

Mobile Application Builder Guide-iOS Guide
Oracle Banking Digital Experience
Patchset Release 22.2.6.0.0

Part No. F72987-01

April 2025

Mobile Application Builder Guide-iOS Guide

April 2025

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax: +91 22 6718 3001

www.oracle.com/financialservices/

Copyright © 2006, 2025, Oracle and/or its affiliates. All rights reserved.

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. 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 failsafe, 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.

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.

This software or hardware and documentation may provide access to or information on 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. 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.



Table of Contents

1. Preface	1-1
1.1 Purpose	1-1
1.2 Audience	1-1
1.3 Documentation Accessibility	1-1
1.4 Critical Patches	1-1
1.5 Diversity and Inclusion	1-1
1.6 Conventions	1-1
1.7 Screenshot Disclaimer	1-2
1.8 Acronyms and Abbreviations	1-2
2. OBDX Servicing Application.....	2-1
2.1 Prerequisite.....	2-1
2.2 Project Setup	2-1
2.3 Create Project Using Remote UI	2-1
2.4 Create Project by hosting UI on local server machine	2-2
2.5 Configurations for the IOS application	2-4
2.6 Enabling SSL pinning in the application	2-13
2.7 Enabling Force update	2-15
2.8 Device Registration and Push Registration Functionality.....	2-16
2.9 Generating Certificates for Development, Production	2-17
2.10 Setup for Push Notification in the application	2-21
2.11 Push Notification Actionable Alerts Configuration	2-23
2.12 Push Notification 2FA configuration	2-24
2.13 ODA Chatbot Inclusion	2-25
2.14 Widget Functionality	2-28
2.15 Scan to Pay from Application Icon.....	2-30
2.16 Passkey (Passwordless login)	2-30
2.17 Deep linking - To open reset password, claim money links with the application	2-34
2.18 Changing App Icons and Assets	2-1
2.19 Archive and Export	2-1
3. OBDX Authenticator Application (Futura Secure).....	3-1
3.1 Authenticator UI (Follow any one step below)	3-1
3.2 Authenticator Application Workspace Setup	3-2
3.3 Archiving Authenticator Application	3-5

3.4	Using SSL in Authenticator App	3-7
4.	Apple Privacy	4-1
5.	Make IOS Application Ready for Production Checklist	5-4

1. Preface

1.1 Purpose

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

1.2 Audience

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

1.3 Documentation Accessibility

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

Access to Oracle Support

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

1.4 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](#).

1.5 Diversity and Inclusion

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

1.6 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.

1.7 **Screenshot Disclaimer**

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

1.8 **Acronyms and Abbreviations**

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

Abbreviation	Description
OBDX	Oracle Banking Digital Experience

2. OBDX Servicing Application

This is the main application for mobile banking.

2.1 Prerequisite

- Download and Install node as it is required to run npm and Cordova commands.
- Latest Xcode to be download from App Store. This document is w.r.t to Xcode 16.2
- OBDX iOS Application is supported on current iOS version and only one version preceding that.

2.2 Project Setup

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 fat xcframeworks for running on devices and simulator both. The same frameworks within the workspace can be used to run on simulator and device as well.
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 on local server machine. 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/server IP address.
5. To enable debugging in network tab, set this property "InspectableWebView" to true in config.xml
6. Also, refer section **Configurations for the IOS application**: for configurations required for the application.
7. Also, apple certificates and provisioning profiles are needed to run the application on devices. Refer section: **Generating Certificates for Development, Production.**

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 hosted.

Proceed to section **Configurations for the IOS application**:

2.4 Create Project by hosting UI on local server machine

a. Building un-built UI:

1. UI is built with webpack hence the built UI cannot be directly modified. Hence, bank can use unbuilt UI and make the necessary changes and host the UI is below 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 OHS 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.
 - a. Bank can follow the UI build steps from “Oracle Banking Digital Experience User Interface Guide”.
3. Additionally, 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',  
},
```

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

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


```

ssakpal@ssakpal-mac channel % npm start

> obdc-build-tool@20.1.0 start
> webpack serve --open --config scripts/webpack/webpack.dev.js

[webpack-dev-server] [HMR] Proxy created: /digx -> http://efss-mum-715.snbomgrshared1.gbucloudsint02bon.oracleevcn.com:17777/
[webpack-dev-server] Project is running at:
[webpack-dev-server] Loopback: http://localhost:4000/
[webpack-dev-server] On Your Network (IPv4): http://192.168.29.50:4000/
[webpack-dev-server] On Your Network (IPv6): http://[fe80::1]:4000/
[webpack-dev-server] Content not from webpack is served from '/Users/ssakpal/Documents/work/svn/trunk/core/channel_11Sept/channel/dist' directory
[webpack-dev-middleware] wait until bundle finished: /

```

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

```

Critical dependency: require function is used in a way in which dependencies cannot be statically extracted
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/ sync ^\.\.*$ ./min/ojmodule-element-utils ./min/ojmodule-element-utils.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojthemamap.js 2617:47-149
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ sync ^\.\.*$ ./ojthemamap ./ojthemamap.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojconfig.js 139:51-152
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojtranslation.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojconverterutils-i18n.js
@ ./framework/js/dom-util.js 6:0-61 446:15-58 643:0-669:2
@ ./framework/js/view-model/generic-view-model.js 2:0-49 29:4-17 50:5-20 56:5-20 62:5-20 84:5-26 165:21-28
@ ./framework/js/configurations/require-config.js 20:4-56

WARNING in ./node_modules/@oracle/oraclejet/dist/js/libs/oj/min/ojmodule-element-utils.js 0:950-945
Critical dependency: require function is used in a way in which dependencies cannot be statically extracted
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/ sync ^\.\.*$ ./min/ojmodule-element-utils ./min/ojmodule-element-utils.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojthemamap.js 2617:47-149
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ sync ^\.\.*$ ./ojthemamap ./ojthemamap.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojconfig.js 139:51-152
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojtranslation.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojconverterutils-i18n.js
@ ./framework/js/dom-util.js 6:0-61 446:15-58 643:0-669:2
@ ./framework/js/view-model/generic-view-model.js 2:0-49 29:4-17 50:5-20 56:5-20 62:5-20 84:5-26 165:21-28
@ ./framework/js/configurations/require-config.js 20:4-56

WARNING in ./node_modules/@oracle/oraclejet/dist/js/libs/oj/min/ojmodule.js 8:2000-2007
Critical dependency: require function is used in a way in which dependencies cannot be statically extracted
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/ sync ^\.\.*$ ./min/ojmodule ./min/ojmodule.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojthemamap.js 2617:47-149
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ sync ^\.\.*$ ./ojthemamap ./ojthemamap.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojconfig.js 139:51-152
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojtranslation.js
@ ./node_modules/@oracle/oraclejet/dist/js/libs/oj/debug/ojconverterutils-i18n.js
@ ./framework/js/dom-util.js 6:0-61 446:15-58 643:0-669:2
@ ./framework/js/view-model/generic-view-model.js 2:0-49 29:4-17 50:5-20 56:5-20 62:5-20 84:5-26 165:21-28
@ ./framework/js/configurations/require-config.js 20:4-56

7 warnings have detailed information that is not shown.
Use 'stats.errorDetails: true' resp. '--stats-error-details' to show it.

webpack 5.89.0 compiled with 27 warnings in 12461 ms

```

- 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 OHS development server machine which is accessible in the network. They “key_server_url” will then point to that local server machine URL.

2.5 Configurations for the IOS application

Application level configurations are present in 'app.plist' (ZigBank/Resources) of the workspace

Note: These are configurations for different features. The description of each is 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.

1. For SERVER_URLs:

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

This is mandatory configuration.

Add URLs to app.plist (ZigBank/Resources)

OBDXTOKEN (Token based mechanism)

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

2. For SIRI:

1. This configuration is Optional.
2. By default, SIRI capability is set to NO. Bank can enable it to add SIRI feature for mobile banking.

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 only 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
------------------	--

1. Siri-Payload.plist

- This is present in ZigBank/Resources folder inside IOS workspace.
- It is provided to specify entries in the Siri payload based on transaction types (internal, domestic).
- 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.
- By default, SIRI will work with the given payload so no need to change anything in it.

3. To Enable SSL:

- Refer section **Enabling SSL pinning in the application** on how to configure the workspace to enable SSL pinning in the application.
- By default, SSL pinning is NO in the workspace.
Recommended to set to YES for production URLs with a valid authorized SSL certificate on server.

SSLPinningEnabled	Type: Boolean Purpose: To enable SSL Pinning. SSL checks are performed on application launch. Value: YES, for enabling. NO for disabling. Configurable: Yes
CertificateType	Type: String Purpose: File extension of SSL Pinned certificates Value: cer Note: the certificate file added in the workspace should also have .cer extension Configurable: Yes
PinnedUrl	Type: Array

	<p>Purpose: Pinning URL to be entered here. This is the https URL of the server against which the certificate will be verified. Can add multiple if required.</p> <p>Value: https server URL</p> <p>Configurable: Yes</p>
PinnedCertificateName	<p>Type: Array</p> <p>Purpose: For verification of SSL, this certificate will be pinned in the application and verified against the server URL.</p> <p>Value: Houses the certificate name (without extension) of the pinning certificate. Old certificate (about to expire) and new one can co-exist.</p> <p>Configurable: Yes</p>
SSLPinningEnabledNoNetworkCall	<p>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.</p> <p>If set to YES and SSLPinningEnabled is set to YES then if SSL Pinning fails, then login page does not load.</p> <p>If set to NO and SSLPinningEnabled is set to YES then if SSL Pinning fails, then login page loads.</p> <p>Configurable: Yes</p>
EnableSSLPinningForEveryRequest	<p>Type: Boolean</p> <p>Purpose: To enable SSL Pinning for every request fired from application pages in the entire application.</p> <p>Value: YES, for enabling. NO for disabling.</p> <p>Configurable: Yes</p>

4. To Enable Force Update:

This is an optional configuration.

Refer section [Enabling Force update](#) on more details on how to configure the workspace for this.

ForceUpdate	<p>Type: Boolean</p> <p>Purpose: To enable force update feature in the application.</p> <p>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</p>
-------------	---

	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 AppStore redirection on click of update button. 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. Do not change the URL.
itunesUrlForVersionCheck	Type: String Purpose: URL to check application version 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. Do not change the URL.

5. WATCH Application parameters:

Applicable only if Watch target is added in the workspace.

These are optional configurations.

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	<p>Type: Boolean</p> <p>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.</p> <p>Value: YES, to display the option. NO to hide that option.</p> <p>Configurable: Yes</p>
WatchLocateUs	<p>Type: Boolean</p> <p>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.</p> <p>Value: YES, to display the option. NO to hide that option</p> <p>Configurable: Yes</p>
WATCHMAXATTEMPTS	<p>Type: Number</p> <p>Purpose: The number of time PIN login is allowed in the watch application</p> <p>Value: Default is 3. Change the value to which Bank wants to restrict the PIN invalid attempts. After attempts exhaust, user will be asked to register the application PIN again.</p> <p>Configurable: Yes</p>

6. For Displaying Maintenance Page:

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	<p>Type: Boolean</p> <p>Purpose: To display a maintenance page if server is down.</p> <p>Value: true to display. If bank doesn't need any page to be displayed, it can be set to false. Default is true</p> <p>Configurable: Yes</p>
-----------------------	--

MAINTENANCE_PAGE_STATUS_CODE	<p>Type: Array</p> <p>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.</p> <p>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.</p> <p>Configurable: Yes</p>
------------------------------	--

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:

4. 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.snбомрshared1.gbucdsint02bom.oraclevcn.com:17777"

    var home_html = "?ojr=home";

    var url = server_url + "/" + home_html;

    plugins.appPreferences.fetch(MAINTENANCE_PAGE_STATUS_CODE_SUCCESS,
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();

}

}

</script>

```

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
PatchSetVersion	Version of the. OBDX application to identify the version of the workspace inside the patch installer.
SUITENAME	Group identifier for sharing keystore information. This Should match the app group added in the profile and in Targets->Signing Capabilities. App Groups are linked with the provisioning profile and its value can be verified from the Zigbank target->Signing Capabilities. 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 always set YES for token-based development. Configurable: No

8. For CHATBOT:

1. This is optional configuration. Bank needs to do this if Oracle Digital Assistant (ODA) is supported in their workspace.
2. Adding chatbot support to mobile application refer section ODA Chatbot Inclusion
3. Refer section ODA Chatbot Inclusion for more details.
4. Below details can be obtained from the Oracle Digital Assistant portal.

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

9. For location tracking metrics

1. 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.
----------------------	--

10. For displaying “Rate Us” to redirect to Appstore page.

1. This is optional. User can have an option (“Rate Us”) in settings to display App Store rating for the application. This option can be enabled/disabled from UI. Also, on click of the option, to open AppStore page for the application set below value.

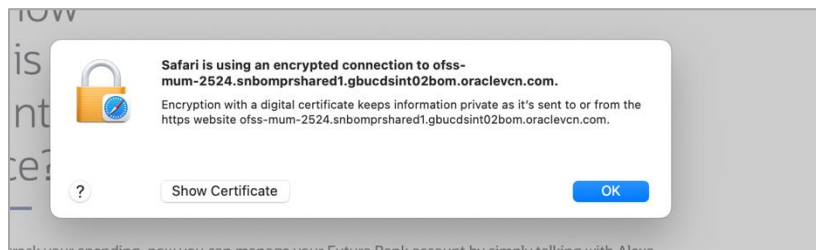
AppStoreURL	Replace @@AppStoreID with that of the application.
-------------	--

2.6 Enabling SSL pinning in the application

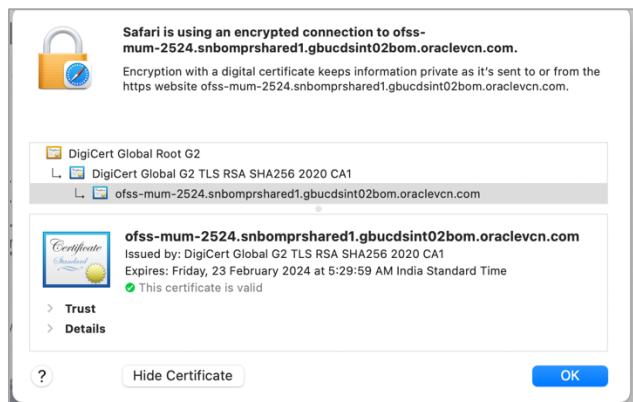
SSL pinning is required to securely connect with a https bank server URL to mitigate Man-in-middle-attack. It is recommended to enable this in production. By default, SSL pinning is set to NO in the application for development purpose so that the application can connect to https URLs without SSL Pinning checks.

Note: OS by default checks for a valid SSL trusted certificate using **App Transport Security (ATS)**. Hence, the server should have a valid certificate chain and adhere to ATS requirements. SSL pinning is additional security measure as per security standards.

1. To enable SSL pinning, bank needs to follow the configurations mentioned in the section:
Configurations for the IOS application:
2. The SSL certificate needs to be added in the workspace. To download and add this certificate, follow below steps:
 - 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:

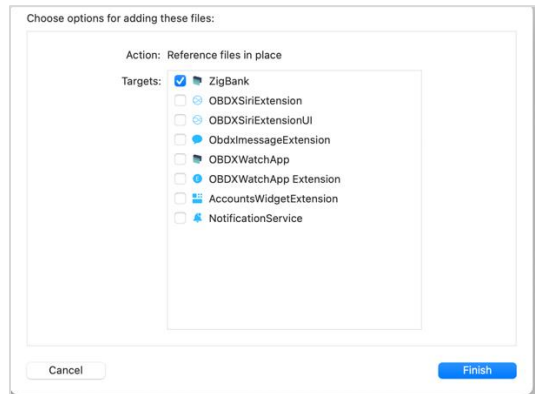


- Click on Show certificate and below window will be displayed



- Press and drag the certificate icon from Safari to any location on your machine.
- Rename it to any certificate.cer
- Copy-paste the certificate inside IOS workspace at this location
/service/workspace_installer/zigbank/platforms/ios/ZigBank/
- Right Click on Resources folder and select "Add Files to Zigbank"
- Select the certificate file which is saved in above step.

- Select ZigBank target



- 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`. Order doesn't matter.

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

▼ PinnedCertificateName	Array	(2 items)
▼ Item 0	Array	(2 items)
Item 0	String	certificate
Item 1	String	@@PINNING_CERTIFICATE_NEW_1
▼ Item 1	Array	(2 items)
Item 0	String	@@PINNING_CERTIFICATE_OLD_2
Item 1	String	@@PINNING_CERTIFICATE_NEW_2

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 before the old certificate expires. This will allow 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 must follow same steps as mentioned above
- 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 must set all those URLs in each item as below:

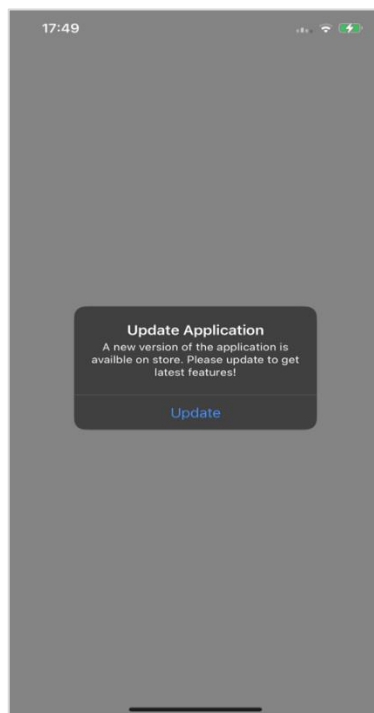
▼ PinnedUrl	Array	(2 items)
Item 0	String	https://abc.bank.com
Item 1	String	@@PINNING_URL_2

2.7 Enabling Force update

- This is an 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: **Configurations for the IOS application.**
- Set the version for the new update 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).

Note: Bundle version string format should be same as that of the AppStore version set on iTunes developer portal. If Bank is uploading AppStore version 1.0 then the bundle version should be 1.0 and so on.

- So, if live app version is 1.0.1, the new application update can have version greater than 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 new updated version.
- If updated version is available, 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

- This feature will work only after 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.

2.8 **Device Registration and Push Registration Functionality**

1. 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.
2. While user registers his second device or same device again (by re-installing the application), a popup will appear to notify the same.
3. If user confirms, then the current device will be registered, and all previous registrations will be removed.



If user cancel, the process is exited.

4. Also, in this version, only one device is allowed to be registered for push.
5. Bank can allow multiple devices to be registered for same username in their setup by setting below two configurations in config table.
6. 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.
7. 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.

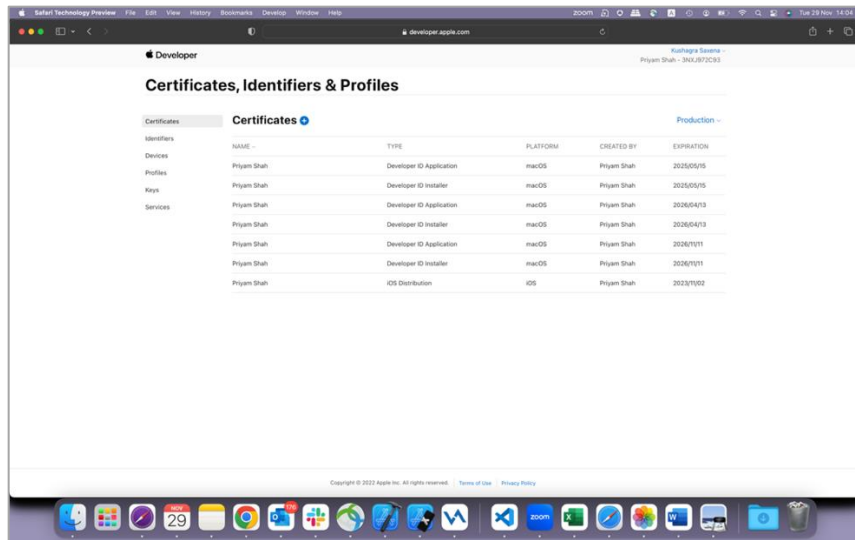
2.9 Generating Certificates for Development, Production

1. This is required for running the application on device for debugging, testing as well for releasing the application to Appstore.
2. Bank can refer to Apple's documentation on how to create certificates and provisioning profiles.
3. Create all certificates (by uploading CSR from keychain utility), provisioning profiles and push certificates by login in developer console.

Below are steps:

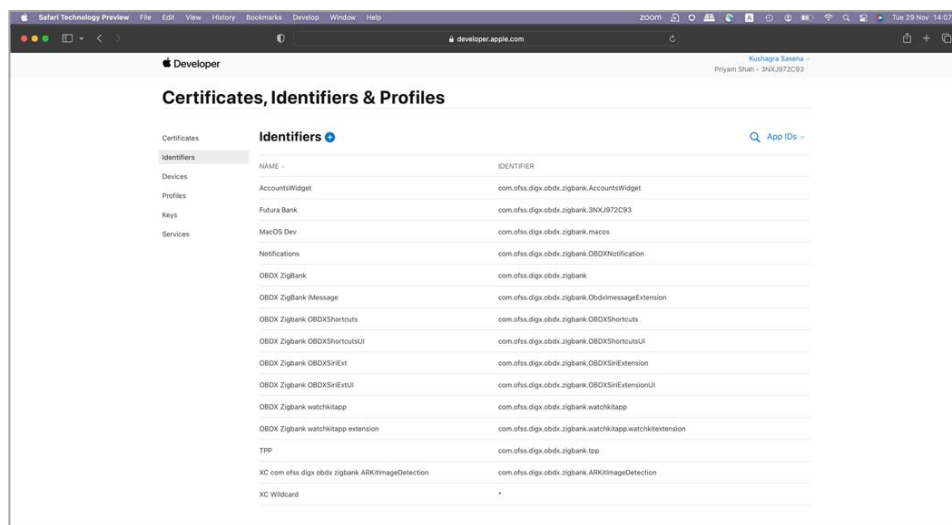
- a. Certificate Creation
- b. AppID creation
- c. Profile creation
- d. Adding device UUIDS to profiles
- e. Generating Push certificate for server

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 bank has configured in workspace. Available targets are

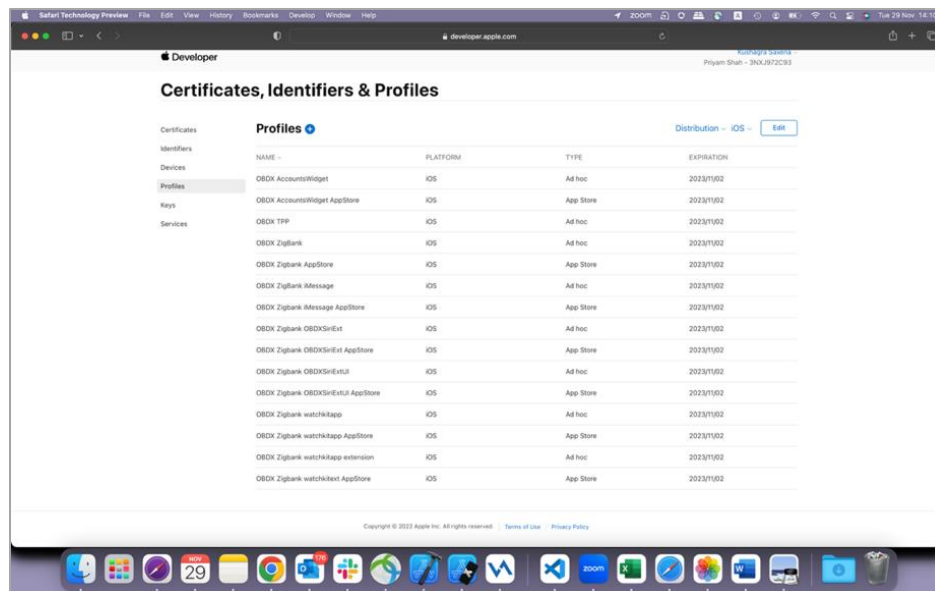
- ZigBank
- OBDSiriExtension
- OBDSiriExtensionUI
- ObdxImessageExtension
- OBDXWatchApp
- OBDXWatchApp Extension
- AccountsWidgetExtension
- Notification Service



1. Add capabilities as shown below and ensure the bundle identifier matches with the capabilities added in Xcode.
2. Ensure AppGroups capability is added to all profiles and for appIDs.
3. Ensure SiriKit, App Groups, Push Notifications, Associated domain capabilities are added in Zigbank appIDs.
4. Bank can refer base workspace for the naming convention followed for bundle identifier for each target. Below are the appIDs which we need for OBDX application in similar format as below:

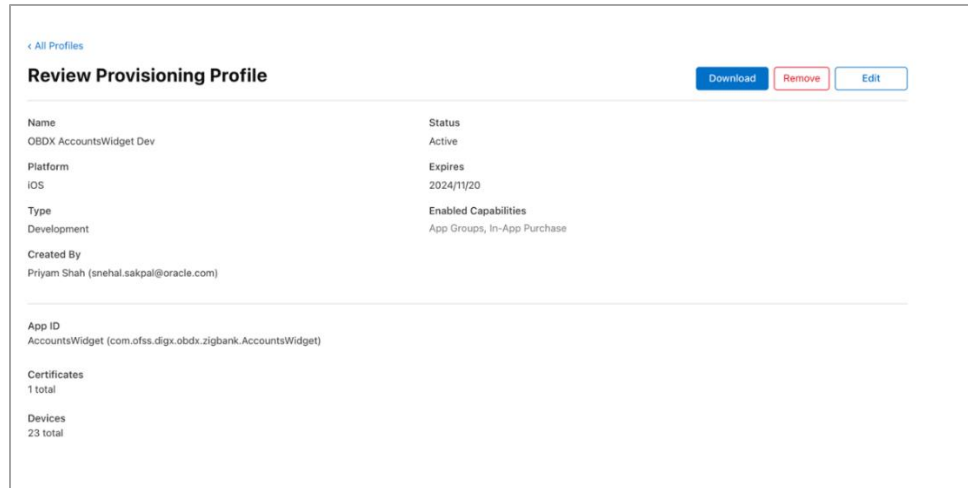
OBDX ZigBank iMessage	com.ofss.digx.obdx.zigbank.ObdxImessageExtension
OBDX Zigbank OBDXSiriExt	com.ofss.digx.obdx.zigbank.OBDXSiriExtension
OBDX Zigbank OBDXSiriExtUI	com.ofss.digx.obdx.zigbank.OBDXSiriExtensionUI
OBDX Zigbank watchkitapp	com.ofss.digx.obdx.zigbank.watchkitapp
OBDX Zigbank watchkitapp 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.

Example: AccountWidget development profile will have development certificate and appld created for AccountWidget. Likewise for other targets.



Adding Bundle Identifiers:

Bundle identifiers need to be added in the Info.plist of each frameworks. Example: if bank's appld for Zigbank is com.ofss.digx.obdx.zigbank then follow below steps

1. Right click on OBDXFramework.framework(in Xcode's Project Navigator) -> Show in Finder
2. When the finder directory opens the right click OBDXFramework.xcframework -> select ios-arm64 -> Select OBDXFramework.framework
3. Open Info.plist and set Bundle identifier as com.ofss.digx.obdx.zigbank.OBDXFramework
4. Repeat the steps for the other three frameworks as well with the following values:

Bundle identifier for Cordova.framework : com.ofss.digx.obdx.zigbank.Cordova

Bundle identifier for OBDXExtensions.framework :
com.ofss.digx.obdx.zigbank.OBDXExtensions

Bundle identifier for OBDXWatchFramework.framework :
com.ofss.digx.obdx.zigbank.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

OBDSiriExtensionSiri – com.ofss.digx.obdx.zigbank.OBDSiriExtension

OBDSiriExtensionUI – com.ofss.digx.obdx.zigbank.OBDSiriExtensionUI

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

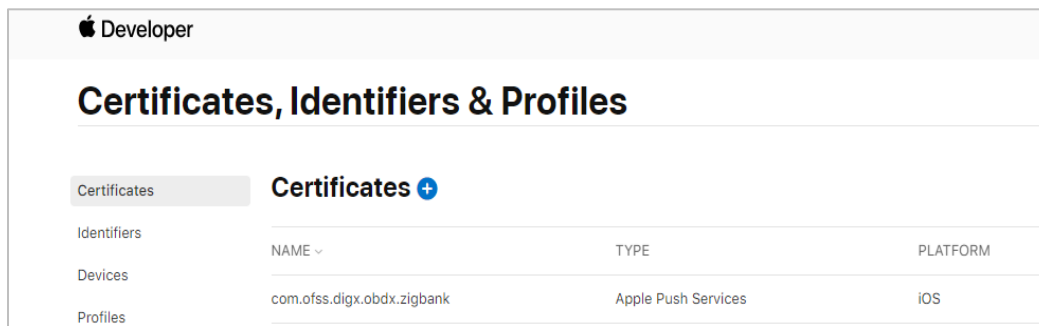
NotificationExtension- om.ofss.digx.obdx.zigbank.OBDXNotificationExtension

Adding device UUIDS to profiles

1. For development profiles, testing device UUIDs need to be added, and same devices need to be selected in the development profile.

Generating Push certificate for server

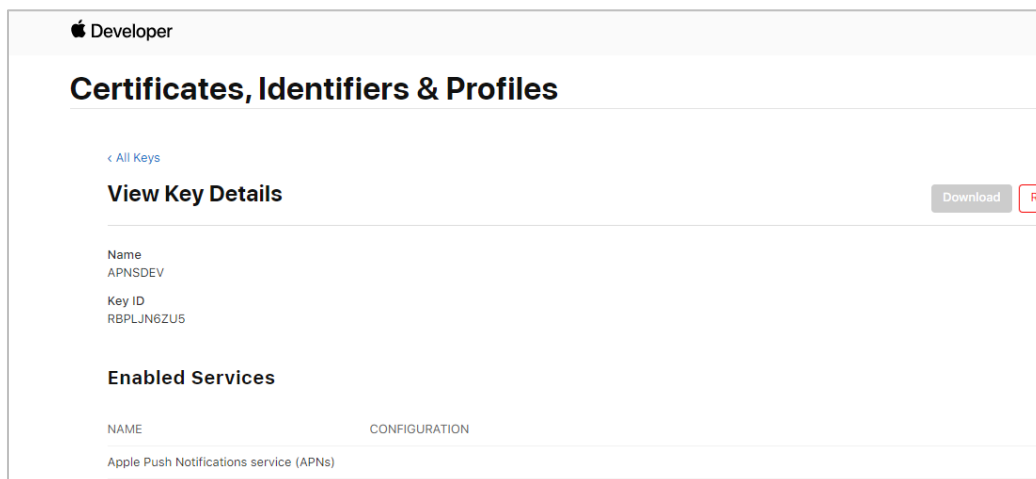
To set up an APNs certificate on your server, bank will need to generate a Certificate Signing Request (CSR), upload it to Apple's Push Certificates Portal, download the resulting certificate, and then install it on your server, along with the necessary root certificates for secure communication



2.10 Setup for Push Notification in the application

1. Push notification services are to be created using .p8
2. For p8, bank needs to setup Key and update database with the details. All details are mentioned below:
3. Create APNS key from developer portal. Navigate to the "Keys" section and create APNS key.

Note: APNS key and download the .p8 file. We need the contents of this file to update in database.



4. 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	Specifies whether to pick certificate password
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 add a row with value 'dev' Possible values for Dev: 1. dev For Production: 2. prod	For prod push this row is not required as it takes prod by default
5	DIGX_FW_CONFIG_VAR_B	APNS_BUNDLE		Eg. com.ofss.digx.obdx.zipbank	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)

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		The content of .p8 is as below Copy the entire string convert it to base64 string online and then update the base 64 value in the column	Open the .p8 file and copy contents and convert it to base 64 and update the base 64 value to column (Update for all entities)

2.11 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".

2.12 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

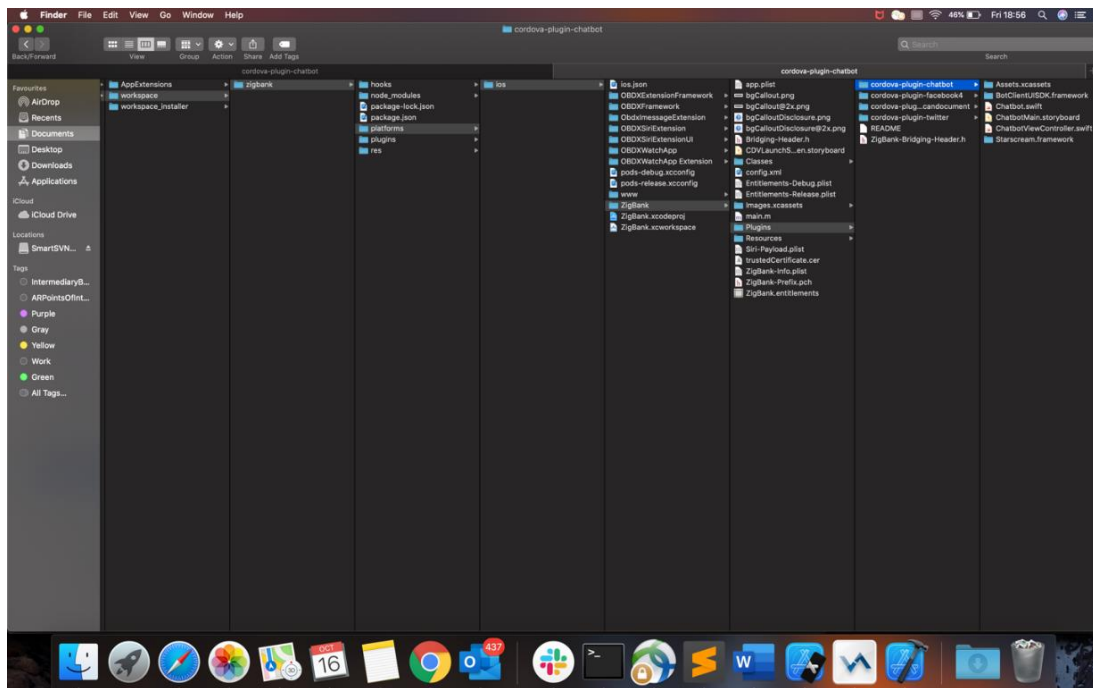
5. There is an OTP expiration time set in “digx_fw_config_ALL_b” table.
6. Also, there is business policy check set to 10 minutes for validation of the generated 2fa token. Bank can write their own business policy where they can modify the 10 minutes time.

So, the time in UI code should not exceed 10 minutes and OTP expiration time in “digx_fw_config_ALL_b” table.

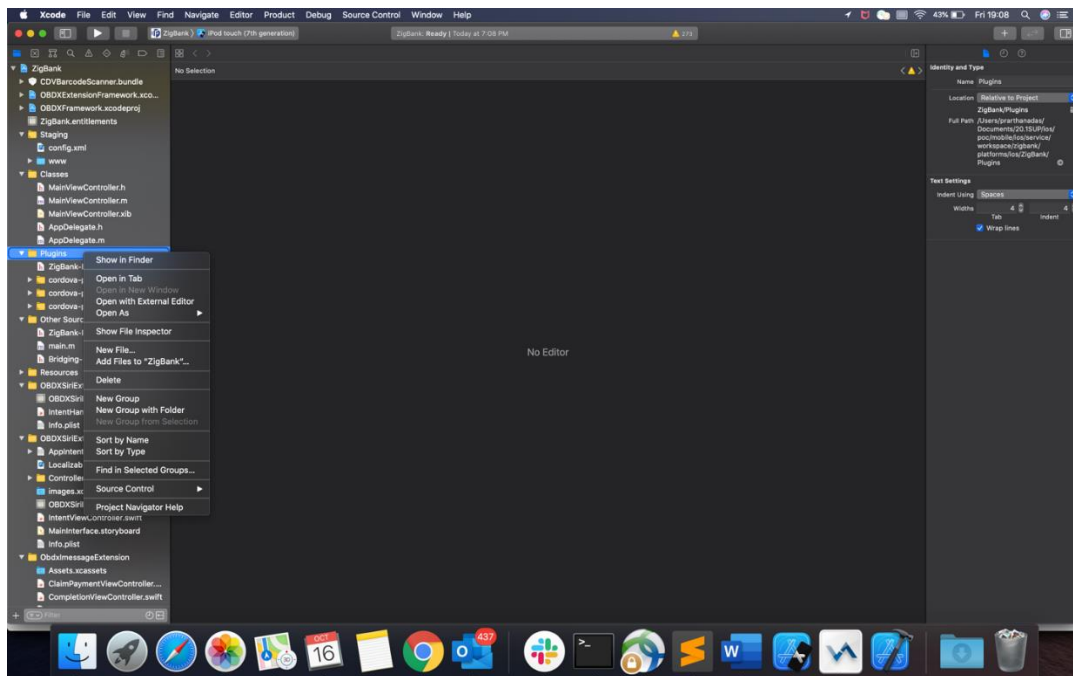
2.13 ODA Chatbot Inclusion

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

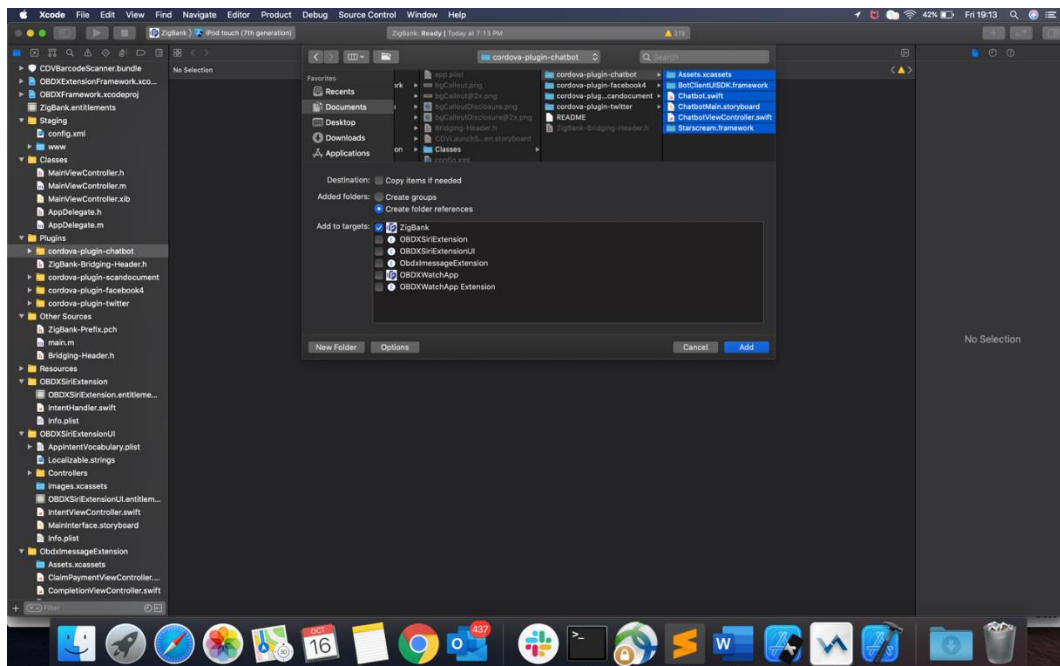
1. Copy the folder "cordova-plugin-chatbot" from the SVN path
: workspace_installer/AppExtensions/ODASDK The frameworks can be found at ODA Client SDK for iOS x.y.z - Latest in <https://www.oracle.com/downloads/cloud/amce-downloads.html#license-lightbox>. After downloading and unzipping the latest version the frameworks for device and simulator as single file – "BotClientUISDK.xcframework" and "Starscream.xcframework". 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.



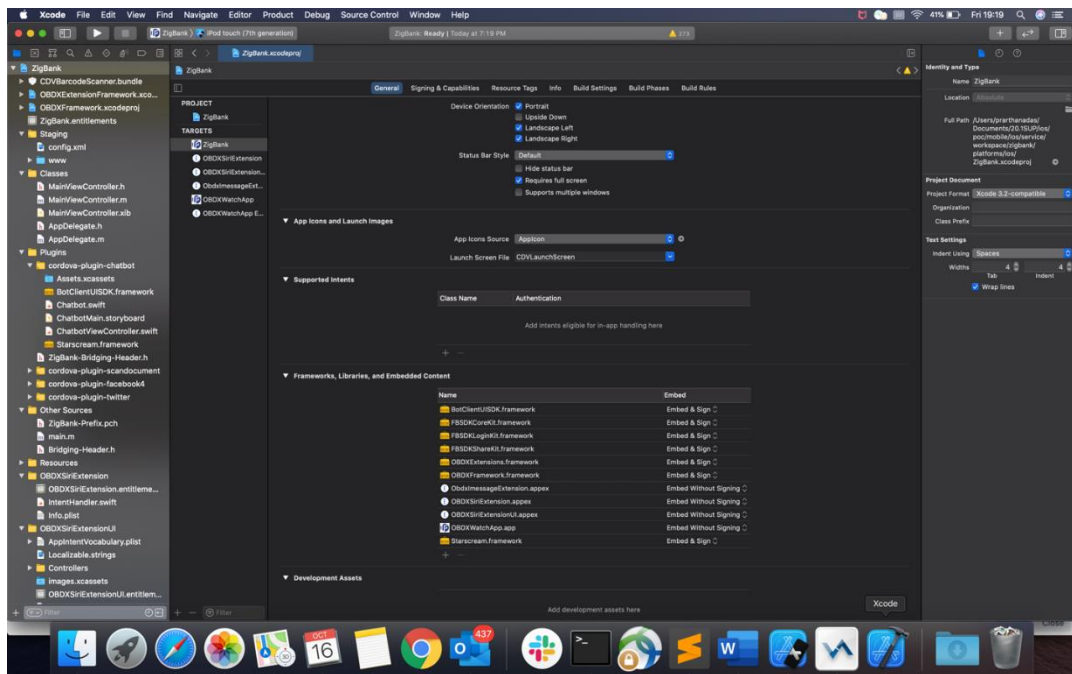
3. Open the Zigbank.xcodeproj file, right click on "Plugins" folder and select "New Group" option. Name the group as "cordova-plugin-chatbot".



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.xcframework" and "BotClientUISDK.xcframework" to "Embed and Sign". Failing to do so will make the app crash after installation.



2.14 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.

Pre-requisite:

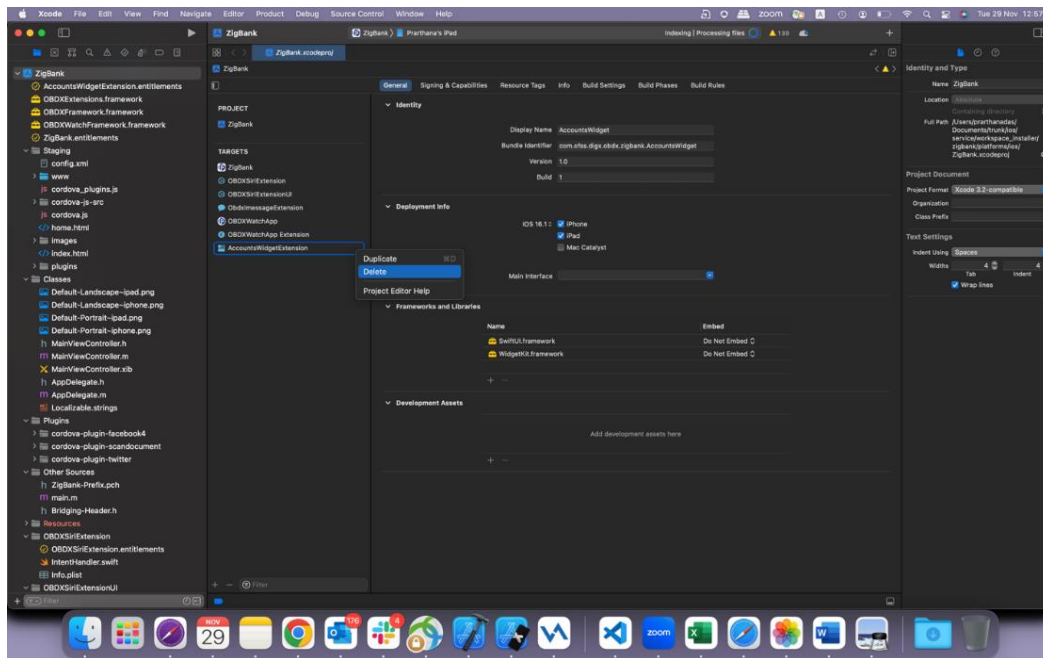
Quick Snapshot feature needs to be enabled in the application from the login screen. (Refer function doc - User Manual Oracle Banking Digital Experience Quick Snapshot.docx)

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

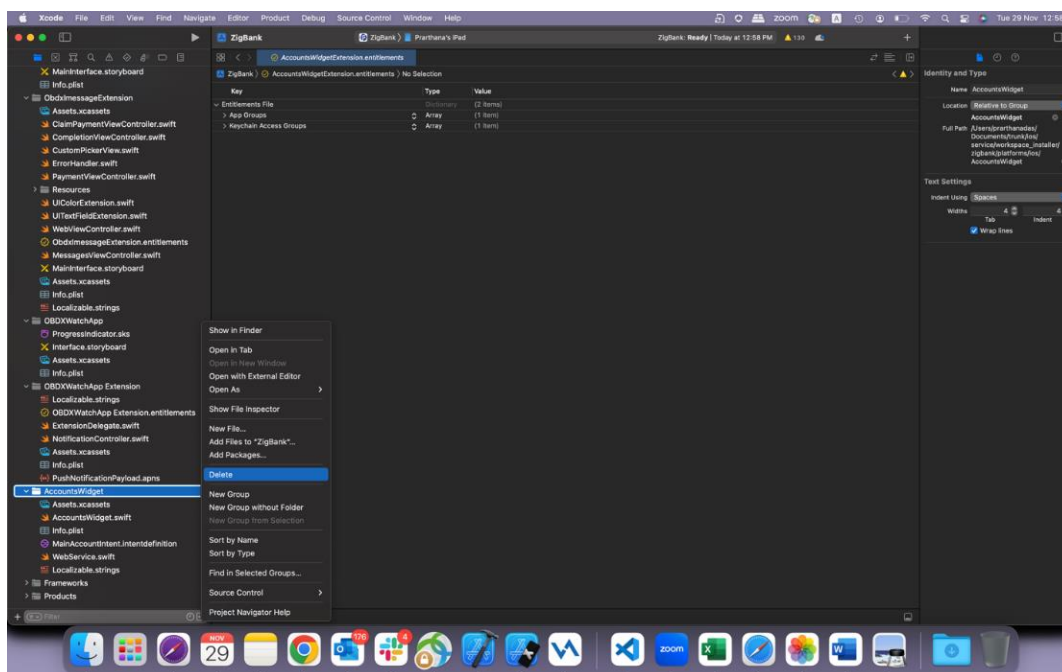
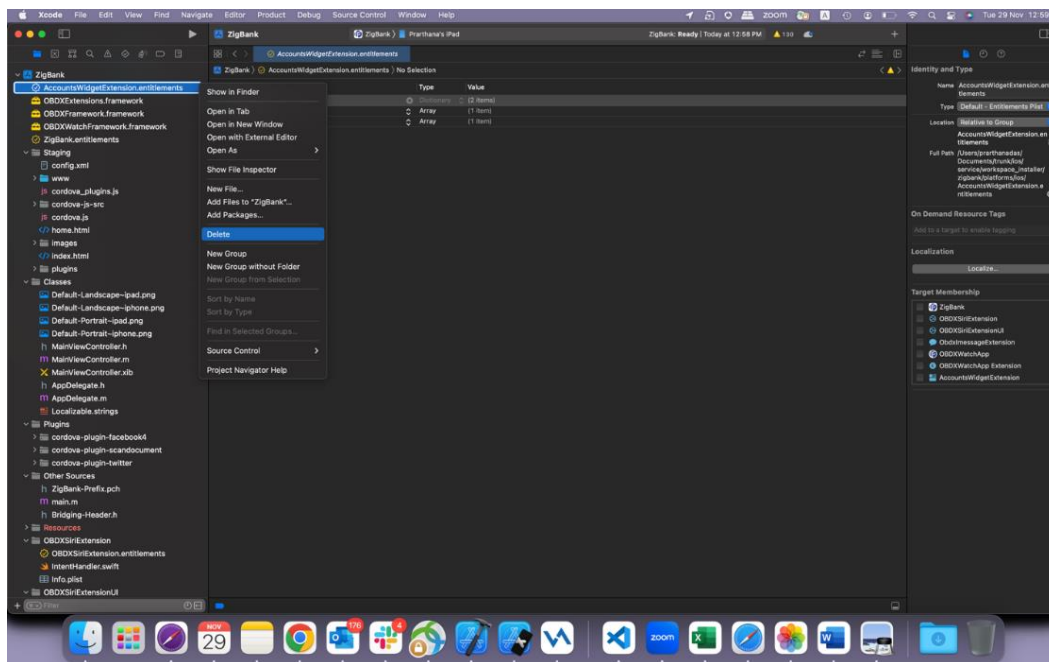
Removal of Widget functionality from workspace:

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

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



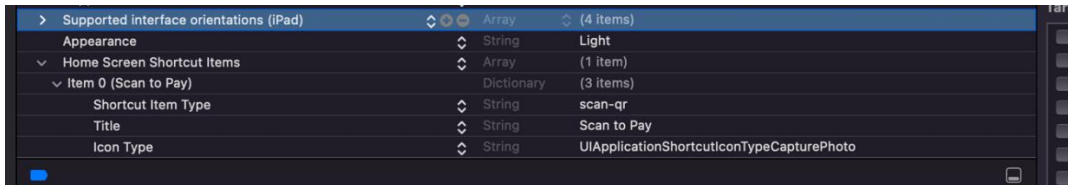
2. Delete “AccountsWidgetExtension.entitlements” file and “AccountsWidget” folder from Project navigator.



2.15 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."



2.16 Passkey (Passwordless login)

Passkey is passwordless login 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.

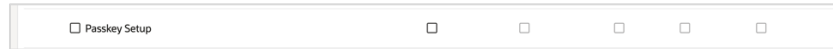
Note: Only one of two login types can exist in the application, Either passkey or alternate (PIN/Pattern/Facelid/touchID). So, setup this option for brand new application update in the market. If the application is already launched and users have enabled alternate login, then enabling passkey option in later application versions will hide the alternate login options from the application and user will have an option to register passkey in his profile settings.

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 within the application. User will not be able to register for passkey and 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
3. thirdPartyAPIs -> passkey -> required -> false
Open Zigbank project in Xcode, Select Zigbank target -> Signing Capabilities -> Delete Associated domain

TO ENABLE THIS OPTION:

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

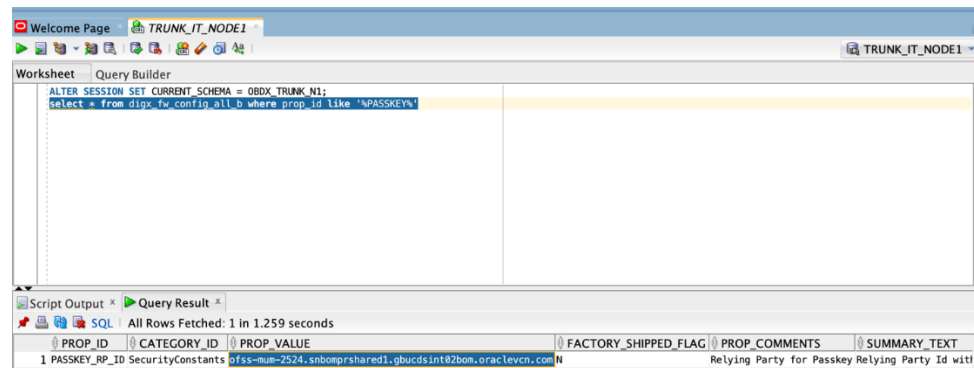


2. Set this flag in channel-framework-js-configurations-config.js to true
thirdPartyAPIs -> passkey -> required -> true
3. Along with above, we need below server side and application side setup

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'



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

2. Hosting AASA file (Apple-app-site-association) on server.
 - a. AASA- Apple App Site Association file which IOS installs on the device when application is installed. This AASA 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.

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

Server-Side AASA Setup:

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

Below are the sample values for a single application supporting deep link. Bank must update banks' teamID and bundle ID.

numberofapps=1

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

paths0=*

3. 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"

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

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

The output should be as below

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

Application side setup:

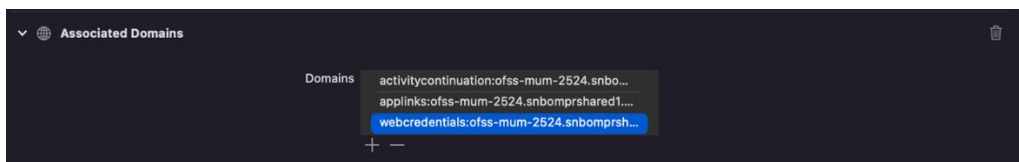
1. Open developer portal and enable Associated domain for your appID



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 AASA file is hosted. Port and "https" should not be added here.



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 Apple ID 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.

2.17 Deep linking - To open reset password, claim money links with the application

1. Deep linking in IOS works with https URL and a valid ASA configuration. Deep linking keeps the application flow within the application when user clicks on bank's reset-password or claim-money link on email or message.
2. AASA- Apple App Site Association file which OS installs on the device when application is installed. This AASA 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 AASA 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. AASA file needs to be hosted on https server with valid SSL certificate. There should be no redirection to this file.
2. 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 deep link. Bank should update banks' teamID and bundle ID.

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>

Team ID is present in developer account in membership details

numberofapps=1

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

paths0=*

3. 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"

4. After the setup is done, this AASA 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

The content output should be as below

```
{
  "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"
    ]
  }
}
```

```
}  
  
}
```

Application side setup:

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

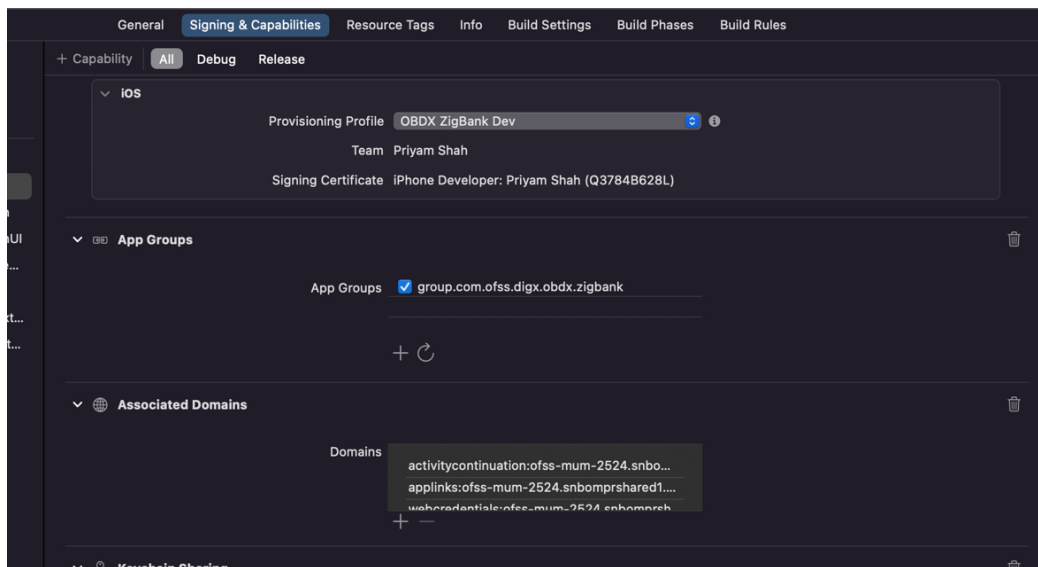


2. Open Zigbank.workspace->Select Zigbank target. Go to Signing and Capabilities – In associated domain section, update the URL with bank's host for activitycontinuation and applinks

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.

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.



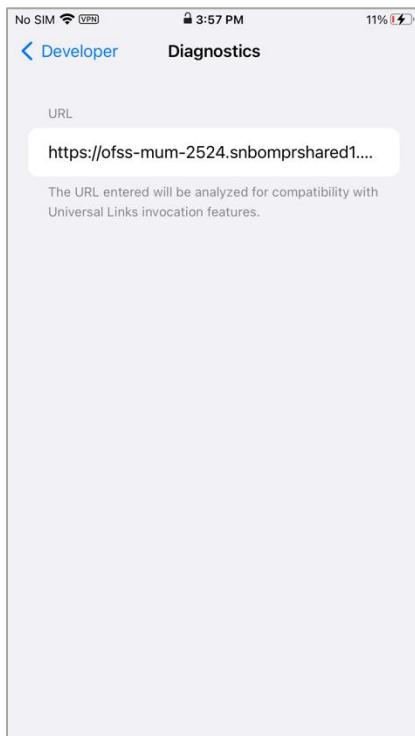
3. Update the key_server_url to https URL in the Zigbank project app.plist

Device Side setup for development and testing:

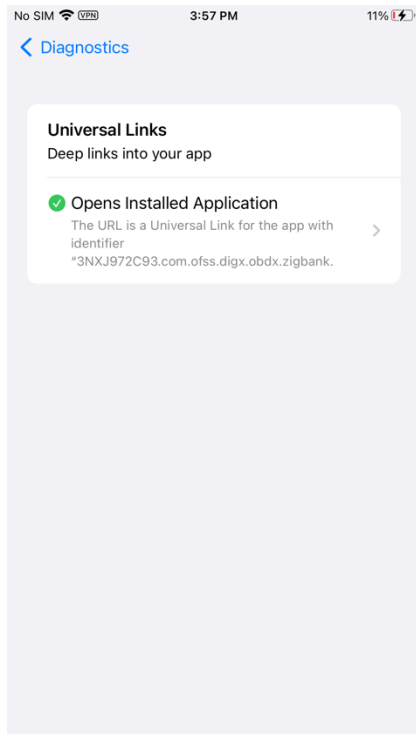
1. To test on device, Developer mode should be enabled. Additionally, go to Phone Settings – Developer mode- > Enable “Associated domain Development”.
2. With all above setup, install the application on the device. Please not while installing the device must be connected to network in which the AASA 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 deep link. If all setup is correct and AASA 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 AASA file is hosted.

<https://ofss-mum-2524.snbomprshared1.gbucdsint02bom.oraclevcn.com/>



If we see below message, then deep link 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 AASA is hosted and should not contain port.

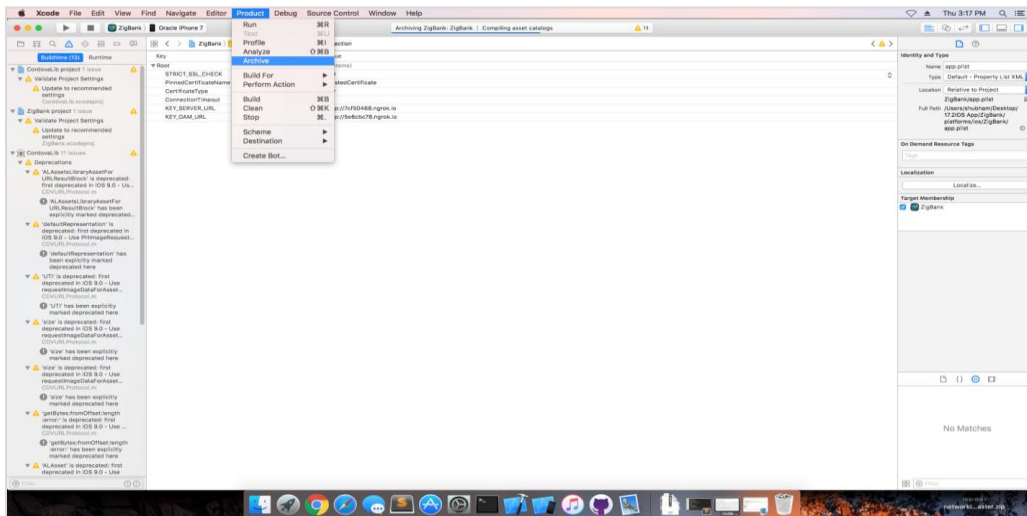
Long press on the link and you must see "Open in Zigbank App" option. Clicking the option page opens in the application.

2.18 Changing App Icons and Assets

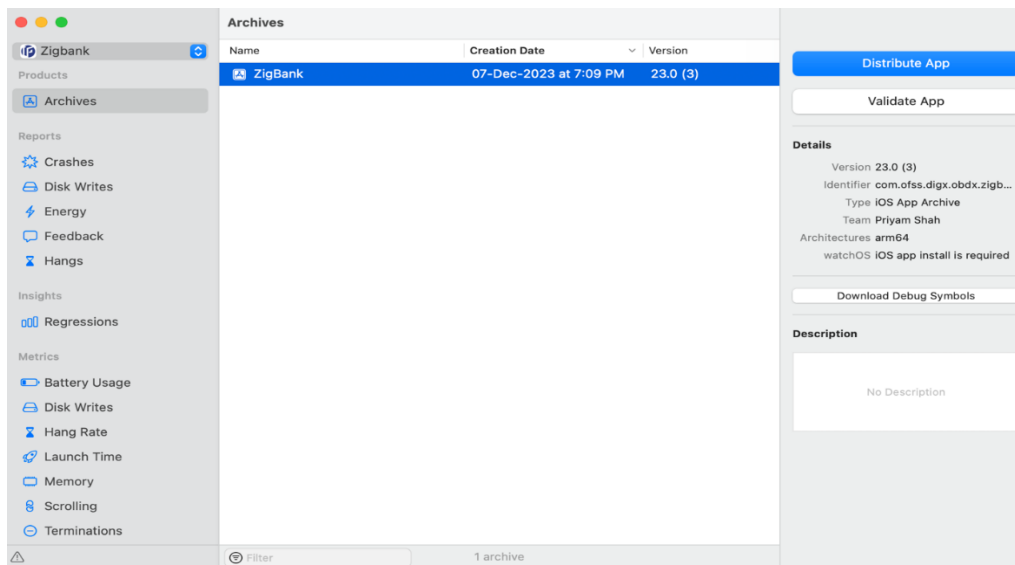
1. All the app icons for all sizes and AppStore icon needs to be replaced in Images.xcassets file under Zigbank->Resources folder
2. The launch images to be added in Images.xcassets file under Zigbank -> Resources folder
3. The application as well shows splash images when application goes in background. These images are present in Zigbank -> Classes folder with names prefixed "Default-". Replace all 4 images with appropriate size.

2.19 Archive and Export

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

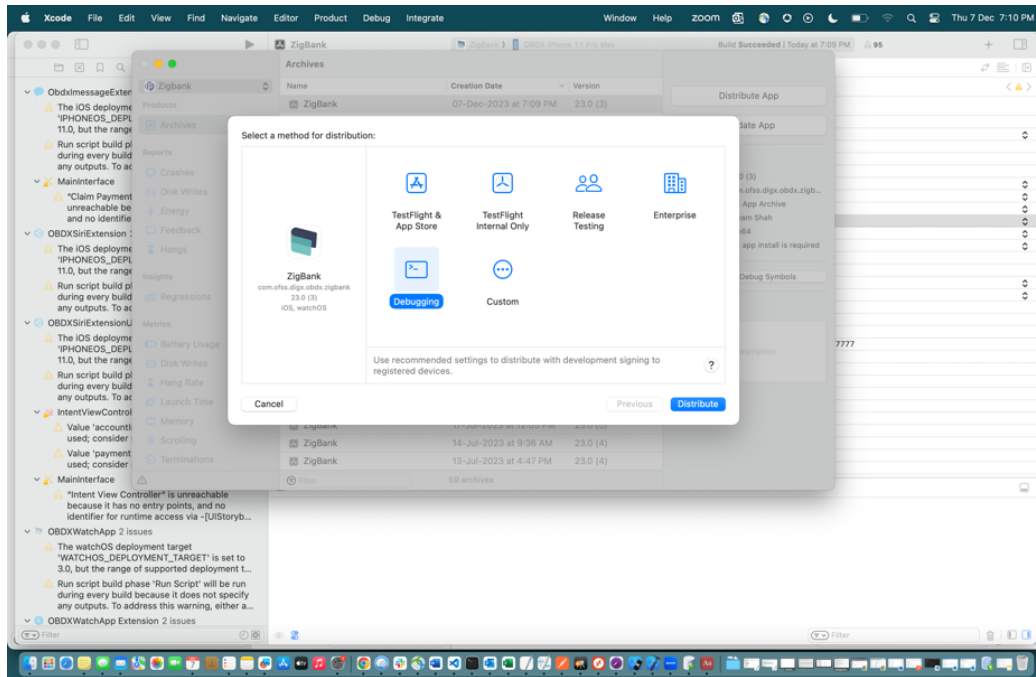


2. After archiving has successfully completed. Following popup will appear.

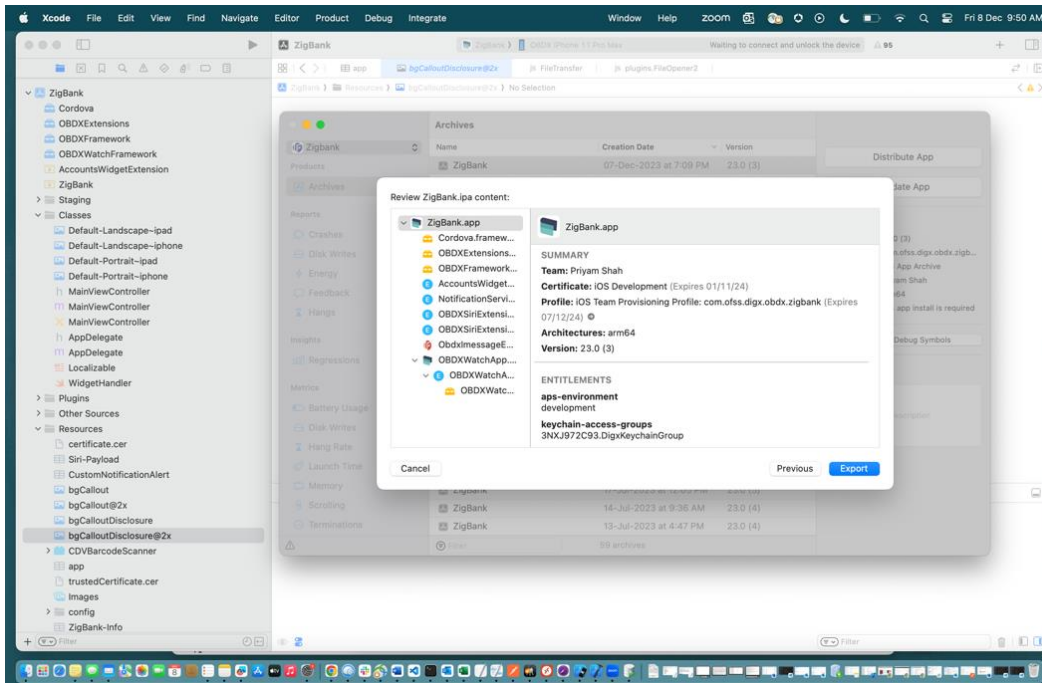


3. Click on Distribute App in the right pane of the popup -> select the **Method of Distribution -> Select Distribute**. Review the contents and click on **Export -> Export** and generate the .ipa

- a. There are multiple options for exporting, select according to what is needed.
- b. Debugging – this will create an ipa with development profile for internal testing.
- c. Release Testing – This will create an ipa with Ad Hoc distribution profile for adHoc testing.
- d. TestFlight Internal Only, TestFlight & AppStore- As the name suggests, this is for TestFlight and AppStore release.

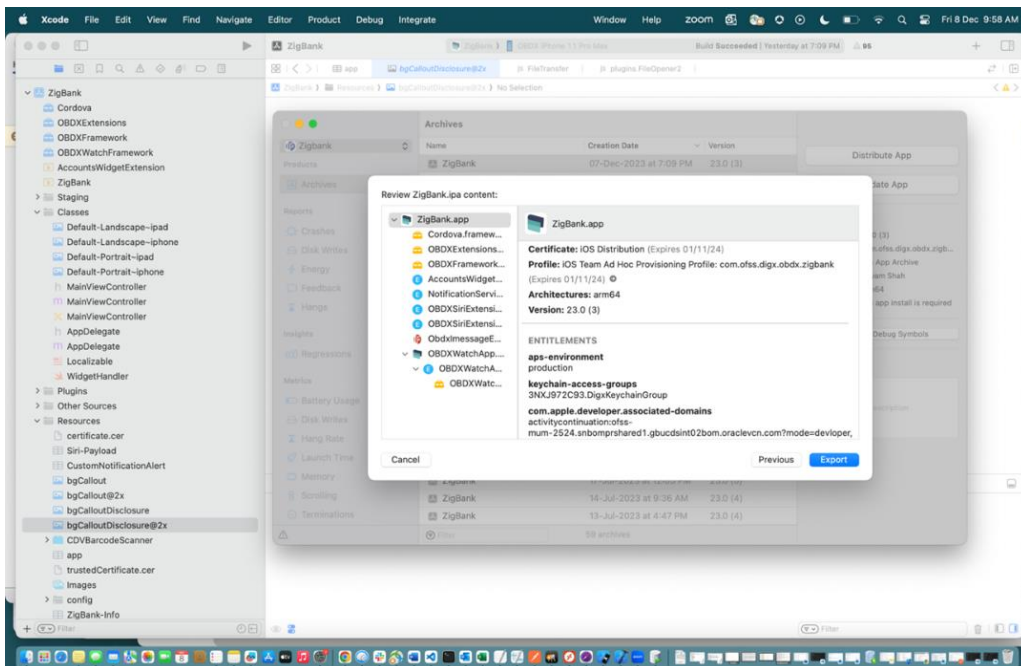


This is window which appear after selecting Debugging option. Note the Certificate and provisioning profile is for development.



4. 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.

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



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

3. 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 for 2fa
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:

1. Download and Install node as it is required to run npm and cordova commands.
2. Latest Xcode to be download from Mac App Store. This document is w.r.t to Xcode 16.2
3. Authenticator Application is supported only on current iOS version and only one version preceding that.

3.1 Authenticator UI (Follow any one step below)

3.1.1 Using built UI

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

1. Copy the contents from the dist inside Authenticator workspace -> platform/ios/www folder

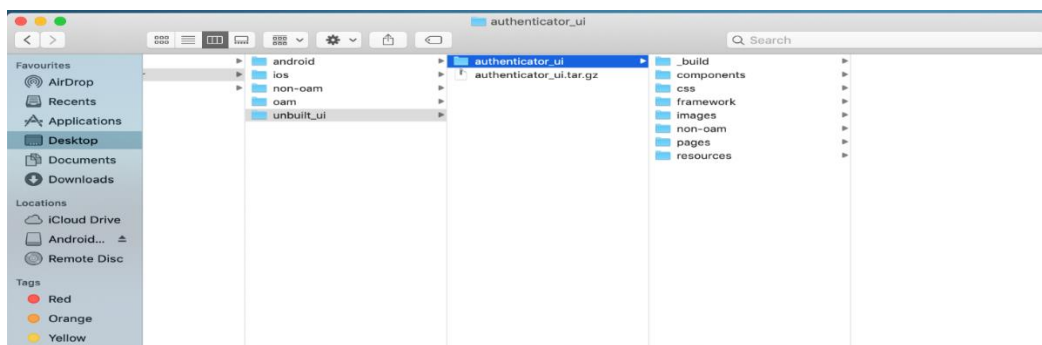
3.1.2 Using unbuilt UI

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

3.1.3 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:

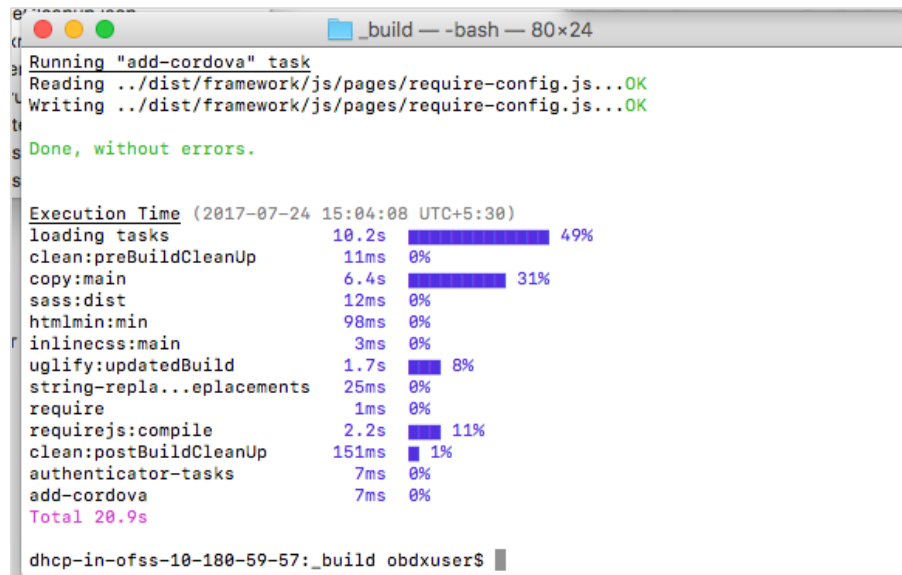
```
sudo npm install -g grunt-cli
```

```
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”.

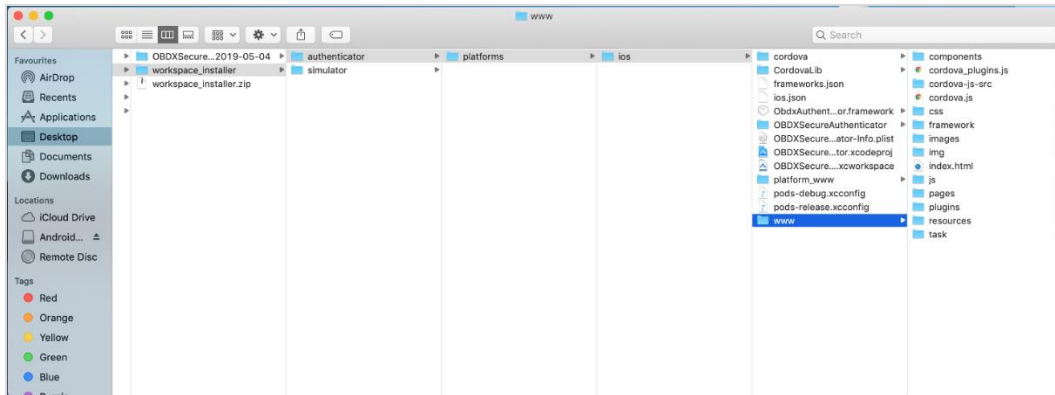


```
Running "add-cordova" task
Reading ../dist/framework/js/pages/require-config.js...OK
Writing ../dist/framework/js/pages/require-config.js...OK
Done, without errors.

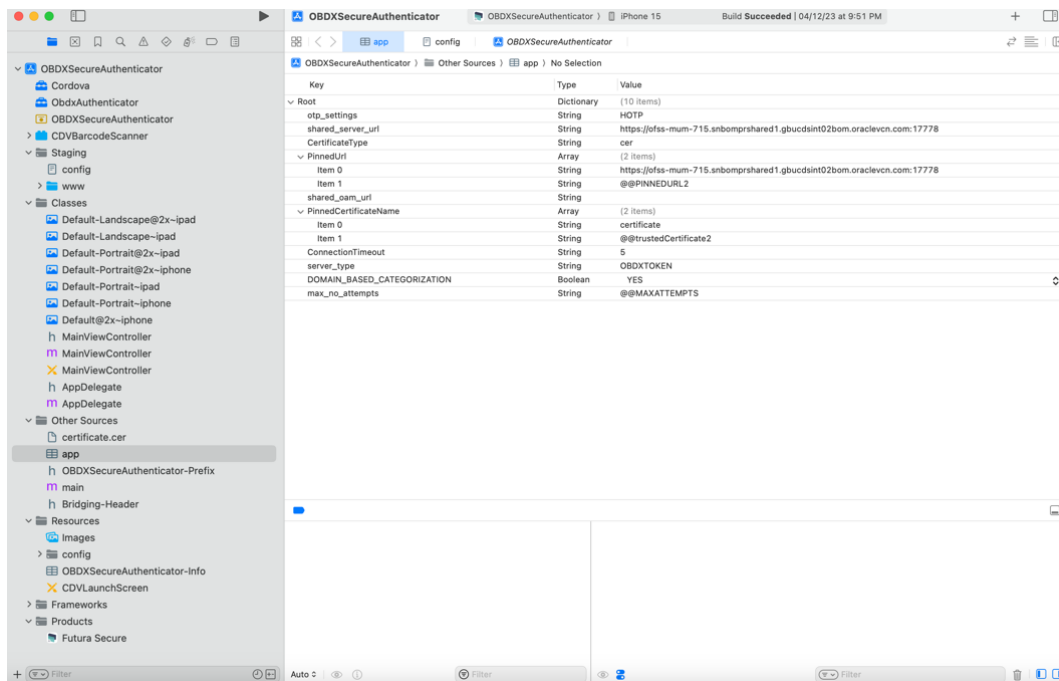
Execution Time (2017-07-24 15:04:08 UTC+5:30)
loading tasks          10.2s ██████████ 49%
clean:preBuildCleanUp   11ms ██████████ 0%
copy:main              6.4s ██████████ 31%
sass:dist              12ms ██████████ 0%
htmlmin:min            98ms ██████████ 0%
inlinecss:main         3ms ██████████ 0%
uglify:updatedBuild    1.7s ██████████ 8%
string-repla...eplacements 25ms ██████████ 0%
require               1ms ██████████ 0%
requirejs:compile      2.2s ██████████ 11%
clean:postBuildCleanUp 151ms ██████████ 1%
authenticator-tasks    7ms ██████████ 0%
add-cordova            7ms ██████████ 0%
Total 20.9s
```

3.2 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.xcodeproj to open the project in Xcode



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

5. **shared_server_url** = <bank's https url>

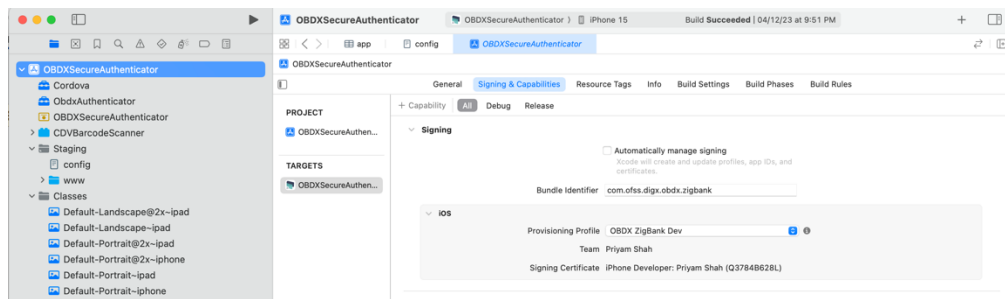
6. Set value of max_no_attempts to value greater than 0.

Server_type - OBDXTOKEN

7. **DOMAIN_BASED_CATEGORIZATION** – YES

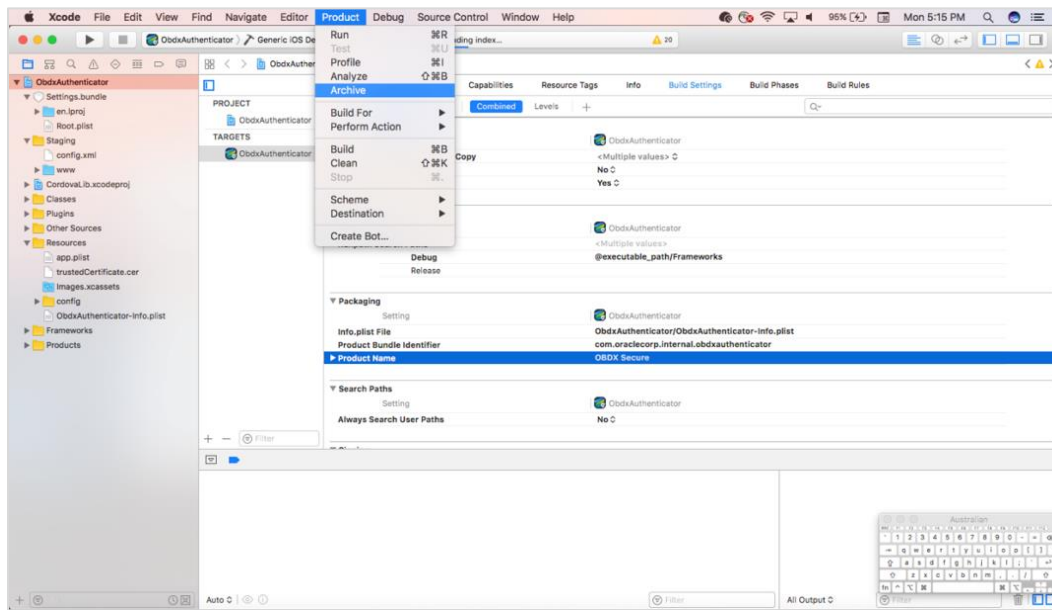
8. Create certificates and profiles on Apple Developer account. Use the bundle identifier in project settings and select appropriate profile in the application

1. Adding bundle identifiers:
9. Bundle identifiers needs to be added in the Info.plist of each the frameworks
10. 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
11. Also, set the identifiers and select appropriate profile in the target -> Signing & Capabilities tab as show below:
12. The application contains frameworks for devices and simulator both. Run the application directly on simulator without copying any frameworks.
13. The application can be archived using steps in Section [Archiving Authenticator Application](#) for running on device.

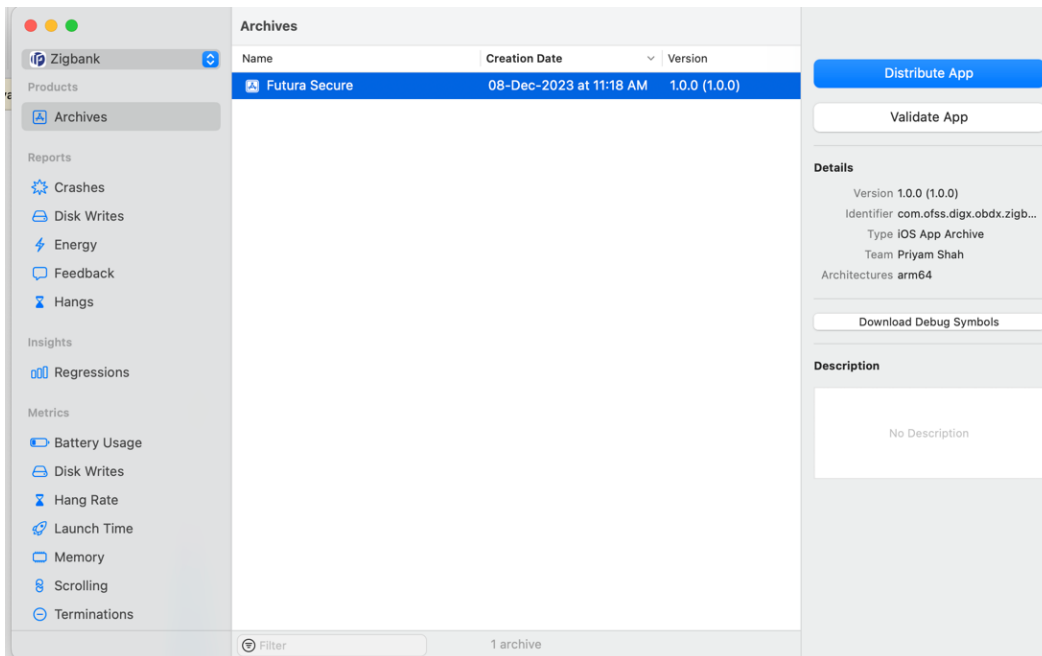


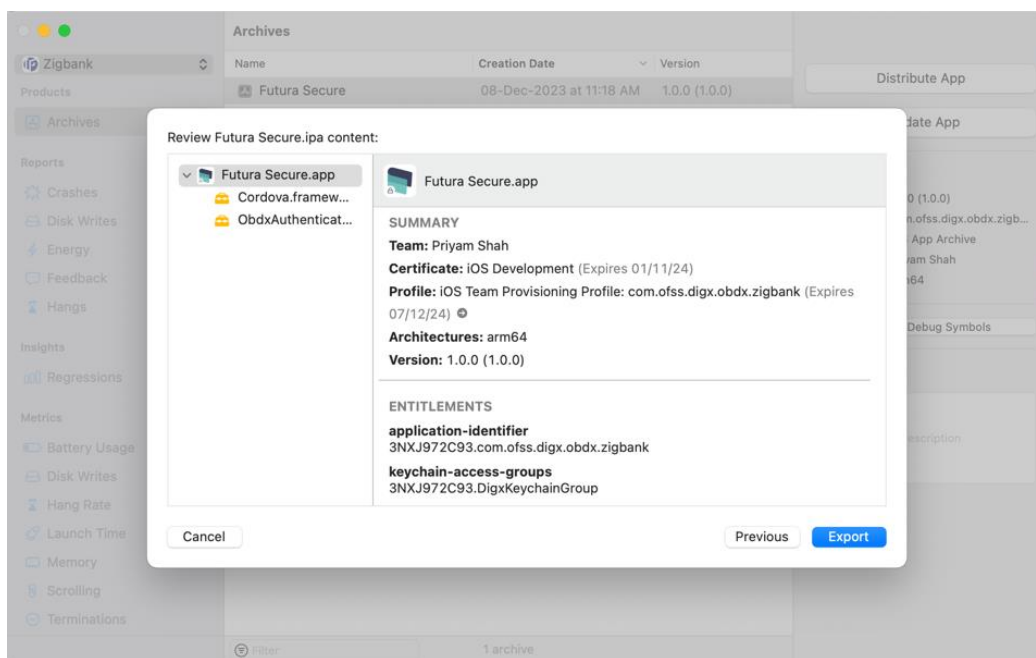
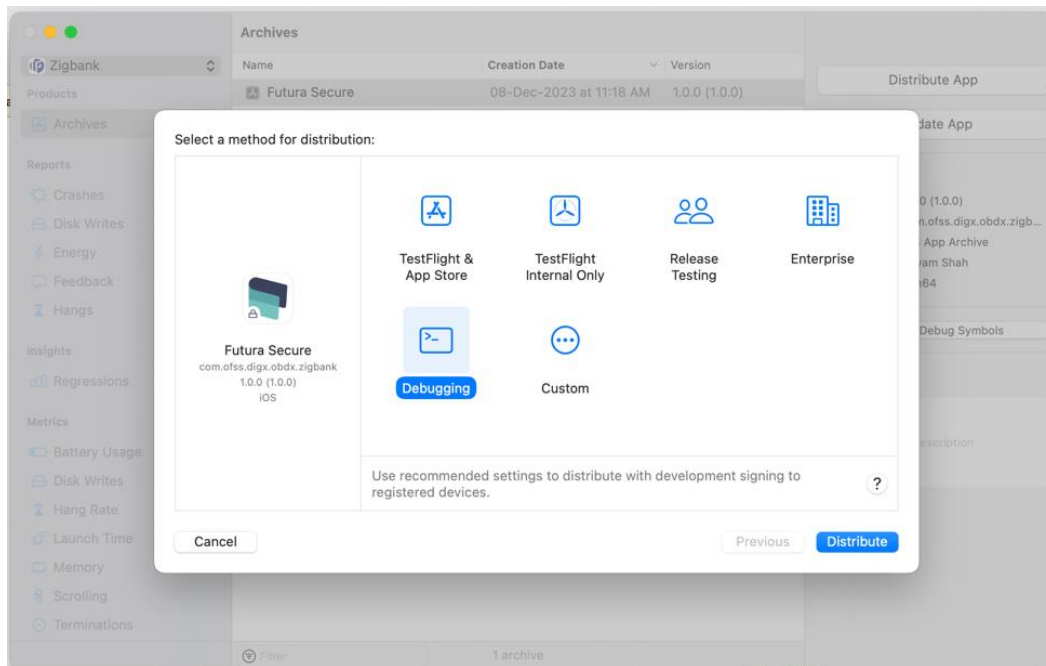
3.3 Archiving Authenticator Application

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”





3.4 Using SSL in Authenticator App

Follow below steps to setup SSL in Authenticator application:

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

1. **PinnedUrl** – Item 0 – replace @@PINNEDURL1 with your https url Enabling SSL pinning in the application (without port)
2. To add SSL certificate into workspace follow steps in section
3. **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”

4. Apple Privacy

Apple requirements for Required Reason's Api and Data Types usage

Note: This document is for bank's reference for the Apple's rejection issue related to "Required Reason's Api and Data Types usage"

References:

WWDC 2023 video.

<https://developer.apple.com/support/third-party-SDK-requirements/>

https://developer.apple.com/documentation/bundleresources/privacy_manifest_files/adding_a_privacy_manifest_to_your_app_or_third-party_sdk#4336740

What's needs to be done in the application

1. Since apple has pointed out Cordova as thirds party SDK which needs to add this, we have added "PrivacyInfo.xcprivacy" inside cordova framework and added below items for required reason Api and data types. Bank should delete its existing Cordova.framework and and OBDXframework.framework and add the new one in the patch fix.

Note: We have added as per what we have used inside the application and Apple's documentation, however Apple's Email with details will be more useful to target the required keys to be added.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>NSPrivacyAccessedAPITypes</key>
    <array>
        <dict>
            <key>NSPrivacyAccessedAPIType</key>
            <string>NSPrivacyAccessedAPICategoryDiskSpace</string>
            <key>NSPrivacyAccessedAPITypeReasons</key>
            <array>
                <string>E174.1</string>
            </array>
        </dict>
        <dict>
            <key>NSPrivacyAccessedAPITypeReasons</key>
            <array>
                <string>C617.1</string>
            </array>
            <key>NSPrivacyAccessedAPIType</key>
            <string>NSPrivacyAccessedAPICategoryFileTimestamp</string>
        </dict>
    </array>
</dict>
```

```

        <dict>
            <key>NSPrivacyAccessedAPITypeReasons</key>
            <array>
                <string>1C8F.1</string>
            </array>
            <key>NSPrivacyAccessedAPIType</key>
            <string>NSPrivacyAccessedAPICategoryUserDefaults</string>
        </dict>
    </array>
    <key>NSPrivacyTracking</key>
    <false/>
    <key>NSPrivacyCollectedDataTypes</key>
    <array>
        <dict>
            <key>NSPrivacyCollectedDataType</key>
            <string>NSPrivacyCollectedDataTypeDeviceID</string>
            <key>NSPrivacyCollectedDataTypeLinked</key>
            <true/>
            <key>NSPrivacyCollectedDataTypeTracking</key>
            <false/>
            <key>NSPrivacyCollectedDataTypePurposes</key>
            <array>
                <string>App functionality</string>
            </array>
        </dict>
    </array>
</dict>
</plist>

```

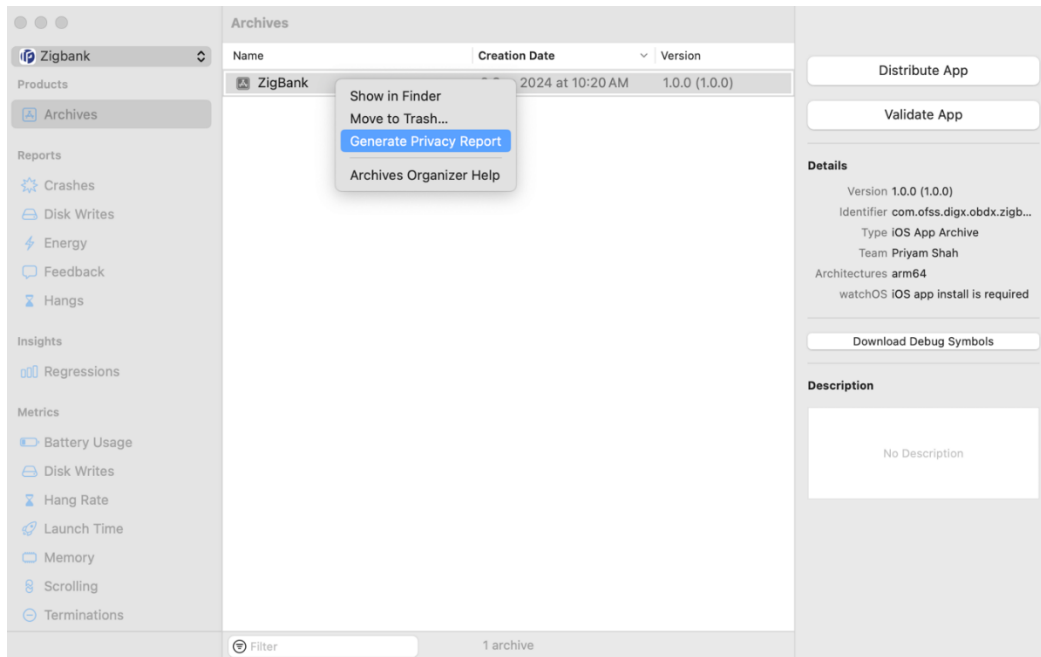
2. Additionally, there is "PrivacyInfo.xcprivacy" file added at Zigbank target level inside Resources folder

Bank needs to add additional items as per their bank customize code in "PrivacyInfo.xcprivacy" file. They can refer Apple documentation link for further details. Also, Apple's mail can contain details of what all is missing in the

PrivacyInfo.xcprivacy. Those items can be added as per Apple's doc.

Note: This step is not mandatory but if there any reference of such file in Apple's mail, bank needs to add privacy items details in application target's "PrivacyInfo.xcprivacy" file

3. Once added, build can be archived, and in organizer, right click on the application and Generate Privacy Report. This report will have the only the details of Nutrition label added in the application. Check if all nutrition labels declared in "PrivacyInfo.xcprivacy" file are present in the generated reported.



4. Generate the application and upload to Appstore for Apple Review.

5. Make IOS Application Ready for Production Checklist

Apart from Apple AppStore Submission guidelines, below are checklists in IOS workspace

- Confirm the bundle identifiers and profiles are AppStore Profiles
- Confirm the suit name in app.plist is matching App Groups mapped to profile and set in Target settings
- Server URL is correct in app.plist. Recommended is to use https URL with SSL certificate from a valid trust Authority
- Recommended to enable SSL Pinning. If enabled, check all the configurations as stated in configuration section
- App icons, splash images are updated
- Bundle version is set in "Bundle version string (short)" in info.plist and in ZigBank Target -> Info and in marketing version in Watch target->Build Settings.
- Push notification configurations are set to prod in config table and APNS keys are correct
- If Associated domains are enabled for usage then remove ?mode=developer
- Other IOS workspace configurations are followed as per section Configurations for the IOS application:
- App Transport security is enabled by setting "Allow Arbitrary Loads" to NO for all target's info.plist. Test this once with production URL
- InspectableWebView property is set to false in config.xml