Oracle B2C Service

Cobrowse In-App SDK for iOS and Android

20B
## Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Downloading the In-App SDK</td>
<td>1</td>
</tr>
<tr>
<td>Download the SDKs</td>
<td>1</td>
</tr>
<tr>
<td>2 Configuring the In-App SDK for iOS</td>
<td>3</td>
</tr>
<tr>
<td>In-App SDK for iOS</td>
<td>3</td>
</tr>
<tr>
<td>Import the Library</td>
<td>3</td>
</tr>
<tr>
<td>Import the Framework</td>
<td>3</td>
</tr>
<tr>
<td>Import the Header File</td>
<td>4</td>
</tr>
<tr>
<td>Application States</td>
<td>5</td>
</tr>
<tr>
<td>OCBManager Methods and Properties</td>
<td>5</td>
</tr>
<tr>
<td>3 Configuring the In-App SDK for Android</td>
<td>7</td>
</tr>
<tr>
<td>In-App SDK for Android</td>
<td>7</td>
</tr>
<tr>
<td>Install the SDK</td>
<td>7</td>
</tr>
<tr>
<td>Integrate the SDK</td>
<td>7</td>
</tr>
</tbody>
</table>
Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center Documentation.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit Oracle's Accessibility Program at https://www.oracle.com/corporate/accessibility/.

Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Contacting Oracle

Access to Oracle Support

Customers can access electronic support through Oracle Support. For information, visit https://cx.rightnow.com/app/landing/get_started_support or visit https://www.oracle.com/corporate/accessibility/learning-support.html if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides. You can complete one of the following surveys:

- For guides: Oracle Service Cloud - Documentation Feedback.
- For tutorials: Oracle Service Cloud - Tutorial Feedback.
1 Downloading the In-App SDK

Download the SDKs

Download the SDKs from the Oracle Software Delivery Cloud.

1. Visit [Oracle Software Delivery Search Software](#).
2. Search for Oracle Cobrowse SDK (with no space or dash).
3. Click **Select Platform** and select **Apple iOS**, **Google Android**, or both.
4. Click **Select**.
5. Click **Continue**.
6. Click **Continue**.
7. Select the check box to accept the Oracle Standard Terms and Restrictions, and click **Continue**.
8. Click **Download**.
2 Configuring the In-App SDK for iOS

In-App SDK for iOS

Oracle Standalone Cobrowse capability can be added to a native iOS application environment to enable cobrowsing of in-app content.

The following procedures outline the steps required to add Oracle Standalone Cobrowse capability to a native iOS application.

- Import the Library
- Import the Framework
- Import the Header File

Import the Library

Import the In-App Cobrowse library into the iOS application where cobrowse capability is required.

1. Right click on the project in the Navigator sidebar and select Add file to project name. The File dialog window will open with OracleCobrowseSDK.framework selected.
2. Click Add.

Import the Framework

Import the framework into the iOS application where cobrowse capability is required.

Import SystemConfiguration.framework.

Your Framework folder should look like this image:
Import the Header File

Follow this procedure to import the header file.

1. In your AppDelegate.m, import the header file of the OCBManager.h class.
   - Objective-C: #import <OracleCobrowseSDK/OCBManager.h>
   - Swift: import OracleCobrowseSDK

2. Locate -application:didFinishLaunchingWithOptions: and add the following lines.
   - Objective-C: OCBManager *manager = [[OCBManager sharedInstance] startWithURL: <your_launcher_url>];

   ```
   #import "AppDelegate.h"
   #import <OracleCobrowseSDK/OCBManager.h>
   @implementation AppDelegate
   - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
     // Override point for customization after application launch.
     // Oracle Cobrowse SDK
     [[OCBManager sharedInstance] startWithURL:@"<YOUR_LAUNCHER_URL>"];
     return YES;
   }
   ```

   - Swift: let manager = OCBManager.sharedInstance() manager.start(withURL: "<your_launcher_url>"

The value for the launcherURL parameter will be provided by your Oracle Cobrowse implementation team. This launcherURL controls the design and position of the Cobrowse launch point.
You'll then see the Cobrowse button appear, as designed, within a second after launching your application.

## Application States

At each moment of time, the Cobrowsing application is in an application state. The application state determines application functionality.

<table>
<thead>
<tr>
<th>Application State</th>
<th>Description</th>
<th>Available Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCBSessionStateNone</td>
<td>Application is not initialized yet.</td>
<td>Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td>OCBSessionStateInit</td>
<td>Application is initializing.</td>
<td>Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td>OCBSessionStateReady</td>
<td>Application has finished initializing, no active</td>
<td>• Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td></td>
<td>cobrowsing session is detected.</td>
<td>• Start cobrowsing session.</td>
</tr>
<tr>
<td>OCBSessionStateStarting</td>
<td>Application is in the process of starting a</td>
<td>• Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td></td>
<td>cobrowsing session. An access code has not yet</td>
<td>• Stop cobrowsing session.</td>
</tr>
<tr>
<td></td>
<td>been received.</td>
<td></td>
</tr>
<tr>
<td>OCBSessionStateActiveWait</td>
<td>A cobrowsing session ID was generated but an agent has</td>
<td>• Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td></td>
<td>not yet connected.</td>
<td>• Stop cobrowsing session.</td>
</tr>
<tr>
<td>OCBSessionStateActive</td>
<td>Session is in progress. At least one agent is</td>
<td>• Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td></td>
<td>connected.</td>
<td>• Stop cobrowsing session.</td>
</tr>
<tr>
<td>OCBSessionStateDisconnecting</td>
<td>Application is in process of terminating the</td>
<td>Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td></td>
<td>cobrowsing session.</td>
<td></td>
</tr>
<tr>
<td>OCBSessionStateError</td>
<td>Application or environment has encountered a</td>
<td>Subscribe to cobrowsing events.</td>
</tr>
<tr>
<td></td>
<td>fatal error.</td>
<td></td>
</tr>
</tbody>
</table>

## OCBManager Methods and Properties

The methods and properties listed here can be accessed through the OCBManager singleton class.

Before any methods described below are called, the caller must make sure that the Cobrowsing application is in the required application state in order to proceed. See Application States.

<table>
<thead>
<tr>
<th>Method</th>
<th>Required Application State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+(OCBManager *) sharedInstance</td>
<td>All</td>
<td>Returns OCBManager instance.</td>
</tr>
</tbody>
</table>
### Configuring the In-App SDK for iOS

<table>
<thead>
<tr>
<th>Method</th>
<th>Required Application State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-(void)_startWithURL: (NSString *)url</td>
<td>All</td>
<td>This method has several functions. The very first call initializes the OCBManager engine, reading all necessary data from the server, and saves the current activity context. Subsequent calls update activity context which is used by the OCBManager engine. You should call this method after each activity transition. The value for the launcherURL parameter will be provided by your Oracle Cobrowse implementation team.</td>
</tr>
<tr>
<td>-(void)_startSessionWithURL: (NSString *)url</td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>-(void)_generateAccessCodeWithCompletionHandler</td>
<td>Ready</td>
<td>Sends a request to the Oracle servers to start a cobrowsing session. Success or failure of the request can be further tracked through activity state.</td>
</tr>
<tr>
<td>-(void)_sessionDisconnect</td>
<td>Starting, Active Waiting, Active</td>
<td>Initiates a request to stop the cobrowsing session. Success or failure of the request can be further tracked through activity state.</td>
</tr>
<tr>
<td>NSString _launcherURL</td>
<td>All</td>
<td>Gets the Cobrowse iOS In-App SDK launcherURL.</td>
</tr>
<tr>
<td>OCBSessionState state</td>
<td>All</td>
<td>Returns the OCBSessionState.</td>
</tr>
<tr>
<td>NSString _SDKVersion</td>
<td>All</td>
<td>Gets the Cobrowse iOS In-App SDK version.</td>
</tr>
<tr>
<td>NSString _supportedOSVersion</td>
<td>All</td>
<td>Gets the supported OS version.</td>
</tr>
</tbody>
</table>

**Note:** The first call configures OCBManager to use a custom or default UI. There is no way to reconfigure OCBManager after the first call.
3 Configuring the In-App SDK for Android

In-App SDK for Android

Oracle Standalone Cobrowse capability can be added to a native Android application environment to enable cobrowsing of in-app content.

The following procedures outline the steps required to add Oracle Standalone Cobrowse capability to a native Android application.

- **Install the SDK**
- **Integrate the SDK**

Install the SDK

Follow this procedure for setting up the SDK.

Upgrading: If you were previously using another release of the Oracle Cobrowse Android SDK, delete all cobrowse source files from your project before proceeding.

Make sure to include Android v4 Support Library (revision 19.1.0 or greater) in your project. See *Support Library* for more information on support libraries.

1. Download the latest version of the Oracle Cobrowse Android SDK.
2. Open the **Modules** settings in Project Structure.
3. Click the + sign.
4. Select **Import JAR/AAR Package** and click **Next**.
5. Provide the path to the LLAndroidLibrary.aar file and click **Finish**.
6. Click **Apply** to synchronize your project.
   
   After synchronization is complete, you see the LLAndroidLibrary in the modules list.
7. Open the **Dependencies** settings in Project Structure.
8. Select the **app** module.
9. Select **Module Dependency** in the Declared Dependencies section and click the + sign.
10. Select **LLAndroidLibrary** and click **OK**.
11. Click **Apply** to synchronize your project.

The Cobrowse In-App SDK for Android library set up is complete. You can close the Project Structure settings.

Integrate the SDK

Follow this procedure for integration.

1. In your application's manifest **AndroidManifest.xml**, add your launcher URL as Meta Data.
2. In **AndroidManifest.xml**, add the following inside the application tag:

   ```xml
   <meta-data android:name="com.oracle.cobrowse.android.sdk.LauncherURL"
   android:value="launcherURL"/>
   
   The value for the launcherURL parameter will be provided by Oracle.
   
3. In **AndroidManifest.xml**, request the following permissions:

   ```xml
   <uses-permission android:name="android.permission.INTERNET" /> 
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   
   4. Import the Cobrowse package by adding the following to the import list of your activity:

   ```java
   import com.oracle.cobrowse.android.sdk.
   
   5. Subclass **FragmentActivity** or **Activity** for your activity class.

   6. In **onResume**, call the `start(activity)` method:

   ```java
   CobrowseManager.getInstance().start(this).
   
   7. Repeat steps 4 through 6 for every other activity in your application.

   Calling `CobrowseManager.getInstance().start(this)` in every activity will reconnect it to the already opened session instead of creating a new one.

   8. By default, the SDK does not capture dialogs. If you have dialogs in your app and you want to share them with the agent during a cobrowse session, pass the dialog instance to the SDK as soon as it becomes visible using the `screenDialog(dialog)` method.

   ```java
   CobrowseManager.getInstance().screenDialog(dialog)
   
After you have integrated the Oracle Cobrowse SDK and have completed this procedure, everything should be up and running. You'll see the Live Expert button within a second after launching your application.