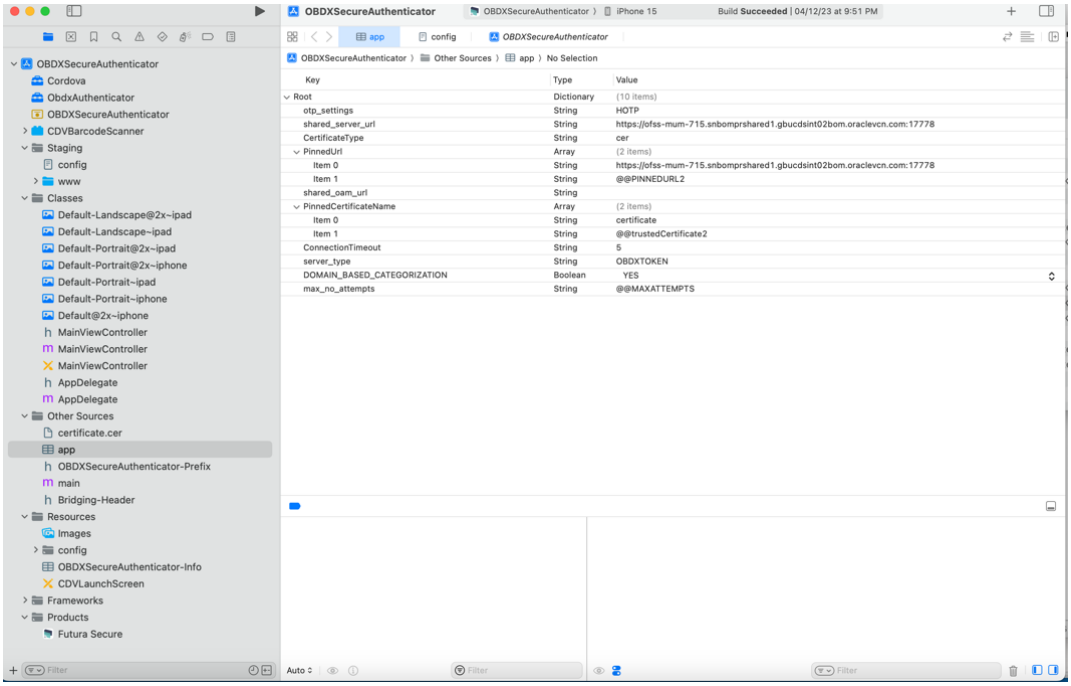
4.2 Authenticator Application Workspace Setup

1. Unzip and navigate to iOS workspace as shipped in installer.
2. Open the workspace as shown below and find and replace the following generated UI files from "*ui/dist*" folder :
   - components
   - css
   - framework
   - images
   - pages

- resources



3. Double click on OBDXSecureAuthenticator.xccodeproj to open the project in Xcode



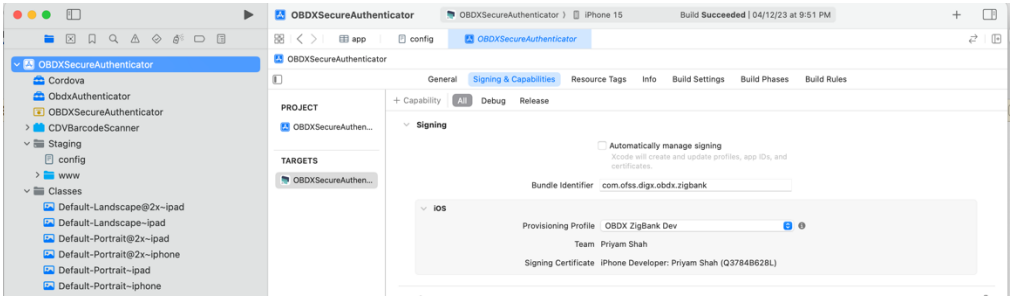
4. Update HOTP or TOTP in above screenshots and update the server URL.

5. Set value of max_no_attempts to value greater than 0.

6. Create certificates and profiles on Apple Developer account, Use the bundle identifier and set appropriate profile in the application.

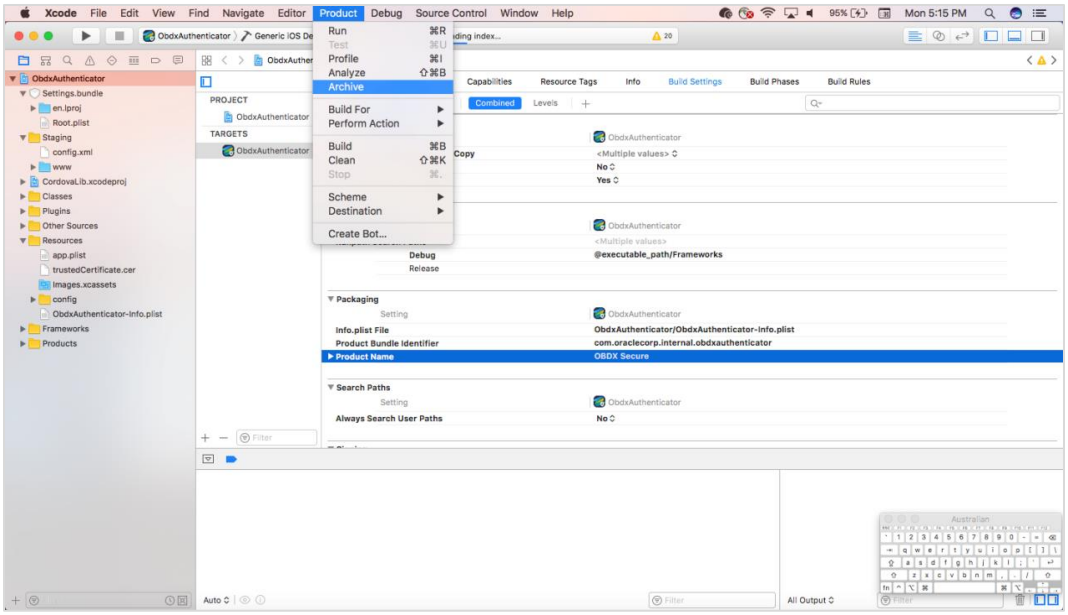
1. Adding bundle identifiers:

7. Bundle identifiers needs to be added in the Info.plist of each the frameworks
8. For example, let us assume that the bundle identifier used is abc.def.ghi.jkl. The steps to be followed are,
    a. Right click on ObdxAuthenticator.xcframework(in Xcode's Project Navigator) -> Show in Finder
    b. When the Finder directory click on ios-arm64 folder-> ObdxAuthenticator.framework.
    c. Open Info.plist and set Bundle identifier as abc.def.ghi.jkl.ObdxAuthenticator
    d. Follow same for Cordova.xcframework and set Bundle identifier for Cordova.framework : abc.def.ghi.jkl.Cordova
9. Also, set the identifiers and select appropriate profile in the target -> Signing & Capbilities tab as show below:
10. The application contains frameworks for devices and simulator both. Run the application directly on simulator without copying any frameworks.
11. The application can be archived using steps in Section 4.3 for running on device.

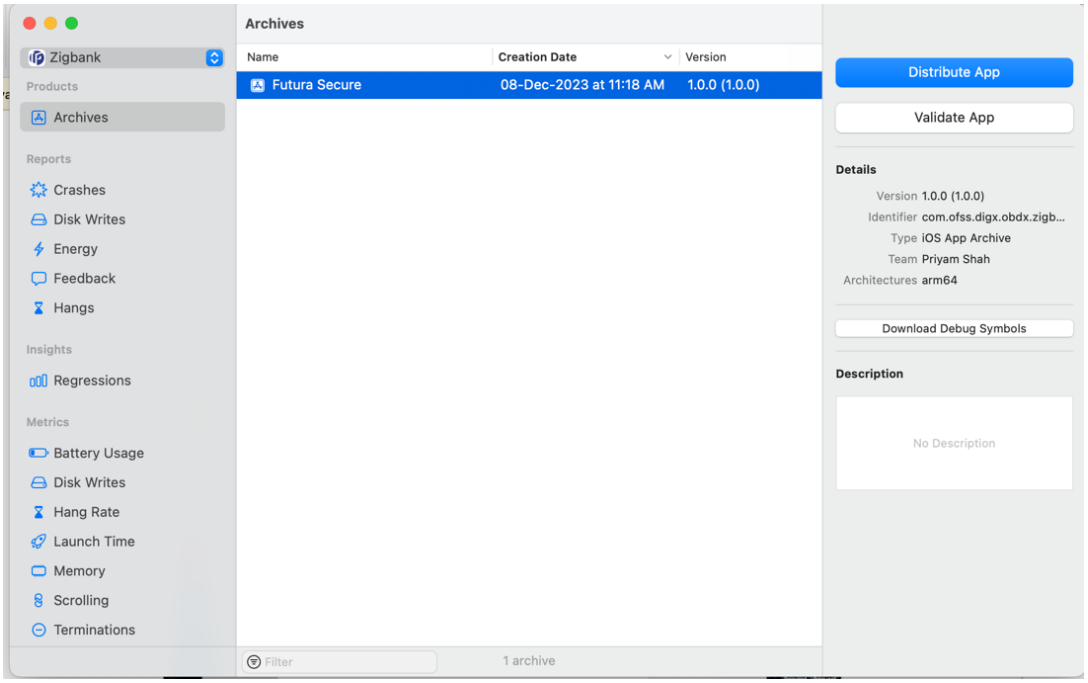


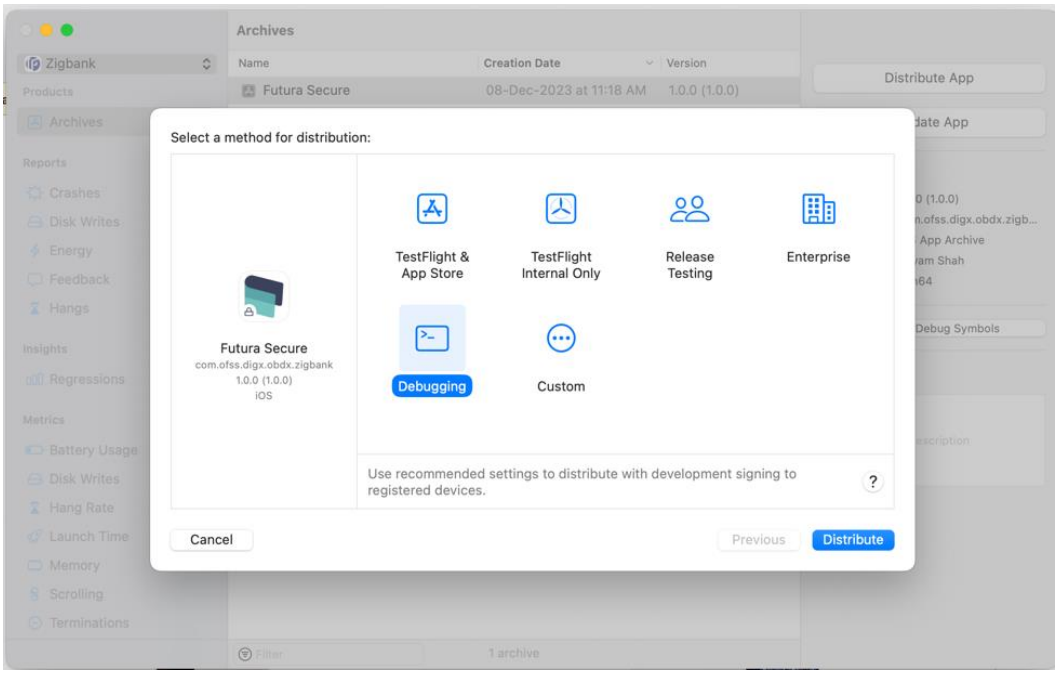
4.3 Archiving Authenticator Application

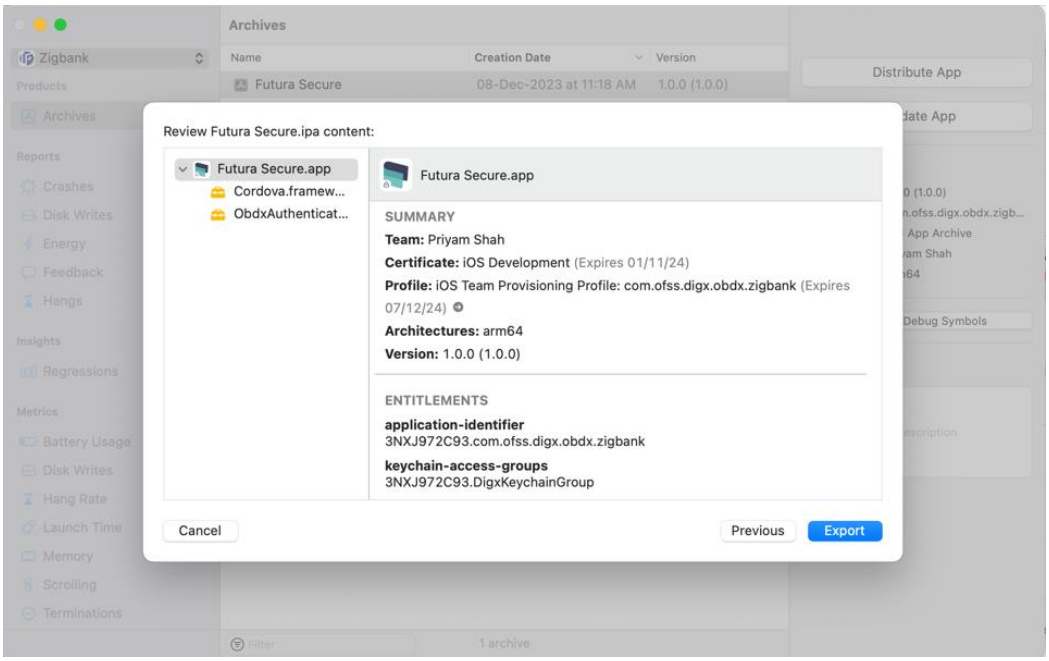
1. Set the device selection to *Generic iOS* device. Then go to *Product -> Archive*.



2. Choose your Archive and then click "*Export*".ipa file will be generated with Name "Futura Secure"




Archives
Zigbank
Name
Creation Date
Version
Futura Secure
08-Dec-2023 at 11:18 AM
1.0.0 (1.0.0)
Distribute App
Select a method for distribution:
TestFlight & App Store
TestFlight Internal Only
Release Testing
Enterprise
Futura Secure
com.ofss.digx.obdx.zigbank
1.0.0 (1.0.0)
iOS
Debugging
Custom
Use recommended settings to distribute with development signing to registered devices.
Cancel
Previous
Distribute


Archives
Zigbank
Name
Creation Date
Version
Futura Secure
08-Dec-2023 at 11:18 AM
1.0.0 (1.0.0)
Distribute App
Review Futura Secure.ipa content:
Futura Secure.app
Cordova.framew...
ObdxAuthenticat...
Futura Secure.app
SUMMARY
Team: Priyam Shah
Certificate: iOS Development (Expires 01/11/24)
Profile: iOS Team Provisioning Profile: com.ofss.digx.obdx.zigbank (Expires 07/12/24)
Architectures: arm64
Version: 1.0.0 (1.0.0)
ENTITLEMENTS
application-identifier
3NXJ972C93.com.ofss.digx.obdx.zigbank
keychain-access-groups
3NXJ972C93.DigxKeychainGroup
Cancel
Previous
Export

4.4 Using SSL in Authenticator App:

Follow below steps to setup SSL in Authenticator application:

Open Authenticator application project - > app.plist. Add below changes.

1. **shared_server_url_url** = <bank's https url>
2. **PinnedUrl** – Item 0 – replace @@PINNEDURL1 with your https url (without port)
3. Open bank's https site on browser, click on the lock icon and "Show certificate", Select the certificate icon and Drag and drop the certificate from Safari to local machine. Rename to certificate.cer.
4. Open Authenticator application, right lick on Resources folder -> Add Files to Authenticator" -> Select the downloaded certificate -> Select "Copy Items if Needed" -> And target selected as below:
5. **PinnedCertificateName** – Item 0 – replace @@trustedCertificate1 with certificate name. Example your certificate name added to your Project is certificate.cer, then @@trustedCertificate1 should be replaced with "certificate"
6. **Server_type** - OBDXTOKEN
7. **DOMAIN_BASED_CATEGORIZATI**ON – YES

Home