Encanta Developer's Guide (Android)

Contents

For any questions, issues, errors, bugs, feature requests, or requests about the meaning of life, the universe, and everything, please contact cory.bair@citrix.com.

1. Create a new app in the Encanta web portal

    a. Go to http://app.getencanta.com
    b. Sign in, or create an account
    c. On the Your Apps page, click '+New' to create a new app
    d. Enter the name of your app and click Create

2. Add Encanta support to your Android app

    Please note that Encanta requires a `minSdkVersion` of at least 9 or greater when using the appcompat version, and 14 or greater for the holo version

    a. Add the Encanta Framework and dependencies - using gradle

        i. Retrieve the correct version of the Encanta .aar for your project based on your theme and if you are using retrofit already
        ii. If using a holo-based theme, retrieve the .aar with 'holo' in the name
        iii. Similarly for an appcompat-based theme, retrieve the .aar with 'appcompat' in the name
        iv. If not using retrofit, or if you are using a version of retrofit that is 1.4.0 or greater, retrieve the .aar with 'retrofit1.9' in the name
            1. If using a version of retrofit that is prior to 1.4.0, retrieve the .aar with 'retrofit1.3' in the name
    b. Add the Encanta library to your project by copying the Encanta .aar to your application's /libs/ directory

i. Make sure the .aar you plug in has the correct permutation of holo/appcompat and retrofit version that fits your app
ii. The possible .aar files are:
    1. encanta-appcompatRetrofit1.3.aar
    2. encanta-appcompatRetrofit1.9.aar
    3. encanta-holoRetrofit1.3.aar
    4. encanta-holoRetrofit1.9.aar

c. Add System Dependencies to your project

i. Add the necessary dependencies to your application's build.gradle:

```
dependencies {
    compile 'com.android.support:appcompat-v7:22.2.0'
    compile 'com.pusher:pusher-java-client:0.3.3'
    compile 'com.squareup.retrofit:retrofit:1.9.0'
    compile ('com.citrix.saas.encanta:encantaFilename:@aar') {
        transitive = true
    }
}
```

1. If you are already using the libraries mentioned here, you should remain using the same versions that you are currently using. For any incompatibilities or issues, please contact Cory Bair at cory.bair@citrix.com.

ii. Update your application's build.gradle to include the following:

```
repositories {
    flatDir {
        dirs 'libs'
    }
    maven {
        url 'http://clojars.org/repo'
    }
}
```

3. Basic Functions

a. Import Encanta to your app
    i. In every class that will utilize Encanta, you can import the library by adding this import statement:

```
import com.citrix.saas.encanta.Encanta;
```

b. Register your app with Encanta

To register, add the following line of code to your project:

```
Encanta.init(this, "YOUR_ENCANTA_APP_TOKEN");
```

Where `YOUR_ENCANTA_APP_TOKEN` is the token for your app. This can be found under App Settings in the Encanta web portal.

The best place to add the above code is in the `onCreate()` method of your root `Application` class.

c. Show the Encanta messaging UI

To show the Encanta messaging UI, paste the following line of code at the desired location (assuming `this` is an instance of the current activity):

```
Encanta.startSupportChatActivity(this);
```

For a further example, here is how you can show the Encanta messaging UI through a button press:

```
Button startEncantaButton =
(Button)findViewById(R.id.your_encanta_button);
startEncantaButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Encanta.startSupportChatActivity(YourActivity.this);
    }
});
```

d. Optionally define the Encanta messaging activity in your AndroidManifest.xml. This will allow a custom title in the action bar, as well as allowing a back button by defining the parent activity.

```
<activity
  tools:replace="android:label"
  android:name="com.citrix.saas.encanta.GetSupportChatsActivity"
  android:label="@string/your_screen_title"
```

```
  android:parentActivityName=".YourParentActivity" >
</activity>
```

e. Optional: Handle incoming messages notification

A custom listener can be set which will be called when a new message is received through Encanta. To define this listener:

```
Encanta.setMessageReceivedListener(new OnNewMessageReceived() {
    @Override
    public void receivedMessage() {
        // Handle event passed after init() to state if there are
        // any unread Encanta messages for this user
    }

    @Override
    public void receivedMessage(String message) {
        // Handle event passed when a new message through Encanta
        // is received by the user. Show any custom
        // badging/dialog/toast/notification/etc
    }
});
```

f. Optional: Handle Encanta chat enabled/disabled being toggled

A custom listener can be set which will be called when real-time chat is toggled through the Encanta web portal. To define this listener:

```
Encanta.setChatToggledListener(new OnChatEnabledToggled() {
    @Override
    public void onChatToggled(boolean enabled) {
        // Handle event when real-time chat is enabled or disabled
        // through the web console, possibly by hiding the entry
        // point to the Encanta messaging UI.
    }

    @Override
    public void onEncantaInitialized(boolean isChatEnabled) {
        // After initialization is complete, this is triggered
        // To notify your app if real time chat is enabled or not
    }
});
```

g. Alternately, define these listeners in the init call

```
Encanta.init(this, "YOUR_ENCANTA_APP_TOKEN",
newMessageReceivedListener, chatToggledListener);
```

h. Set the customer's name

By default, your users will appear in the Encanta web portal and identified using auto-generated names (e.g., "User 951"). You can provide an actual name as follows:

```
Encanta.setUserName("Android Bob");
```

4. Advanced Functions

a. Enable push notifications

When a user receives a new Encanta message from the web portal, the Encanta server can send your app a GCM push notification. To configure push notifications, you first need to configure push notifications for your app (https://developers.google.com/cloud-messaging/android/client).

Next, you need to configure your app for push notifications through the Google Play portal and obtain your GCM API key. You must then upload this API key to Encanta through the Encanta web portal in going to the 'App Setup' section for your app.

Finally, at runtime, your app needs to provide the device token to Encanta by calling the following API:

```
Encanta.registerForPush("Device_Token");
```

The device token is provided by Google after you register your application with the GCM servers.

b. Set the user ID

This will be useful for users that must log in, or that might use multiple devices and their Encanta chat sessions should be preserved among devices. Please note that it MUST be called before .init() is called. Restrictions for the user ID are that it can only be alphanumeric (a-zA-Z0-9) or contain '@', '.', or '-'.

```
Encanta.setUserId("YourUserId");
```

c. Get the Encanta version

Retrieve the version string for the Encanta framework with the following call:

```
Encanta.getEncantaVersion();
```

d. Retrieve the unread message count

Retrieve the number of unread Encanta messages with the following call:

```
Encanta.getUnreadMessageCount();
```

e. Determine if messaging is enabled

In-app messaging can be enabled or disabled via the web portal. When in-app messaging is disabled, you may wish to hide any messaging-related UI in your application. You can optionally force a blocking API call if you want to directly retrieve this from the server. Retrieve the status of in-app messaging with the following call:

```
Encanta.isEnabled(true|false);
```

f. Set custom properties

The Encanta web portal displays a number of properties for each user running a registered app. This default property set includes:

i. Application version: the version of your app
ii. Sign-up date: the date when the user first started using your app
iii. App Opens: the number of times the user opens your app

In addition to this default set of properties, Encanta allows your app to set any number of custom properties. Each property consists of a name and a value. To set a property, use one of the following APIs:

```
Encanta.setStringProperty("Property Name", "Value");
```

```
Encanta.setIntProperty("Property Name", 1);
```

To increment a numeric property, use the following API:

```
Encanta.incrementProperty("propertyName");
```

g. Let Encanta prompt for the user's name when first viewing the Encanta messaging activity

By default, Encanta will not prompt for a user's name and will instead use an auto-generated name (assuming you didn't call Encanta.setUserName(). To prompt for a dialog to pop up when first starting the Encanta messaging activity:

```
Encanta.setPromptForUserName(true);
```

Note that this dialog's styling can be customized, as defined in a later section.

h. Customize default colors

The default colors of the Encanta messaging activity can be customized to better match the style of your application by adding the following to your colors.xml:

```
<!-- Background color to use for the 'sent' message bubble -->
<color name="encanta_sent_message_bg">#5d7c81</color>
<!-- Background color to use for the 'received' message bubble -->
<color name="encanta_received_message_bg">#8d9cf1</color>
<!-- Background color to use for the messages activity -->
<color name="encanta_activity_bg">#114478</color>
<!-- Color to use for the action bar -->
<color name="encanta_action_bar_color">#225555</color>
<!-- Entry text color in the footer -->
<color name="encanta_chat_entry_text_color">#ff000000</color>
<!-- Background color of the footer text entry -->
<color name="encanta_footer_background">#D7D5D5</color>

<!-- Various colors within the popup, only used when
Encanta.setPromptForUserName(true) is called -->
<color name="encanta_popup_message_bg">#1F2028</color>
<color name="encanta_popup_text">#FF000000</color>
<color name="encanta_popup_pressed">#506B6F</color>
<color name="encanta_popup_button_unpressed">#6BBEB8</color>
```

```
<color name="encanta_popup_button_background">#6DBEB8</color>
<color name="encanta_popup_title_text">#ff000000</color>
<color name="encanta_popup_description_text">#535459</color>
```

i. Customize default strings

The default strings used in Encanta can be customized to better fit the needs of your application by adding the following to your strings.xml:

```
<string name="encanta_action_bar_label">Support Chat</string>
<string name="encanta_send_support_chat_hint">Send Message</string>
<string name="encanta_no_chat_messages_text">Have questions? Send us a
message! We are here to help.</string>
<string name="encanta_disabled_text">We are currently out of the
office. Please leave a message and we will respond to your query upon
our return.</string>

<!-- Used in the name entry popup, which is only shown when
Encanta.setPromptForUserName(true) is called -->
<string name="encanta_name_entry_popup_title">Support Chat</string>
<string name="encanta_name_entry_popup_description">Please enter your
name to start your support chat</string>
<string name="encanta_name_entry_popup_hint_text">Your name</string>
<string name="encanta_name_entry_popup_submit">OK</string>
```

j. Pause notifications from being sent to the user

If there is a state your app can be in where it would be detrimental for your custom-defined notifications to appear (for example, when the user is in the middle of a VoIP call), the following can be used to pause and resume notifications from being sent to your custom listeners:

```
Encanta.pauseNotifications(true);
```