HappyFox Chat SDK for Android

Overview

HappyFox Chat SDK is a library to provide in-app Live chat support for your app's users, by connecting with HappyFox Chat. Your app's users can chat with your agents from inside your app.

Before you begin

You need an HappyFox Chat account to integrate chat widget SDK in your app.

Service Instances

HappyFox Chat is an Incredibly powerful live chat software that is simple, easy and fast. Putting experience first, HappyFox Chat is lightweight and mobile responsive. With integrations, sync and streamline data between HappyFox Chat and all your favorite business apps. HappyFox Chat helps bring the most focused and highly relevant data to every chat, this helps you to deliver personalized customer service like never before. Native applications for iOS, Android and Mac enables you to have meaningful customer conversations anywhere, anytime, anyplace.

Minimum Requirements

Android Studio 2.3.3 or higher API level - 9

Setting up SDK

Step 1 - Configure:

Create a custom Application class which extends the Application class. Be sure to mention it in your manifest as well.

```
<application
  android:name=".WidgetApplication"
```

```
  ...
/>
```

Don't forget to add the below permission in the manifest.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Obtain the necessary credentials for the widget from the Goodies section of the Happyfox Chat Account. **Initialising the Widget** The general structure of initialising a widget is given below:

```
public class WidgetApplication extends Application {

    private static HFC mWidget;

    @Override
    public void onCreate() {
        super.onCreate();
            // Call init method only once for initializing the chat SDK.
        mWidget = HFC.init(this, "<License Token>", "<Embed Token>");
  }
  // Use this method to get the get the active instance of the widget.
    public static HFC getWidget() {
        return mWidget;
    }
}
```

Step 2 - Entry point for UI:

Create an object of UserInfo with the details of the visitor. *For example*,

```
 HFCUserInfo user = new HFCUserInfo.Builder()
    .setName("Visitor") // Mandatory
    .setEmail("visitor@test.com") // Mandatory
    .setPhone("+911234567890") // Optional with prefixed country code
    .setFcmToken("FCM Token here") // Optional as of now
    .build();
```

SDK uses the below library for validating the phone number if it is provided. If the the phone number is not valid then error log will be printed. libphonenumber-android

Now set the above created HFCUserInfo object to the HFCWidget.

```
WidgetApplication.getWidget().setHFCUserInfo(user);
```

Add a clickable item (probably a button) in your UI, wherever appropriate. Set a click listener to it. Within the click listener, use the show() API of the SDK to open up the Widget UI.

```
WidgetApplication.getWidget().show(MainActivity.this);
```

Step 3 - Listening to the events when the widget is not visible during an active conversation:

1. Let the Application class implement the HFCNotificationSubscriptionListener interface *For example*,

```
public class WidgetApplication extends Application implements
HFCNotificationSubscriptionListener
```

2. Make the widget subscribe to this interface after the init is called. *For example*,

```
mWidget.subscribeForHFCNotificationListener(this);
```

3. Implement the overridden the method of the interface in the Application class.

```
    @Override
    public void onSubscribedNotificationEventReceived(Object obj) {

        // TODO: You can process the below events and use it for custom
notifications for the app.

        if (obj instanceof HFCNotificationModel.AgentEndedTheChat) {
           // TODO: Process this event.
        } else if (obj instanceof HFCNotificationModel.AgentJoinedTheChat) {
           // TODO: Process this event.
        } else if (obj instanceof HFCNotificationModel.AgentSentMessage) {
           // TODO: Process this event.
        } else if (obj instanceof HFCNotificationModel.VisitorDepartmentChanged)
{
            // TODO: Process this event.
        } else if (obj instanceof HFCNotificationModel.VisitorNameChanged) {
            // TODO: Process this event.
        } else if (obj instanceof HFCNotificationModel.VisitorEmailChanged) {
            // TODO: Process this event.
        } else if (obj instanceof HFCNotificationModel.VisitorPhoneChanged) {
           // TODO: Process this event.
        }

    }
```

Step 4 - Configuring gradle file:

Add the below dependencies in your app level build.gradle file. These dependencies are internally used by the SDK so it is mandatory to be added for the SDK to function as required.

```
implementation 'com.google.code.gson:gson:2.8.5'
implementation 'com.loopj.android:android-async-http:1.4.9'
implementation 'android.arch.lifecycle:extensions:1.1.0'
implementation 'io.michaelrocks:libphonenumber-android:8.9.10'
```

Unset Visitor Details

Call this method to forget/remove currently loaded visitor in Chat SDK. This can be called when the user logouts from his app.

```
WidgetApplication.getWidget().resetHFCWidgetData();
```

For checking the error logs in logcat, you can filter the log by using the below tag
HFCVSDKLog