

# LiveEngage Enterprise In-App Messaging SDK Deployment Guide: Android v.2.1.3

## [Quick Start](#)

### [Prerequisites](#)

#### [Step 1: Download and unzip the SDK](#)

#### [Step 2: Configure project settings to connect LiveEngage SDK](#)

#### [Step 3: Code integration for basic deployment](#)

## [Advanced Configurations](#)

### [SDK Initialization and Lifecycle](#)

#### [Initialization](#)

#### [LivePerson Callbacks Interface](#)

### [Authentication](#)

### [Conversation Lifecycle](#)

### [Notifications](#)

### [User Data](#)

### [Logs and Info](#)

## [Methods](#)

### [Initialize \(Deprecated\)](#)

### [Initialize \(with SDK properties object\)](#)

### [showConversation](#)

### [showConversation \(with authentication support\)](#)

### [hideConversation](#)

### [getConversationFragment](#)

[getConversationFragment \(with authentication support\)](#)

[reconnect](#)

[setUserProfile](#)

[setUserProfile \(deprecated\)](#)

[registerLPPusher](#)

[unregisterLPPusher](#)

[handlePush](#)

[getSDKVersion](#)

[setCallback](#)

[removeCallBack](#)

[checkActiveConversation](#)

[checkAgentID](#)

[markConversationAsUrgent](#)

[markConversationAsNormal](#)

[checkConversationIsMarkedAsUrgent](#)

[resolveConversation](#)

[shutDown](#)

[shutDown \(deprecated\)](#)

[ClearHistory](#)

[logout](#)

[Interface and class definitions](#)

[AgentData](#)

[InitLivePersonProperties](#)

[ConsumerProfile](#)

[Callbacks Index](#)

## [LivePersonCallback](#)

[Error indication](#)

[Token Expired](#)

[Conversation started](#)

[Conversation resolved](#)

[Connection state has changed](#)

[Agent avatar tapped](#)

[Agent details changed](#)

[Agent typing](#)

[CSAT Screen dismissed](#)

[CSAT Screen submitted](#)

[Conversation marked as urgent](#)

[Conversation marked as normal](#)

[Offline Hours Changes](#)

## [LogoutLivePersonCallback](#)

## [ICallback](#)

## [InitLivePersonCallBack](#)

## [ShutDownLivePersonCallback](#)

## [Configuring the SDK](#)

### [Attributes](#)

[Brand](#)

[Brand Message Bubble - the first brand message](#)

[Agent Message Bubbles](#)

[Consumer Bubbles](#)

[System messages](#)

[Unread messages indicator Bubbles](#)

[Survey screen](#)

[Message Edit Text](#)

[Connection status bar](#)

[In page navigation - Scroll down indicator](#)

[Photo Sharing](#)

[General Style](#)

[Conversation Activity Style - \(activity mode only!\)](#)

[Accessibility](#)

[Miscellaneous](#)

[Deprecated Attributes](#)

[Configuring the message's EditText](#)

[ProGuard Configuration \(Android only\)](#)

[String localization in SDK](#)

[Modifying String](#)

[Modifying resources](#)

[Plural String Resource Example](#)

[Timestamps Formatting](#)

[Off Hours](#)

[Date & Time](#)

[Timezone](#)

[Bubble timestamp](#)

[Separator timestamp](#)

[Resolve message](#)

[CSAT Behavior](#)

[Overview](#)

[Show CSAT flow](#)

[Dismiss CSAT](#)

[CSAT UI content](#)

[agentView \(avatar and agent name\)](#)

[ratingQuestionView \(stars\)](#)

[resolutionConfirmationView \(yes/no\)](#)

[Photo Sharing - Beta](#)

[Overview](#)

[Enable Photo Sharing](#)

[LiveEngage Configuration](#)

[Enable Push Notifications](#)

[Appendix](#)

[Security](#)

[Dependencies](#)

[Open Source List](#)

[Localization Strings](#)

[Demo Project](#)

[Project structure explained](#)

[MainActivity class](#)

[setCallBack method](#)

[initActivityConversation method](#)

[openFragmentContainer method](#)

[FragmentContainerActivity class](#)

[Push package](#)

[branding.xml](#)

# Quick Start

The LivePerson In-App Messaging SDK provides brands with a simple, yet enterprise-grade and secure in-app messaging solution. Through in-app messaging, brands will foster connections with their customers and increase app engagement and retention.

This Quick Start will quickly get you up and running with a project powered by LivePerson. When you're done, you'll be able to send messages between an Android device and LiveEngage.

## Prerequisites

To use the LivePerson In-App Messaging SDK, the following are required:

- LiveEngage account with messaging enabled
- **Embeddable library for AAR:** Binary distribution of an Android Library Project
- **Installers:** Gradle

*Note: For information on supported operating systems and devices, refer to [System Requirements](#).*

## Step 1: Download and unzip the SDK

Follow the steps below to download and unzip the

1. Download the latest Messaging SDK from the following link: [SDK Repository](#).
2. Extract the ZIP file to a folder on your computer.

The downloaded package should contain the following three items:

- LP\_Messaging\_SDK/lp\_messaging\_sdk - Module that should be added to your project. This module contains the following:
  - LivePerson.java - Main entry point for the Messaging SDK
  - Resources (.aars files)
- SampleApp-Source - demonstrate how to use the Messaging SDK.
- SampleApp-APK - sample app installation file.

## Step 2: Configure project settings to connect LiveEngage SDK

Follow the steps below to configure the project settings to connect to the SDK.

1. Import the downloaded lp\_messaging\_sdk module into your project.
  - In the Android Studio menu bar, select: **File** → **New** → **Import module**.

- Navigate to the folder where you extracted the SDK project. Navigate to the `lp_messaging_sdk` module, and click **Finish**.
2. Add the following lines to the `build.gradle` of your app :
- `compileSdkVersion` and `buildToolsVersion` (should be at least Version 23).
  - Add the following code under the Android section:

```
repositories {  
    flatDir {  
        dirs project(':lp_messaging_sdk').file('aars')  
    }  
}
```

3. Under the Dependencies section, add the following line:

```
compile project(':lp_messaging_sdk')
```

*Example: Build.gradle file*

```
apply plugin: 'com.android.application'  
  
android {  
    compileSdkVersion 24  
    buildToolsVersion "24.0.3"  
  
    repositories {  
        flatDir {  
            dirs project(':lp_messaging_sdk').file('aars')  
        }  
    }  
}
```



```

}

defaultConfig {
    applicationId "xxx"

    minSdkVersion xx

    targetSdkVersion xx

    versionCode 1

    versionName "1.0"
}

buildTypes {
    release {
        minifyEnabled false

        proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
    }
}
}

dependencies {
    compile project(':lp_messaging_sdk')
}

```

### Step 3: Code integration for basic deployment

1. Add the following permission to your app's AndroidManifest.xml file:

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

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

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

2. Add the following imports to your class imports section:

```
import com.Liveperson.api.LivePersonCallback;
```

```
import com.Liveperson.infra.InitLivePersonProperties;
```

```
import com.Liveperson.infra.callbacks.InitLivePersonCallBack;
```

```
import com.Liveperson.messaging.TaskType;
```

```
import com.Liveperson.messaging.model.AgentData;
```

```
import com.Liveperson.messaging.sdk.api.LivePerson;
```

3. Initialize the Messaging SDK

You can initialize the SDK in your Activity before showing LivePerson's Activity/Fragment, but it is recommended to initialize the SDK once, in your app's Application class.

```
String brandID = "YourLivepersonAccountIdString";

String appID = "your app package name"

LivePerson.initialize(MainActivity.this, new InitLivePersonProperties(brandID, appID,
new InitLivePersonCallBack() {

    @Override

    public void onInitSucceed() {

    }

    @Override

    public void onInitFailed(Exception e) {

    }

})
```

```
});
```

| Element       | Description   |
|---------------|---|
| brandID       | Your LivePerson account ID. If you don't have one, please contact your LivePerson representative. |
| appID         | Your app id, used for registering LP pusher service.  |
| onInitSuccess | Callback that indicates the init process has finished successfully.                               |
| onInitFailed  | Callback that indicates the init process has failed.  |

*Example implementation:*

```
LivePerson.initialize(context, new InitLivePersonProperties(brandID, appID, new
InitLivePersonCallback() {

    @Override

    public void onInitSucceed() {

        initFragment();

        LivePerson.setUserProfile(appId, firstName, lastName, phone);

    }

    @Override

    public void onInitFailed(Exception e) {

        Toast.makeText(MainActivity.this, "Init Failed",
Toast.LENGTH_SHORT).show();

    }

});
```

#### 4. Show conversation screen.

The SDK supports two operation modes:

- Activity mode
- Fragment mode

### Activity mode

Activity mode implements the toolbar that displays the agent name the consumer is talking with. The 'Is Typing' indicator displays when the agent is typing and the menu button. In addition to this, when using the activity mode, the SDK deals with initializing the SDK.

To open conversation window in separate activity. This will start a new conversation activity:

```
LivePerson.showConversation(getActivity());
```

Using this method the SDK implements the controls on the action bar.

### Fragment mode

In fragment mode the SDK returns the conversation fragment to the caller that needs to be placed inside a container. Also, the caller is responsible for initializing the SDK and, if needed, implementing a toolbar or other indicators according to the provided SDK [callbacks](#).

*Note: Ensure that the init process finished successfully. These should be called from the [onInitSucceed\(\)](#) callback.*

To open conversation window in a fragment: This returns a conversation fragment to be placed in a container in your activity:

```
LivePerson.getConversationFragment();
```

When using fragment mode, you should use the provided SDK callbacks in your app in order to implement functionalities such as menu items, action bar indications, agent name, and typing indicator.

## Fragment mode - Handle CSAT (feedback)

In Fragment mode, there is an option to get notified of the CSAT screen state (visible/invisible). For example- show different title on Toolbar , show a close csat button etc...

The container Activity (the Activity that hosts the fragment) needs to implement ConversationFragmentCallbacks interface:

```
public interface ConversationFragmentCallbacks {

    void setFeedBackMode(boolean on, IFeedbackActions actions);

    // boolean on - Notify whether feedback (csat) screen is visible or dismisses.
    // IFeedbackActions actions - provides set of actions for the feedback screen.

    void onSurveySubmitted(IFeedbackActions actions);

    // IFeedbackActions actions - provides set of actions for the feedback screen.
}

public interface IFeedbackActions {

    void closeFeedBackScreen();

    //close the screen, for example- after submitting the csat. When showing "thank you"
    //screen.

    void skipFeedBackScreen();

    //skip and close the whole feedback process.
}
```

Once the CSAT screen is visible, “setFeedBackMode” will be called with “true” value, when the CSAT is not visible anymore (skip/submitted) - “setFeedBackMode” will be called with “false” value.

Example - how to use “ConversationFragmentCallbacks” (code from the container Activity)

```
class ContainerActivity extends FragmentActivity implements ConversationFragmentCallbacks {

    @Override
    public void setFeedBackMode(boolean on, final IFeedbackActions actions) {
        toolbar.setTitle("Csat mode: " + ( on ? "on!" : "off!" ));
        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (actions != null){
                    actions.closeFeedBackScreen();
                }
            }
        });
    }
}
```

```
        }

    }

    });

}

@Override
public void onSurveySubmitted(IFeedbackActions actions) {
    toolbar.setTitle("survey submitted");
}

...
}
```

# Advanced Configurations

## SDK Initialization and Lifecycle

### Initialization

Add the code below to initialize the SDK:

```
String brandID = "Your-Liveperson-Account-Id-String";

String appID = "your-app-package-name"

LivePerson.initialize(context, new InitLivePersonProperties( brandID, appID,

    new InitLivePersonCallBack() {

        @Override

        public void onInitSucceed() {

        }

        @Override

        public void onInitFailed(Exception e) {

        }

    }

));
```

| Element       | Description  |
|---------------|--|
| brandID       | Your LivePerson account ID. If you don't have one, please contact your LivePerson representative.  |
| appID         | Your app ID, used for registering LP pusher service.   |
| onInitSuccess | Callback that indicates the init process has finished successfully.  |
| onInitFailed  | Callback that indicates the init process has failed.<br><i>Note: You can call initialize before showing LivePerson's Activity/Fragment, but it is recommended to initialize the SDK in your app's Application class.</i> |

Once initialization is completed (onInitSucceed), you can call LivePerson methods.

The SDK supports two operation modes: Activity and Fragment. For more information about each mode, refer to [Step 3: Code integration for basic deployment](#).

To start LivePerson's Activity mode:

```
LivePerson.showConversation(Activity activity);
```

To start LivePerson's Fragment mode: (Attach the returned fragment to a container in your activity) :

```
LivePerson.getConversationFragment();
```

When using fragment mode, you could use the provided SDK callbacks in your app in order to implement functionalities such as menu items, action bar indications, agent name, and typing indicator.

### LivePerson Callbacks Interface

The SDK provides a callback mechanism to keep the host app updated on events related to the conversation.

To register the callback call:

```
LivePerson : public static void setCallback(final LivePersonCallback listener)
```

To remove a callback:

```
LivePerson : public static void removeCallBack()
```

Click [here](#) for more information.

### Shut Down

Close LivePerson Messaging SDK- Uninitialized SDK without cleaning data.

```
public static void shutDown(final ShutDownLivePersonCallback shutdownCallback)
```

Click [here](#) for more information.

### Logout

Close LivePerson Messaging SDK- Clear LivePerson Messaging SDK data and unregistering push.



```
public static void logOut(final Context context, final String brandId, final
String appId, final LogoutLivePersonCallback logoutCallback)
```

Click [here](#) for more information.

## Authentication

For users of OAuth 2.0 for customer authentication, the following functions apply:

To start LivePerson's Activity mode:

```
LivePerson : LivePerson.showConversation(Activity activity, String authKey);
```

To start LivePerson's Fragment mode: (Attach the returned fragment to a container in your activity):

```
LivePerson : LivePerson.getConversationFragment(String authKey);
```

Once Authentication key is expired, you will be notified with callback "[void onTokenExpired\(\)](#)".

To re-connect with new Authentication key:

```
LivePerson : public static void reconnect\(String authKey\)
```

*Note: errors while trying to connect will call callback : void onError(TaskType type, String message);*

## UI

To determine the layout of messaging within the app, you can utilize various actions to control the behavior and UI such as menus, typing indication, etc.

LivePerson callbacks:

```
void onAgentTyping(boolean isTyping);
void onAgentDetailsChanged(AgentData agentData);
void onCsatDismissed();
void onCsatSubmitted(String conversationId);
void onConversationMarkedAsUrgent();
void onConversationMarkedAsNormal();
void onOfflineHoursChanges(boolean isOfflineHoursOn);
void onAgentAvatarTapped(AgentData agentData);
```

## Conversation Lifecycle

During the course of the conversation, consumers can take several actions such as Mark as urgent to receive a faster service, or Resolve conversation to let your agents know they have received their answers.

LivePerson API:

```
public static void checkActiveConversation(final ICallback<Boolean,
Exception> callback)
public static void checkConversationIsMarkedAsUrgent(final ICallback<Boolean,
Exception> callback)
public static void checkAgentID(final ICallback<AgentData, Exception>
callback)
public static void markConversationAsUrgent()
public static void markConversationAsNormal()
public static void resolveConversation()
public static boolean clearHistory()
```

Note: Click [here](#) for more information.

Also via Callbacks:

```
void onConversationStarted(LPConversationData convData);
void onConversationResolved(LPConversationData convData);
void onConnectionChanged(boolean isConnected);
```

Note: Click [here](#) for more information.

## Notifications

Push and local notifications are a key factor that make the experience better for consumers - they never have to stay in your app or keep the window open as they will get a proactive notification as soon as a reply or notice is available.

*Note: In order to enable push notifications, you must also configure them within the LiveEngage UI. [See instructions here](#).*

To implement push notifications on the client side:

1. Get your app's AppKey from [Google GCM](#) or [Google FCM](#) and set it in the LiveEngage backend, as explained [here](#), to identify your app by LiveEngage.
2. On every app launch get the GCM Token from your device and register it on the LiveEngage push service using the [registerLPPusher\(\)](#) API call so it knows which device should get each push message.

3. Upon receiving a push message to your app, [handle](#) it so it is displayed to the customer.

```
public class MyGcmListenerService extends GcmListenerService {

    /**
     * Called when message is received.
     *
     * @param from SenderID of the sender.
     * @param data Data bundle containing message data as key/value pairs.
     *
     * For Set of keys use data.keySet().
     */
    @Override
    public void onMessageReceived(String from, Bundle data) {

        // Sends the message into the SDK

        LivePerson.handlePush(this, data, LpAccount, true);

    }
}
```

## User Data

Pass and display consumer information to agents, and agent information to consumers. See more information about each method [here](#).

```
public static void setUserProfile(ConsumerProfile profile)
public static void checkAgentID(final ICallback<AgentData, Exception>
callback)
```

## Logs and Info

Upon errors, we send logs include different severity levels of errors and warnings.

## Methods

The In-App Messaging SDK for Android includes the following methods:

| Name   | Description   |
|--|---|
| <a href="#"><u>Initialize</u></a> (Deprecated)                             | Initialize the resources required by the SDK.   |
| <a href="#"><u>Initialize</u></a> (with SDK properties object)             | Initialize the resources required by the SDK with properties object.  |
| <a href="#"><u>showConversation</u></a>                                    | Display the messaging activity.   |
| <a href="#"><u>showConversation</u></a> (with authentication support)      | Display the messaging activity with the addition of authentication support.   |
| <a href="#"><u>hideConversation</u></a>                                    | Hide the conversation activity.   |
| <a href="#"><u>getConversationFragment</u></a>                             | Get the conversation fragment.  |
| <a href="#"><u>getConversationFragment</u></a> with authentication support | Get the conversation fragment with the addition of authentication support.  |
| <a href="#"><u>reconnect</u></a>   | Reconnect with new authentication key.  |
| <a href="#"><u>setUserProfile</u></a>                                      | Take custom parameters about the consumer as an input, set them for the messaging agent, and attach them to the transcript. |
| <a href="#"><u>setUserProfile</u></a> (Deprecated)                         | Take custom parameters about the consumer as an input, set them for the messaging agent, and attach them to the transcript. |
| <a href="#"><u>registerLPPusher</u></a>                                    | Register to LivePerson push services.   |
| <a href="#"><u>unregisterLPPusher</u></a>                                  | Unregister from LivePerson push services.   |

|  |   |
|--|---|
| <a href="#"><u>handlePush</u></a>                        | Receive all incoming push messages in a single function.                              |
| <a href="#"><u>getSDKVersion</u></a>                     | Return the SDK version.   |
| <a href="#"><u>setCallback</u></a>                       | Get events from SDK - need to implement <b>LivePersonCallback</b> .                   |
| <a href="#"><u>removeCallBack</u></a>                    | Stop getting events from the SDK.   |
| <a href="#"><u>checkActiveConversation</u></a>           | Check whether there is an active conversation.  |
| <a href="#"><u>checkAgentID</u></a>                      | Return agent data such as, first name, last name, email, avatarURL, through callback. |
| <a href="#"><u>markConversationAsUrgent</u></a>          | Mark the current conversation as urgent.  |
| <a href="#"><u>markConversationAsNormal</u></a>          | Mark the current conversation as normal.  |
| <a href="#"><u>checkConversationIsMarkedAsUrgent</u></a> | Check whether the current conversation is marked as urgent.                           |
| <a href="#"><u>resolveConversation</u></a>               | Resolve the current conversation.   |
| <a href="#"><u>shutDown</u></a>                          | Shut down the SDK.  |
| <a href="#"><u>shutDown</u></a> (Deprecated)             | Shut down the SDK.  |
| <a href="#"><u>clearHistory</u></a>                      | Clear all conversations from device.  |
| <a href="#"><u>logOut</u></a>                            | Logout from the SDK - when all user data should be removed.                           |

## Initialize (Deprecated)

| <i>public static void initialize (Context context, String brandId, <a href="#">InitLivePersonCallBack</a> initCallBack)</i> |  |
|---|--|
| context   | A context from the host app                              |
| brandId   | An account ID  |
| initCallBack  | An <a href="#">InitLivePersonCallBack</a> implementation |

To allow user interaction, the Messaging Mobile SDK must be initiated. This API initializes the resources required by the SDK. All subsequent API calls, except to the `handlePush`, assume that the SDK has been initialized.

When the conversation screen is displayed, the server connection for messaging will be established. If a user session is already active and an additional SDK init call is made, it will be ignored and will not start an additional session.

*Note: This method was deprecated - please use the new method below.*

## Initialize (with SDK properties object)

| <i>public static void initialize (Context context, <a href="#">InitLivePersonProperties</a> initProperties)</i> |  |
|---|--|
| context   | A context from the host app                                    |
| <i>initProperties</i>   | An object with all the properties needed to initialize the SDK |

To allow user interaction, the Messaging Mobile SDK must be initiated. This API initializes the resources required by the SDK; all subsequent API calls. Except for the `handlePush`, assume that the SDK has been initialized.

When the conversation screen is displayed, the server connection for messaging will be established. If a user session is already active and an additional SDK init call is made, it will be ignored and will not start an additional session. This method gets `InitLivePersonProperties`, which includes the properties needed for the init phase of the SDK.

## showConversation

| <i>public static boolean showConversation(Activity activity)</i> |                      |
|--|----------------------|
| activity   | The calling activity |

The *showConversation* API displays the messaging screen as a new activity with the conversation fragment. The consumer can then start or continue a conversation. The conversation screen is controlled entirely by the SDK.

This method returns a Boolean value to indicate success or failure in opening the messaging screen. If the operation is successful, this method returns *true*, else it returns *false*.

Initiating the conversation screen opens the webSocket to the LivePerson Messaging Server.

## showConversation (with authentication support)

| <i>public static boolean showConversation(Activity activity, String authenticationKey)</i> |                        |
|--|------------------------|
| activity   | The calling activity   |
| authenticationKey  | The authentication key |

Same as above with the addition of authentication support. You should use this alternative if you know your system implementation involves an authentication step. Usually this means that the LivePerson backend will verify the authentication token sent by the SDK with your system servers. If the key cannot be verified on your company's backend servers, this call will fail.

## hideConversation

| <i>public static void hideConversation(Activity activity)</i> |                      |
|---|----------------------|
| activity  | The calling activity |

The *hideConversation* API hides the conversation activity. The conversation screen is shown again by calling Start Conversation.

Notes:

- *Hiding the conversation closes the websocket.*
- *When using the SDK's activity, the back button performs the same function.*

## getConversationFragment

```
public static Fragment getConversationFragment();
```

The *getConversationFragment* method creates and return the conversation fragment.

*Note: This API does not show the actual screen, but only creates the fragment. Your implementation needs to handle when and how to show it.*

## getConversationFragment (with authentication support)

```
public static Fragment getConversationFragment(String authKey)
```

|         |                        |
|---------|------------------------|
| authKey | The authentication key |
|---------|------------------------|

Same as above with the attention of authentication support. You should use this alternative if you know your system implementation involves an authentication step. Usually this means the LivePerson backend will verify the authentication token sent by the SDK with your system servers. If the key cannot be verified, or your backend isn't set up with the LivePerson backend, this call will fail.

## reconnect

```
public static void reconnect(String authKey)
```

|         |                        |
|---------|------------------------|
| authKey | The authentication key |
|---------|------------------------|



Reconnect with a new authentication key. When connecting with an authentication key, the connection may be closed once the token is expired. When this happens, the [onTokenExpired](#) callback method is called. In this case, the application needs to obtain a fresh key and reconnect by calling the *reconnect* method.

## setUserProfile

| <i>public static void setUserProfile(<a href="#">ConsumerProfile</a> profile)</i> |                    |
|---|--------------------|
| profile   | The user's profile |

The *setUserProfile* API takes custom parameters about the consumer as an input and sets it to be displayed on the messaging Agent Workspace consumer transcript. This can be set at any time either before, after, or during a messaging session.

## setUserProfile (deprecated)

**(Deprecated. Please use the [setUserProfile](#) (*String firstName, String lastName, String phone*) method above).**

| <i>public static void setUserProfile(String appId, String firstName, String lastName, String phone)</i> |                   |
|---|-------------------|
| appId   | The host app ID   |
| firstName   | User's first name |
| lastName  | User's last name  |
| phone   | User's phone      |

The *setUserProfile* API takes custom parameters about the consumer as an input and sets it to be displayed on the messaging Agent Workspace consumer transcript. This can be set at any time either before, after, or during a messaging session.

## registerLPPusher

| <i>public static void registerLPPusher(String brandId, String appld, String gcmToken)</i> |  |
|---|--|
| brandId   | The account Id (e.g. 652838922).   |
| appld   | The host app Id (e.g. com.liveperson.myApp).   |
| gcmToken  | The <a href="#">GCM Token</a> . Usually used to pass the Google provided token.<br><br>However, this parameter can contain any string value. |

*Note: If you use the gcmToken as a custom value, you need to handle the mapping between this custom value and the actual gcm token in your server.*

## unregisterLPPusher

| <i>public static void unregisterLPPusher(String brandId, String appld)</i> |                  |
|--|------------------|
| brandId  | The account ID.  |
| appld  | The host app ID. |

Unregister from registered push notification service.

## handlePush

| <i>public static void handlePush(Context context, Bundle data, String brandId, boolean showNotification)</i> |   |
|--|---|
| context  | A context from the host app.  |
| data   | A Bundle that contains the message. The bundle should hold a string with key named "message". |

|                  |   |
|------------------|---|
| brandId          | The account ID.   |
| showNotification | Used to instruct the SDK to either show or not show a notification to the user. If you wish your app will handle the display of the notification you can set this as false. |

All incoming push messages are received by the host app. The host app can choose to fully handle any push message and display a notification message, or partially handle it and allow the SDK to display the notification.

Handling the push message allows the host app to do the following:

- Receive non-messaging related push messages.
- Handle custom in-app alerts upon an incoming message.

*Note: Whether the host app fully handles any push messages or partially, any messaging push message should be sent to the SDK using the `handlePush` method.*

## getSDKVersion

```
public static String getSDKVersion()
```

Returns the SDK version.

## setCallback

```
public static void setCallback(final LivePersonCallback listener)
```

|          |   |
|----------|---|
| listener | A <a href="#">LivePersonCallback</a> implementation |
|----------|---|

Sets the SDK callback listener. The host app gets updates from the SDK using this callback listener. See [LivePerson Callbacks Interface](#) for more information.

## removeCallBack

```
public static void removeCallBack()
```

Removes the registered [LivePersonCallback](#) callback.

## checkActiveConversation

```
public static void checkActiveConversation(final ICallback<Boolean, Exception> callback)
```

|          |   |
|----------|---|
| callback | An <a href="#">ICallback</a> implementation |
|----------|---|

Checks whether there is an active (unresolved) conversation. The result will be returned to the provided callback.

## checkAgentID

```
public static void checkAgentID(final ICallback<AgentData, Exception> callback)
```

|          |   |
|----------|---|
| callback | An <a href="#">ICallback</a> implementation |
|----------|---|

If there is an active conversation, this API returns agent data through the provided callback. If there is no active conversation, the API returns null.

[AgentData definition](#)

## markConversationAsUrgent

```
public static void markConversationAsUrgent()
```

Marks the current conversation as urgent.

## markConversationAsNormal

```
public static void markConversationAsNormal()
```

Marks the current conversation as normal.

## checkConversationIsMarkedAsUrgent

```
public static void checkConversationIsMarkedAsUrgent(final ICallback<Boolean, Exception> callback)
```

|          |   |
|----------|---|
| callback | An <a href="#">ICallback</a> implementation |
|----------|---|

Checks whether the current conversation is marked as urgent. The result is returned through the provided callback.

## resolveConversation

```
public static void resolveConversation()
```

Resolves the current conversation.

## shutDown

```
public static void shutDown(final ShutDownLivePersonCallback shutdownCallback)
```

|                  |  |
|------------------|--|
| shutdownCallback | A <a href="#">ShutDownLivePersonCallback</a> implementation to get indication whether the shutdown succeeded or failed |
|------------------|--|

Shuts down the SDK and removes the footprint of the user session from local memory. After shutdown the SDK is unavailable until re-initiated. Message history is saved locally on the device and synced with the server upon reconnection.

The server continues to send push notifications when the SDK is shut down. To unregister from push services, call [unregisterLPPusher](#) API.

ShutDownLivePersonCallback callback description:

- *onShutdownSucceed()* method is called when the shutdown process finished successfully.
- *onShutdownFailed()* method is called when the shutdown process failed (for example, shutdown was called when the conversation screen is displayed in the foreground).

*Note: This does not end the current messaging conversation.*

## shutDown (deprecated)

**(Deprecated. Please use the above *shutDown(ShutDownLivePersonCallback)* method)**

```
public static void shutDown()
```

Shuts down the SDK and removes the footprint of the user session from local memory. After shutdown the SDK is unavailable until re-initiated. Message history is saved locally on the device and synced with the server upon reconnection.

The server continues to send push notifications when the SDK is shut down. To unregister from push services, call [unregisterLPPusher](#) API.

*Note: This does not end the current messaging conversation.*

*Important: This method must not be called when the conversation screen is displayed.*

## ClearHistory

```
public static boolean clearHistory()
```

Clear all conversations from the device. This clears all conversations and messages from the device only and does not remove them from the server. If the account has history enabled and is used on a new device, all conversations will be loaded from the server.

The return value indicates whether the action was completed successfully or not:

*True* - All conversations were cleared successfully.

*False* - Conversations were not cleared since there is an open conversation.

*Note: The clearHistory API call will work only if there is currently no active conversation.*

## logout

|   |   |
|---|---|
| <pre>public static void logOut(Context context, String brandId, String appld, <a href="#">LogoutLivePersonCallback</a> logoutCallback){</pre> |   |
| context   | A context from the host app.                                |
| brandId   | An account ID.  |
| appld   | The host app ID.  |
| logoutCallback  | An <a href="#">LogoutLivePersonCallback</a> implementation. |

Logout from the SDK - when all user data should be removed.

Calls [unregisterLPPusher](#), [shutDown](#) and, in addition, deletes all user data (messages and user details) from the device.

In order to unregister from push, it must be called when there is network available.

After logout the SDK is unavailable until re-initiated.

**This method does not require the SDK to be initialized.**

*Note: This does not end the current messaging conversation.*

*Important: This method must not be called when the conversation screen is displayed.*

## Interface and class definitions

### AgentData

```
public class AgentData {
```

```
public String mFirstName;  
  
public String mLastName;  
  
public String mAvatarURL;  
  
public String mEmployeeId;  
  
public String mNickName;  
  
}
```

## InitLivePersonProperties

```
Public class InitLivePersonProperties{  
  
    Private string brandId;  
  
    Private string appld;  
  
    Private InitLivePersonCallBack initCallBack;  
  
}
```

## ConsumerProfile

```
public class ConsumerProfile {  
  
    private String mFirstName;  
  
    private String mLastName;  
  
    private String mPhoneNumber;  
  
    private String mNickName;  
  
    private String mAvatarUrl;
```



```
}
```

## LPConversationData

```
Public class LPConversationData{  
    Private CloseReason closeReason;  
    Private String conversationId;  
}
```

## Callbacks Index

The SDK provides a callback mechanism to keep the host app updated on events related to the conversation. This section details each callback.

## LivePersonCallback

Definition:

```
public interface LivePersonCallback{  
    void onError(TaskType type, String message);  
    void onTokenExpired();  
    void onConversationStarted(LPConversationData convData);  
    void onConversationResolved(LPConversationData convData);  
    void onConnectionChanged(boolean isConnected);  
    void onAgentTyping(boolean isTyping);  
    void onAgentDetailsChanged(AgentData agentData);  
    void onCsatDismissed();  
    void onCsatSubmitted(String conversationId);  
    void onConversationMarkedAsUrgent();  
    void onConversationMarkedAsNormal();  
    void onOfflineHoursChanges(boolean isOfflineHoursOn);  
}
```

```
enum TaskType {  
    CSDS,  
    IDP,  
    VERSION,  
    OPEN_SOCKET  
}
```

### Error indication

The `onError(TaskType type, String message)` method is called to indicate that an internal SDK error has occurred.

| Parameter | Description  |
|-----------|--|
| type      | The type of error. Indicates the category of the error. See the table below. |
| Message   | A detailed message on the error.   |

### TaskType enum:

| Type        | Description  |
|-------------|--|
| CSDS        | Internal server error.   |
| IDP         | An error occurred during the authentication process. This is usually due to a wrong or expired authentication key. |
| VERSION     | Your host app is using an old SDK version and cannot be initialized.   |
| OPEN_SOCKET | Error opening a socket to the server.  |

### Token Expired

The `onTokenExpired()` method is called if the token used in the session has expired and no longer valid. The host app needs to [reconnect](#) with a new authentication key.

### Conversation started

The `onConversationStarted()` method is called whenever a new conversation is started by either the consumer or the agent.

## Conversation resolved

The `onConversationResolved(CloseReason reason)` method is called when the current conversation is marked as resolved by either the consumer, agent or system (auto close).

`public enum CloseReason { AGENT, CONSUMER, SYSTEM }`

Note : `onConversationResolved()` is deprecated.

## Connection state has changed

The `onConnectionChanged(boolean isConnected)` method is called when the connection to the conversation server has established or disconnected.

Parameters:

*isConnected* - indicates the connection state. true - connection establish, false - disconnected.

## Agent avatar tapped

The `onAgentAvatarTapped (AgentData agentData)` method is called when the user taps on the agent avatar.

The icon is available next to the agent message bubble or on the top of the toolbar (if using activity mode)

## Agent details changed

The `onAgentDetailsChanged(AgentData agentData)` method is called when the assigned agent of the current conversation has changed or their details are updated.

This callback is also called with null value when there is no agent that is associated with the conversation, for instance when the consumer is returned to queue. You need to check for null value before using the `agentData` object.

Parameters:

*agentData* - contains first name, last name, avatar url and employee ID.

## Agent typing

The `onAgentTyping(boolean isTyping)` method is called when the assigned agent is typing a message. When there is 2 seconds of idle time, this method is called again to notify with `isTyping` false to indicate that the agent stopped typing.

## CSAT Screen dismissed

The `onCsatDismissed()` method is called when the feedback screen is dismissed (user clicked Submit button, user clicked Back button, etc.).

## CSAT Screen submitted

The `onCsatSubmitted(String conversationId)` method is called when the user clicked the Submit button on the feedback screen.

`conversationId` - The id of the conversation the survey is related to.

This callback comes in addition to the `onCsatDismissed` callback when clicking Submit .

## Conversation marked as urgent

The `onConversationMarkedAsUrgent()` method is called when the current conversation is marked as urgent.

## Conversation marked as normal

The `onConversationMarkedAsNormal()` method is called when the current conversation is marked as normal.

## Offline Hours Changes

The `onOfflineHoursChanges(boolean isOfflineHoursOn)` is called when there is a change in agent availability. When the agent is in off hours mode this method is called with `isOfflineHoursOn` true. When the agent return to online state, `isOfflineHoursOn` is called with `isOfflineHoursOn` false.

## LogoutLivePersonCallback

```
public interface LogoutLivePersonCallback{  
  
    void onLogoutSucceed();  
  
    void onLogoutFailed();  
  
}
```

## ICallback

```
public interface ICallback<T, E extends Throwable> {  
  
    void onSuccess(T value);  
  
    void onError(E exception);  
  
}
```

```
}
```

## InitLivePersonCallBack

```
public interface InitLivePersonCallBack {  
  
    void onInitSucceed();  
  
    void onInitFailed(Exception e);  
  
}
```

## ShutDownLivePersonCallback

```
public interface ShutDownLivePersonCallback {  
  
    void onShutdownSucceed();  
  
    void onShutdownFailed();  
  
}
```

## Configuring the SDK

The SDK allows you to configure the look and feel of the conversation screen with your branding.xml file. In order to do so, you need to create, under the **values** folder, a new resource file called branding.xml.

This file MUST contain all the resource-names as listed below. The Customer notes column includes space for you to add your own branding.

## Attributes

### Brand

| Name   | Description  | Default  |
|--|--|----------|
| <code>&lt;string name="brand_name"&gt;</code>                          | The brand name will be shown as a title on the toolbar when there is no active conversation.   | My Brand |
| <code>&lt;integer<br/>name="message_receive_icons"&gt;</code>          | For each message, there are three indicators available: Message sent, Message received, Message read. You can customize the indicators according to your needs, by using a number between 1 and 3:<br>0 - text (sent, delivered etc.) instead of icons<br>1 - Sent only<br>2 - Sent+received<br>3 - Sent+received+read |          |
| <code>&lt;string-array<br/>name="message_receive_text"&gt;</code>      | If you set 0 in the resource message_receive_icons, you can specify what texts appears for each state. You must have 4 items, in the following order:<br>1st item - message sent<br>2nd item - message delivered<br>3rd item - message read<br>4th item - message not delivered<br>5th item - message sending          |          |
| <code>&lt;bool<br/>name="clear_history_show_confirm_dialog"&gt;</code> | Define if to show confirm dialog before clearing history or not.<br>True by default.   | true     |

### Brand Message Bubble - the first brand message

| Name  | Description                       | Default |
|---|-----------------------------------|---------|
| <code>&lt;dimen<br/>name="brand_bubble_stroke_width"&gt;</code> | Int number for the outline width. | 0dp     |

|  |   |                      |
|--|---|----------------------|
| <code>&lt;color<br/>name="brand_bubble_stroke_color"&gt;</code>            | Color code for the outline color.   | #004DC9 (blue)       |
| <code>&lt;color<br/>name="brand_bubble_message_text_color"&gt;</code>      | Color code for the text of the brand bubble                                 | @android:color/white |
| <code>&lt;color<br/>name="brand_bubble_message_link_text_color"&gt;</code> | Color code for links in the text of the brand bubble.                       | @android:color/white |
| <code>&lt;color<br/>name="brand_bubble_timestamp_text_color"&gt;</code>    | Color code for the timestamp of the brand bubble.                           | #46474A (dark gray)  |
| <code>&lt;color<br/>name="brand_bubble_background_color"&gt;</code>        | Color code for the background of the brand bubble.                          | #004DC9 (blue)       |
| <code>&lt;color<br/>name="brand_logo_background_color"&gt;</code>          | Color code for the background of the default brand logo next to the bubble. | #007AFF (light blue) |

## Agent Message Bubbles

| Name  | Description                                  | Default              |
|---|--|----------------------|
| <code>&lt;dimen<br/>name="agent_bubble_stroke_width"&gt;</code>       | Int number for the outline width.            | 0dp                  |
| <code>&lt;color<br/>name="agent_bubble_stroke_color"&gt;</code>       | Color code for the outline color.            | #004DC9 (blue)       |
| <code>&lt;color<br/>name="agent_bubble_message_text_color"&gt;</code> | Color code for the text of the agent bubble. | @android:color/white |

|   |   |                        |
|---|---|------------------------|
| <code>&lt;color<br/>name="agent_bubble_message<br/>_link_text_color"&gt;</code>         | Color code for links in the text of the agent bubble.                                   | @android:color/white   |
| <code>&lt;color<br/>name="agent_bubble_timestamp_text_color"&gt;</code>                 | Color code for the timestamp of the agent bubble.                                       | #46474A<br>(dark gray) |
| <code>&lt;color<br/>name="agent_bubble_background_color"&gt;</code>                     | Color code for the background of the agent bubble.                                      | #004DC9<br>(blue)      |
| <code>&lt;color<br/>name="agent_avatar_background_color"&gt;</code>                     | Color code for the background of the agent default avatar next to the bubble            | #949596<br>(gray)      |
| <code>&lt;color<br/>name="agent_avatar_icon_color"&gt;</code>                           | Color code for the agent default icon in the avatar next to the bubble.                 | @android:color/white   |
| <code>&lt;color<br/>name="agent_bubble_link_previous_background_color"&gt;</code>       | Color code for the background of the agent bubble when url is presented                 | @android:color/white   |
| <code>&lt;color<br/>name="agent_bubble_link_previous_title_text_color"&gt;</code>       | Color code for the background of the agent title text color when url is presented       | @android:color/black   |
| <code>&lt;color<br/>name="agent_bubble_link_previous_description_text_color"&gt;</code> | Color code for the background of the agent description text color when url is presented | #555555<br>(gray)      |

## Consumer Bubbles

| Name   | Description   | Default    |
|--|---|------------|
| <code>&lt;dimen<br/>name="consumer_bubble_stroke_width"&gt;</code> | integer in dp for the bubble stroke width of the consumer bubble. | 1dp        |
| <code>&lt;color</code>   | Color code for the text of the consumer                           | @android:c |



|   |  |                        |
|---|--|------------------------|
| <code>name="consumer_bubble_message_text_color"&gt;</code>                                | bubble.  | olor/black             |
| <code>&lt;color<br/>name="consumer_bubble_message_link_text_color"&gt;</code>             | Color code for links in the text of the consumer bubble.                                   | #004DC9<br>(blue)      |
| <code>&lt;color<br/>name="consumer_bubble_timestamp_text_color"&gt;</code>                | Color code for the timestamp of the consumer bubble.                                       | #46474A<br>(dark gray) |
| <code>&lt;color<br/>name="consumer_bubble_background_color"&gt;</code>                    | Color code for the background of the consumer bubble.                                      | #EDED<br>(light gray)  |
| <code>&lt;color<br/>name="consumer_bubble_state_text_color"&gt;</code>                    | Color code for state text next to the consumer bubble.                                     | #46474A<br>(dark gray) |
| <code>&lt;color<br/>name="consumer_bubble_stroke_color"&gt;</code>                        | Color code for the stroke of the consumer bubble.  | #EDED<br>(light gray)  |
| <code>&lt;color<br/>name="consumer_bubble_link_preview_background_color"&gt;</code>       | Color code for the background of the consumer bubble when url is presented                 | @android:color/white   |
| <code>&lt;color<br/>name="consumer_bubble_link_preview_title_text_color"&gt;</code>       | Color code for the background of the consumer title text color when url is presented       | @android:color/black   |
| <code>&lt;color<br/>name="consumer_bubble_link_preview_description_text_color"&gt;</code> | Color code for the background of the consumer description text color when url is presented | #555555<br>(gray)      |

## System messages

| Name | Description | Default |
|------|-------------|---------|
|------|-------------|---------|

|   |  |                        |
|---|--|------------------------|
| <code>&lt;color<br/>name="system_bubble_text_color"&gt;</code>              | Color code for the text of the system messages.                | #46474A<br>(dark gray) |
| <code>&lt;bool<br/>name="enable_conversation_resolved_message"&gt;</code>   | Enable/disable the conversation resolved message               | true                   |
| <code>&lt;bool<br/>name="enable_conversation_resolved_separator"&gt;</code> | Enable/disable separators between conversations                | true                   |
| <code>&lt;color<br/>name="conversation_separator_text_color"&gt;</code>     | Color code for the conversation resolved message and separator | #555555<br>(gray)      |

## Unread messages indicator Bubbles

| Name   | Description   | Default               |
|--|---|-----------------------|
| <code>&lt;bool<br/>name="unread_indicator_bubble_enable"&gt;</code>            | Enable/disable the unread message indicator (shown or invisible) - true by default. | true                  |
| <code>&lt;color<br/>name="unread_indicator_bubble_text_color"&gt;</code>       | Enable/disable the unread message indicator (shown or invisible) - true by default. | #004DC9<br>(blue)     |
| <code>&lt;color<br/>name="unread_indicator_bubble_background_color"&gt;</code> | Color code for the background of the unread messages bubble.                        | #EDED<br>(light gray) |

## Survey screen

| Name | Description | Default |
|------|-------------|---------|
|------|-------------|---------|

|   |  |                        |
|---|--|------------------------|
| <code>&lt;integer<br/>name="csatSurveyExpiration<br/>InMinutes"&gt;</code>                      | Define the expiration time in minutes for the survey to appear after closing the conversation. | 1440                   |
| <code>&lt;color<br/>name="feedback_fragment_ba<br/>ckground_color"&gt;</code>                   | Feedback dialog background color.  | @android:color/white   |
| <code>&lt;color<br/>name="feedback_fragment_ti<br/>tle_question"&gt;</code>                     | Feedback dialog title color.   | @android:color/black   |
| <code>&lt;color<br/>name="feedback_fragment_st<br/>ar"&gt;</code>                               | Feedback dialog star color.  | #229A49<br>(green)     |
| <code>&lt;color<br/>name="feedback_fragment_ra<br/>te_text"&gt;</code>                          | Feedback dialog rating title color.  | #5b5c5e<br>(dark grey) |
| <code>&lt;color<br/>name="feedback_fragment_ti<br/>tle_yesno"&gt;</code>                        | Feedback dialog yes/no color.  | #5b5c5e<br>(dark grey) |
| <code>&lt;color<br/>name="feedback_fragment_ye<br/>sno_btn_selected_backgroun<br/>d"&gt;</code> | Feedback dialog yes/no selected background color.  | #229A49                |
| <code>&lt;color<br/>name="feedback_fragment_ye<br/>sno_btn_default_background<br/>&gt;</code>   | Feedback dialog yes/no default background.   | @android:color/white   |
| <code>&lt;color<br/>name="feedback_fragment_ye<br/>sno_btn_text_selected"&gt;</code>            | Feedback dialog yes/no text color when selected.   | @android:color/white   |
| <code>&lt;color<br/>name="feedback_fragment_ye</code>   | Feedback dialog yes/no text color when in default.   | #5B5C5E                |

|   |  |                      |
|---|--|----------------------|
| sno_btn_text_default">  |  |                      |
| <color<br>name="feedback_fragment_yesno_btn_stroke_default">        | Feedback dialog yes/no stroke color when in default.       | #E2E2E3              |
| <color<br>name="feedback_fragment_yesno_btn_stroke_selected">       | Feedback dialog yes/no stroke color when selected.         | #229A49              |
| <dimen<br>name="feedback_fragment_yesno_btn_stroke_width_default">  | Feedback dialog yes/no stroke width size when in default.  | 1dp                  |
| <dimen<br>name="feedback_fragment_yesno_btn_stroke_width_selected"> | Feedback dialog yes/no stroke width size when in selected. | 1dp                  |
| <color<br>name="feedback_fragment_submit_message">                  | Feedback dialog submit message text color.                 | #565656              |
| <color<br>name="feedback_fragment_submit_btn_enabled">              | Feedback dialog submit button color when enabled.          | #229A49              |
| <color<br>name="feedback_fragment_submit_btn_text_enabled">         | Feedback dialog submit button text color when enabled.     | @android:color/white |
| <color<br>name="feedback_fragment_submit_btn_disabled">             | Feedback dialog submit button color when disabled.         | @android:color/white |
| <color<br>name="feedback_fragment_submit_btn_text_disabled">        | Feedback dialog submit button text color when disabled.    | #BDBDBD              |
| <color<br>name="feedback_fragment_submit_btn_stroke_enabled">       | Feedback dialog submit button stroke color when enabled.   | #229A49              |

|  |   |                          |
|--|---|--------------------------|
| <code>bmit_btn_stroke_enabled"&gt;</code>  |   |                          |
| <code>&lt;color<br/>name="feedback_fragment_su<br/>bmit_btn_stroke_disabled"&gt;</code>            | Feedback dialog submit button stroke color when disabled.   | #E2E2E3                  |
| <code>&lt;dimen<br/>name="feedback_fragment_su<br/>bmit_btn_stroke_width_enab<br/>led"&gt;</code>  | Feedback dialog submit button stroke width size when enabled.   | 1dp                      |
| <code>&lt;dimen<br/>name="feedback_fragment_su<br/>bmit_btn_stroke_width_disa<br/>bled"&gt;</code> | Feedback dialog submit button stroke width size when disabled.  | 1dp                      |
| <code>&lt;color<br/>name="feedback_fragment_ag<br/>ent_details_name"&gt;</code>                    | Define the color of the agent name on agent details section in feedback dialog.<br>Visible only if <i>show_agent_details_csat</i> is true.  | @android:c<br>olor/black |
| <code>&lt;bool<br/>name="show_feedback"&gt;</code>   | Defines whether to show the feedback dialog.  | true                     |
| <code>&lt;bool<br/>name="show_agent_details_c<br/>sat"&gt;</code>                                  | Define if the agent's name and avatar are visible on top of feedback dialog.<br><br>(true=show, false=hide)<br><br>NOTE: if both <i>show_yes_no_question</i> and <i>show_agent_details_csat</i> are set to true, <i>show_yes_no_question</i> <b>will be ignored and will not be visible</b> . | true                     |
| <code>&lt;bool<br/>name="show_yes_no_questio<br/>n"&gt;</code>                                     | Defines whether to show or hide the yes/no question in the feedback dialog (true=show, false=hide)<br><br>NOTE: if both <i>show_yes_no_question</i> and <i>show_agent_details_csat</i> are set to true, <i>show_yes_no_question</i> <b>will be ignored and will not be visible</b> .          | true                     |

|   |  |      |
|---|--|------|
| <pre>&lt;bool name="show_csat_thank_you" &gt;</pre> | Define if “thank you” screen will appear after submitting the survey.<br>(true=show, false=hide) | true |
|---|--|------|

## Message Edit Text

| Name  | Description   | Default                    |
|---|---|----------------------------|
| <pre>&lt;color name="edit_text_underline_color"&gt;</pre>   | Color code for the Enter Message control underline color. | #90CAF9                    |
| <pre>&lt;color name="lp_enter_msg_text"&gt;</pre>           | Define the input message text color.                      | @android:color/black       |
| <pre>&lt;color name="lp_enter_msg_hint"&gt;</pre>           | Define the input message hint color.                      | @android:color/darker_gray |
| <pre>&lt;color name="lp_send_button_text_enable"&gt;</pre>  | Define the color of the send button when it's enabled.    | #004DC9<br>(blue)          |
| <pre>&lt;color name="lp_send_button_text_disable"&gt;</pre> | Define the color of the send button when it's disabled.   | #B7B8B9                    |
| <pre>&lt;bool name="use_send_image_button"&gt;</pre>        | Use an icon for the send button instead of “Send” text    | false                      |

## Connection status bar

| Name | Description | Default |
|------|-------------|---------|
|------|-------------|---------|

|  |  |                      |
|--|--|----------------------|
| <code>&lt;color<br/>name="connection_status_connecting_bg_color"&gt;</code>      | Define the color of statusbar background color while trying to connect.        | #F2F5F5F5            |
| <code>&lt;color<br/>name="connection_status_not_connected_bg_color"&gt;</code>   | Define the color of statusbar background color when connection is unavailable. | #CC000000            |
| <code>&lt;color<br/>name="connection_status_connecting_text_color"&gt;</code>    | Define the color of statusbar text color while trying to connect.              | #46474A              |
| <code>&lt;color<br/>name="connection_status_not_connected_text_color"&gt;</code> | Define the color of statusbar text color when connection is unavailable.       | @android:color/black |

## In page navigation - Scroll down indicator

| Name  | Description  | Default              |
|---|--|----------------------|
| <code>&lt;bool<br/>name="scroll_down_indicator_enabled"&gt;</code>                    | Enable/disable the scroll down indicator (shown or invisible). True by default   | true                 |
| <code>&lt;bool<br/>name="scroll_down_indicator_unread_summary_enabled"&gt;</code>     | Enable/disable the summary in scroll down indicator (shown or invisible).<br><br>If <i>unread_indicator_bubble_enable</i> is false, it will be in minimized mode without a badge indicating number of unread message. And tap will scroll to the last message. | true                 |
| <code>&lt;color<br/>name="scroll_down_indicator_unread_counter_text_color"&gt;</code> | Define the color of the unread messages counter text color.  | @android:color/white |
| <code>&lt;color<br/>name="scroll_down_indicator_unread_summary_text_color"&gt;</code> | Define the color of the unread message summary (preview) text color.   | @android:color/white |

|   |   |                          |
|---|---|--------------------------|
| <code>r"&gt;</code>   |   |                          |
| <code>&lt;color<br/>name="scroll_down_indicato<br/>r_unread_counter_stroke_co<br/>lor"&gt;</code> | Define the color of the unread messages counter stroke color. | #CC000000                |
| <code>&lt;dimen<br/>name="scroll_down_indicato<br/>r_unread_counter_stroke_wi<br/>dth"&gt;</code> | Define the dimen of the unread messages counter stroke width. | 1dp                      |
| <code>&lt;color<br/>name="scroll_down_indicato<br/>r_unread_counter_solid_col<br/>or"&gt;</code>  | Define the color of the unread messages counter solid color.  | #FF0000<br>(red)         |
| <code>&lt;color<br/>name="scroll_down_indicato<br/>r_background_color"&gt;</code>                 | Define the color of the scroll down background color.         | #CC000000                |
| <code>&lt;color<br/>name="scroll_down_indicato<br/>r_arrow_down_color"&gt;</code>                 | Define the color of the image arrow scrolling down.           | @android:c<br>olor/white |

## Photo Sharing

| Name   | Description  | Default |
|--|--|---------|
| <code>&lt;bool<br/>name="enable_photo_sharing<br "=""/>&gt;</code>       | Enable/disable the photo sharing feature.                    | false   |
| <code>&lt;integer<br/>name="max_number_stored_im<br/>ages"&gt;</code>    | Define the max number of images that will be stored locally. | 20      |
| <code>&lt;integer<br/>name="full_image_compressi<br/>on_rate"&gt;</code> | Define the image compression rate (percentage)               | 50      |



|   |  |                         |
|---|--|-------------------------|
| <code>&lt;integer<br/>name="thumbnail_longer_dimension_resize"&gt;</code>   | Define the size of the thumbnail image longer dimension after resizing it (pixels) | 100                     |
| <code>&lt;integer<br/>name="full_image_longer_dimension_resize"&gt;</code>  | Define the size of the full image longer dimension after resizing it (pixels).     | 800                     |
| <code>&lt;integer<br/>name="max_image_size_kb"&gt;</code>                   | Define the maximum image size in KB.   | 3000                    |
| <code>&lt;color<br/>name="attachment_menu_item_background_color"&gt;</code> | Define the background color of the items in the attachment menu.                   | #004DC9<br>(blue)       |
| <code>&lt;color<br/>name="lp_attachment_menu_background_color"&gt;</code>   | Define the background color of the attachment menu                                 | #F5F5F5<br>(light gray) |
| <code>&lt;color<br/>name="lp_attachment_menu_item_text_color"&gt;</code>    | Define the items' text color in the attachment menu                                | #46474A<br>(gray)       |
| <code>&lt;color<br/>name="lp_attachment_menu_item_icon_color"&gt;</code>    | Define the items' icon color in the attachment menu                                | #F5F5F5<br>(light gray) |

## General Style

| Name  | Description  | Default              |
|---|--|----------------------|
| <code>&lt;color<br/>name="conversation_background"&gt;</code>     | Define the color code for the entire view background.<br>In activity mode - Also the color of android:windowBackground | @android:color/white |
| <code>&lt;bool<br/>name="link_preview_use_big_picture"&gt;</code> | Define which configuration to show when sending / receiving s link (big / small picture)                               | true                 |

|  |  |   |
|--|--|---|
| <pre>&lt;bool name="link_preview_enable_real_time_preview" "&gt;</pre> | Define whether or not we should show a real time link preview. A preview while the consumer is typing an url | true  |
| <pre>&lt;bool name="link_preview_to_use_more_than_og_tags" "&gt;</pre> | parse only <og:> tags or others as well  | false - use <og:title> tags only.<br><br>true - use <og:title> and <title> tags |

### Conversation Activity Style - (activity mode only!)

| Name   | Description                                    | Default                  |
|--|--|--------------------------|
| <pre>&lt;color name="lp_colorPrimary"&gt;</pre>      | Define the primary color of the activity.      | android:colorPrimary     |
| <pre>&lt;color name="lp_colorPrimaryDark" &gt;</pre> | Define the primary dark color of the activity. | android:colorPrimaryDark |

### Accessibility

| Name  | Description  | Default |
|---|--|---------|
| <pre>&lt;integer name="snackbar_duration_for_accessibility"&gt;</pre> | Number of milliseconds to show the TTR snackbar if the accessibility TalkBack option is on | 60,000  |

## Miscellaneous

| Name   | Description   | Default                   |
|--|---|---------------------------|
| <code>&lt;bool<br/>name="disableTTRPopup"&gt;</code>                   | Defines whether to disable the TTR snackbar popup (true=disable) false by default.  | false                     |
| <code>&lt;bool<br/>name="vibrate_enabled"&gt;</code>                   | Enable/Disable vibrate upon receiving messages from agent while conversation screen is in foreground. false by default.   | false                     |
| <code>&lt;bool<br/>name="contextual_menu_on_toolbar"&gt;</code>        | Enable multiple message copy menu over the app toolbar.<br>If true, when long pressing a message on chat it will select the message and enable a context menu over the toolbar, enabling the user to copy multiple messages.<br>If false, long pressing a message will display a copy popup menu. | true                      |
| <code>&lt;color<br/>name="bubble_selected_background_color"&gt;</code> | Define the background color of item when it's selected to be copied (if multiple message copy is enabled).  | #5597a7e3                 |
| <code>&lt;integer<br/>name="encryptionVersion"&gt;</code>              | Defines the encryption version to use.<br>Currently available version 1 only.<br>1 - encrypt data<br>0 - disable encryption   | 1                         |
| <code>&lt;string name="csds_url"&gt;</code>                            | For vanity URL purposes.<br>For regular use please use:<br><b>adminlogin.liveperson.net</b>   | adminlogin.liveperson.net |
| <code>&lt;integer<br/>name="idp_num_history_conversation"&gt;</code>   | When user is authenticated, this indicates the number of recent conversations to reload from the server (including their messages) when running for the first time.   | 2                         |

|  |   |          |
|--|---|----------|
| <code>&lt;bool<br/>name="show_timestamp_in_ttr_notification"&gt;</code>                  | When true the TTR snackbar will display the time until the agent responds.<br>If set to false, a general message is displayed.  | true     |
| <code>&lt;integer<br/>name="ttr_duration"&gt;</code>                                     | Set the duration that the TTR snackbar will be visible (ms).  | 3,000    |
| <code>&lt;bool<br/>name="send_agent_profile_updates_when_conversation_closed"&gt;</code> | When true the callback <a href="#"><i>LivePersonCallback#onAgentDetailsChanged</i></a> will be called with the agent details updates even if the last conversation is closed (in this case it will provide the assigned agent of the last conversation). If false, this callback will be called only when the current conversation is active. | true     |
| <code>&lt;bool<br/>name="ttr_message_off_hours_enabled"&gt;</code>                       | Defines whether to show the off hours snackbar popup (true=enable).   | true     |
| <code>&lt;integer<br/>name="ttrShowFrequencyInSeconds"&gt;</code>                        | Define the frequency of the TTR (time to response) messages.  | 8        |
| <code>&lt;bool<br/>name="enable_client_only_masking"&gt;</code>                          | Defines whether to enable or disable client side only masking.<br>False by default.   | false    |
| <code>&lt;bool<br/>name="enable_real_time_masking"&gt;</code>                            | Defines whether to enable or disable real time masking.<br>False by default.  | false    |
| <code>&lt;string<br/>name="client_only_masking_regex"&gt;</code>                         | Defines the java regex for client side only masking.<br>By default does not contain any value.  | No value |
| <code>&lt;string<br/>name="client_only_mask_character"&gt;</code>                        | The character used to mask client only string.  | '*'      |

|   |  |          |
|---|--|----------|
| <code>&lt;string<br/>name="real_time_masking_regex"&gt;</code>      | Defines the Java regex for real time masking.  | No value |
| <code>&lt;string<br/>name="real_time_mask_character"&gt;</code>     | The character used to mask the real time message.  | '**'     |
| <code>&lt;string<br/>name="lp_bubble_phone_links_regex"&gt;</code>  | Defines the java regex for phone links in bubble messages.<br>By default does not contain any value. | No value |
| <code>&lt;string<br/>name="lp_bubble_url_links_regex"&gt;</code>    | Defines the java regex for url links in bubble messages.<br>By default does not contain any value.   | No value |
| <code>&lt;string<br/>name="lp_bubble_email_links_regex"&gt;</code>  | Defines the java regex for email links in bubble messages.<br>By default does not contain any value. | No value |
| <code>&lt;string<br/>name="lp_date_format"&gt;</code>               | Define date format. More info <a href="#">here</a> .   | No value |
| <code>&lt;string<br/>name="lp_time_format"&gt;</code>               | Define time format . More info <a href="#">here</a> .  | No value |
| <code>&lt;string<br/>name="lp_date_time_format"&gt;</code>          | Define date-time format. More info <a href="#">here</a> .  | No value |
| <code>&lt;integer<br/>name="sendMessageTimeoutInMinutes"&gt;</code> | Define timeout for automatic resending pending message before moving it to failed.                   | 60       |

## Deprecated Attributes

| Name   | Description                         |
|--|-------------------------------------|
| <code>&lt;string name="custom_button_icon_name"&gt;</code> | Custom button icon filename without |

|   |  |
|---|--|
|   | extension. This will be displayed on the toolbar.  |
| <pre>&lt;string name="custom_button_icon_description"&gt;</pre> | <p>Content description for custom button.</p> <p>It briefly describes the view and is primarily used for accessibility support. Set this property to enable better accessibility support for your application.</p> |
| <pre>&lt;string name="notification_large_icon_name"&gt;</pre>   | <p>The name of a resource to use as the large icon of the push notification</p>  |

## Configuring the message's EditText

There is an option to change the whole style of the message EditText. In the app's styles.xml file, override the lp\_enter\_message\_style with the required style.

Example:

```
<style name="lp_enter_message_style" parent="Theme.AppCompat.Light.NoActionBar">
<item name="colorControlActivated">#F8BBD0</item>
```

...

```
</style>
```

## ProGuard Configuration (Android only)

The SDK handles its own obfuscation and all its dependencies according to ProGuard rules. There is no need to add any ProGuard specific rules that relate to the SDK.

The SDK ProGuard will run automatically when the ProGuard option is enabled in the gradle file of your application.

In case there is no ProGuard activated, the SDK ProGuard will also be disabled.

# String localization in SDK

## Modifying String

You may change every string appearing on the SDK interface by overriding the respective string key.

### General

| String name                              | Used in   | Default value   |
|--|---|---|
| lp_enter_message                         | Enter message text box when empty.  | Write a message   |
| lp_send                                  | The “Send” button text.   | Send  |
| lp_no_network_toast_message              | A toast message when there is no network.   | No internet connection. Please check your connection and try again. |
| lp_no_action_not_available_toast_message | A toast message when the required action is not available (e.g. Mark as urgent when there is no active conversation). | Action not available - no open conversation                         |
| lp_today                                 | Today header in conversation.   | Today   |
| lp_yesterday                             | Yesterday header in conversation.   | Yesterday   |
| lp_first_message                         | System message before the first conversation.   | How can I help you today?   |
| lp_loading_message                       | Text above the loading icon when loading previous   | Loading...  |

|  |   |   |
|--|---|---|
|  | messages.   |   |
| lp_conversation_ended_by_agent_with_name | <p>Message when the conversation was resolved when we have an agent name.</p> <p>%1\$s - agent name</p> <p>%2\$s - time</p> | Conversation resolved by %1\$s \n %2\$s |
| lp_conversation_ended_by_agent_no_name   | <p>Message when the conversation was resolved when we don't have the agent name.</p> <p>%1\$s - time</p>                    | Conversation resolved by Agent \n %1\$s |
| lp_conversation_ended_by_you             | <p>Message when the conversation was resolved by the client.</p> <p>%1\$s - time</p>  | Conversation resolved by You \n %1\$s   |
| lp_is_typing                             | Text in conversation activity when agent is typing.   | typing...                               |
| lp_mark_as_urgent_menu_text              | "Mark as urgent" string in menu and snack bar.  | Mark as urgent                          |
| lp_mark_as_resolved_menu_text            | "Mark as resolved" string in menu.  | Mark as resolved                        |
| lp_clear_history_menu_text               | "Clear history" string in menu  | Clear history                           |
| lp_dismiss_as_urgent_menu_text           | Dismiss urgent menu text.   | Dismiss urgent                          |
|  | "Clear history" string in   | Clear history                           |



|                                    |  |  |
|------------------------------------|--|--|
|                                    | menu.  |  |
|                                    | “Clear history” confirmation dialog text   | All of your existing conversation history will be lost. Are you sure?        |
|                                    | “Clear” button text on “Clear history” dialog.   | Clear  |
| lp_end_conversation_first          | Dialog text that is shown in case trying to clear history when a conversation is open. | Please resolve the conversation first.                                       |
| lp_dismiss_as_urgent_two_lines     | “Dismiss urgent” string in menu and snack bar.   | Dismiss urgent   |
| lp_mark_as_urgent_dialog_header    | Mark as urgent confirmation dialog header.   | Are you sure you want to mark this conversation as urgent?                   |
| lp_dismiss_urgent_dialog_header    | Dismiss urgent confirmation dialog header.   | Are you sure you want to mark this conversation as not urgent?               |
| lp_mark_as_resolved_dialog_message | Resolve conversation confirmation dialog text.   | Are you sure this topic is resolved?   |
| lp_mark_as_urgent_dialog_message   | Mark as urgent confirmation dialog text.   | This means that your conversation will get top priority.                     |
| lp_dismiss_urgent_dialog_message   | Dismiss urgent confirmation dialog text.   | This means that your conversation will get normal priority.                  |
| lp_ttr_message_with_timestamp      | Text in TTR snackbar when timestamp is shown.  | An agent will respond within the next %1\$s (please do not remove the %1\$s) |

|                        |   |               |
|------------------------|---|---------------|
| lp_ttr_message_minutes | (plurals string that contains: “one” and “others”)<br><br>The <i>one</i> or <i>others</i> strings is concatenated to the <i>lp_ttr_message_with_timest amp</i> string above according to whether it’s single <b>minute</b> multiple <b>minutes</b> .<br><br><a href="#">Example</a> |               |
|                        | one   | %1\$s minute  |
|                        | others  | %1\$s minutes |
| lp_ttr_message_hours   | (plurals string that contains: “one” and “others”).<br><br>The <i>one</i> or <i>others</i> strings is concatenated to the <i>lp_ttr_message_with_timest amp</i> string above according to whether it’s single <b>hour</b> multiple <b>hours</b> .<br><br><a href="#">Example</a>    |               |
|                        | one   | %1\$s hour    |
|                        | others  | %1\$s hours   |
| lp_ttr_message_days    | (plurals string that contains: “one” and “others”)<br><br>The <i>one</i> or <i>others</i> strings is concatenated to the <i>lp_ttr_message_with_timest amp</i> string above according to whether it’s single <b>day</b> multiple <b>days</b> .                                      |               |

|                             |  |  |
|-----------------------------|--|--|
|                             | <a href="#">Example</a>  |  |
|                             | one  | %1\$s day                                  |
|                             | others   | %1\$s days                                 |
| lp_ttr_message_no_timestamp | Text in TTR snackbar when timestamp is not shown.                      | An agent will respond shortly              |
| lp_feedback_1               | String displayed when one star is selected in the feedback dialog.     | Very Dissatisfied                          |
| lp_feedback_2               | String displayed when two stars are selected in the feedback dialog.   | Dissatisfied                               |
| lp_feedback_3               | String displayed when three stars are selected in the feedback dialog. | Neither                                    |
| lp_feedback_4               | String displayed when four stars are selected in the feedback dialog.  | Satisfied                                  |
| lp_feedback_5               | String displayed when five stars are selected in the feedback dialog.  | Very Satisfied                             |
| lp_feedback_thank_you       | Text displayed after the feedback dialog is submitted.                 | Survey submitted successfully.\nThank you! |
| lp_feedback_submit          | The feedback submit button text.                                       | Submit                                     |
| lp_feedback_yesno_question  | Yes/No question text in feedback dialog.                               | Did we solve your issue today?             |

|                                      |   |   |
|--------------------------------------|---|---|
| lp_feedback_submit_message           | Submit message text at the bottom of feedback dialog.               | Your feedback helps us serve you better.\n It will not be shared with any customer service representatives. |
| lp_feedback_yesno_negative_title     | Negative button text in the feedback dialog.                        | NO  |
| lp_feedback_yesno_positive_title     | Positive button text in the feedback dialog.                        | YES   |
| lp_feedback_question                 | Feedback dialog rate question text.                                 | How would you rate your connection with our agent?  |
| lp_end                               | End conversation “End” button text.                                 | End   |
| lp_skip                              | Feedback dialog toolbar skip button text.                           | Skip  |
| lp_done                              | Feedback dialog toolbar done button text (after submitting).        | Done  |
| lp_ok                                | Confirmation dialog OK button.                                      | OK  |
| lp_cancel                            | Confirmation dialog Cancel button.                                  | Cancel  |
| lp_menu_copy                         | Copy menu button text when selecting messages in conversation.      | Copy  |
| lp_end_conversation                  | End conversation title.   | Resolve the conversation  |
| lp_resend_failed_conversation_closed | Toast message displayed when trying to resend a failed message when | This conversation has already been resolved.  |

|                                       |   |   |
|---------------------------------------|---|---|
|                                       | conversation is already closed.   |   |
| lp_resend_failed_masked_message       | Toast message displayed when trying to resend a failed masked message.  | Message failed to send. Please re-enter message and send again. |
| lp_new_messages                       | Notification message displayed when there are multiple push messages.   | new messages  |
| lp_message_time_now                   | Message timestamp for the latest messages ("Now").  | Now   |
| lp_message_time_now_with_state        | Message timestamp for the latest messages that has a sending state ("now").   | Now   |
| lp_message_time_min_ago               | Message timestamp for older messages ("5 min ago").   | Min ago   |
| lp_ttr_message_off_hours_time_zone_id | Represents Java timezone ID that is used in the off hours message.<br><br>For a full list of the available IDs, use the "Aliases" from <a href="#">here</a> . | US/Pacific  |
| lp_ttr_message_off_hours_message      | Message to show when the online hours is more than 2 days from now.<br><br>includes 1 param:<br><br>%1\$s - for the full date (MMM dd, yyyy hh:mm a)          | Thanks for your message. We will be back online at %1\$s        |

|   |  |   |
|---|--|---|
| lp_unread_message                         | (plurals string that contains: “one” and “others”).<br><br>Used in the unread messages indicator to indicate how many unread messages<br><br><a href="#">Example</a> |   |
|   | one  | %1\$d UNREAD MESSAGE  |
|   | others   | %1\$d UNREAD MESSAGES   |
| lp_still_loading_message                  | Message displayed when loading conversation takes longer the usual   | Still loading conversation...                                     |
| lp_date_time_format                       | Date and time format to be used instead of the standard format   | No Value  |
| lp_failed_upload_toast_message            | Toast message displayed when uploading a photo failed  | Failed to upload file   |
| lp_failed_download_toast_message          | Toast message displayed when downloading a photo failed  | Failed to download file   |
| brand_name                                | The default agent name displayed on the toolbar  | My Brand  |
| lp_ttr_message_off_hours_message_today    | A snackbar content when the agent is in off hours and TTR is sometime today  | Thanks for your message. We will be back online today at %1\$s    |
| lp_ttr_message_off_hours_message_tomorrow | A snackbar content when the agent is in off hours and TTR is sometime tomorrow   | Thanks for your message. We will be back online tomorrow at %1\$s |

|  |  |  |
|--|--|--|
| lp_add_a_caption                       | Hint text in the Enter Message EditText on the image preview screen    | Add a caption                                |
| lp_connection_status_connecting        | Connection bar text when connecting                                    | Connecting...                                |
| lp_connection_status_trying_to_connect | Connection bar text when the connection is longer than 5 seconds       | Trying to connect...                         |
| lp_connection_status_failed_to_connect | Connection bar text when could not connect to the messaging server     | Failed to connect to the server.             |
| lp_connection_status_no_connection     | Connection bar text when there is no internet connection on the device | No connection. Please check your connection. |
| lp_attachment_menu_gallery_item_text   | The Gallery icon's text in the attachment menu                         | GALLERY                                      |
| lp_attachment_menu_camera_item_text    | The Camera icon's text in the attachment menu                          | CAMERA                                       |

### Clear History dialog

| String name                      | Used in  | Default value   |
|----------------------------------|--|---|
| lp_clear_history_dialog_title    | Title of the Clear History confirm dialog        | Clear history   |
| lp_clear_history_dialog_message  | Body message of the Clear History confirm dialog | All of your existing conversation history will be lost. Are you sure? |
| lp_clear_history_dialog_positive | Positive button text                             | Clear   |

|                           |  |  |
|---------------------------|--|--|
| e_button                  |  |  |
| lp_end_conversation_first | Message text displayed when trying to clear history and the conversation is not resolved | Please resolve the conversation first. |

## Masking

| String name                          | Used in  | Default value  |
|--------------------------------------|--|--|
| lp_system_message_real_time_masked   | Text of system message, added after detecting a real time masked message (if this feature is enabled).   | Your personal data has been masked to protect your security and cannot be read by the agent. |
| real_time_mask_character             | The character used to mask the real time message.  | *  |
| lp_system_message_client_only_masked | Text of system message, added after detecting a client only masked message (if this feature is enabled). | Your personal data has been masked to protect your security. Only the agent can read it.     |
| client_only_mask_character           | The character used to mask client only string.   | *  |

## Accessibility strings (used by the Accessibility TalkBack)

| String name               | Used in                 | Default value |
|---------------------------|-------------------------|---------------|
| lp_accessibility_received | Received message status | received      |



|   |   |                                  |
|---|---|----------------------------------|
| lp_accessibility_selected                         | Used to indicate the selected star on the feedback screen | selected                         |
| lp_accessibility_agent                            | Used as a message prefix on the message from the agent    | Agent                            |
| lp_accessibility_you                              | Used as a message prefix on the message from the consumer | You                              |
| lp_accessibility_attachment_menu_button_collapsed | The attachment menu button name when collapsed            | Attachment menu button collapsed |
| lp_accessibility_attachment_menu_button_expanded  | The attachment menu button name when expanded             | Attachment menu button expanded  |
| lp_accessibility_photo_preview                    | Used on the image on the preview screen                   | Photo preview                    |
| lp_accessibility_attachment_menu                  | Used on the attachment menu                               | Attachment menu                  |
| lp_accessibility_gallery                          | Used on the gallery button (on the attachment menu)       | Gallery                          |
| lp_accessibility_camera                           | Used on the camera button (on the attachment menu)        | Camera                           |
| lp_accessibility_image                            | Used on the thumbnail image on the conversation screen    | Image                            |
| lp_accessibility_full_image                       | Used on the image in the full image screen                | Full image                       |
| lp_accessibility_resend                           | Used on the resend button                                 | Resend                           |

|                                     |   |                     |
|-------------------------------------|---|---------------------|
| lp_accessibility_agnet_icon         | Used on the agent avatar  | Agent Icon          |
| lp_accessibility_chat_message       | Used as a label on the 'Enter message' EditText control                   | Chat message        |
| lp_accessibility_image_caption      | A label string for the Enter Message EditText in the image preview screen | Image caption       |
| lp_accessibility_photo              | Used on an image on the conversation screen                               | Photo               |
| lp_accessibility_new_agent_message  | Used when receive an incoming message from agent                          | New agent message:  |
| lp_accessibility_new_system_message | Used when receive an incoming system message                              | New system message: |

## Modifying resources

The SDK utilizes several resources as part of its GUI. To customize those resources, please add appropriate resources to your project:

| Description   | Resources name             | Size |
|---|----------------------------|------|
| <p>Default brand avatar on the avatar next to brand bubble (the first brand message) and on agent avatar appearing on the action bar before an agent is assigned.</p> <p>In case you want to define the background color for this avatar - override "brand_logo_background_color" resource id. (This is</p> | lp_messaging_ui_brand_logo |      |

|   |                                 |  |
|---|---------------------------------|--|
| relevant for bubble brand's avatar only. Background color of agent avatar on action bar is "agent_avatar_background_color").  |                                 |  |
| <p>Default agent avatar appearing next to an agent's bubble when no avatar URL is assigned on LiveEngage and on agent avatar appearing on the action bar.</p> <p>In case you want to define the background color for this avatar, override "agent_avatar_background_color" resource id.</p> | lp_messaging_ui_ic_agent_avatar |  |

## Plural String Resource Example

Following is an example on how to add a plural string resource:

```
<plurals name="lp_ttr_message_hours">
    <item quantity="one">" %1$s hour"</item>
    <item quantity="other">" %1$s hours"</item>
</plurals>
```

## Timestamps Formatting

Android provides 4 different default types of date and time formats:

**SHORT** is completely numeric (12.13.52 or 3:30pm)

**MEDIUM** is longer and contains the first 3 letters of the month (Jan 12, 1952)

**LONG** is longer (January 12, 1952 or 3:30:32pm)

**FULL** specifies the complete time and date (Tuesday, April 12, 1952 AD or 3:30:42pm) PST.

For each feature we added a special resource ID in case customizing the date/time formatting is needed. By default, all these formatting resources are empty in order to take the default device locale.

We define 3 configurable formatting resources:

- For date only (separator):

```
<string name="lp_date_format"></string>
```

- For time only (bubble's timestamp & off hours time in case of today/tomorrow):

```
<string name="lp_time_format"></string>
```

- For date & time together (resolve message & off hours time in case of other date):

```
<string name="lp_date_time_format"></string>
```

## Off Hours

### Date & Time

Today and tomorrow off hours message use default **SHORT** time without date according to the locale (default or custom) and to device setting.

If device is set to 12 hours format :

“Thanks for your message. We will be back online today/tomorrow at 3:30pm”

If device is set to 24 hours format :

“Thanks for your message. We will be back online today/tomorrow at 15:30”

In case you want special **hour** format, you can use

```
<string name="lp_time_format"></string>
```

With any **time** format. For ex. - “hh:mm a”, “HH:mm” etc..

**Date off hours message** (not today/tomorrow) use default **LONG** date and **SHORT** time according to the locale (default or custom) and to device setting.

If device is set to 12 hours format :

“Thanks for your message. We will be back online January 12, 2017 at 3:30pm”

If device is set to 24 hours format :

“Thanks for your message. We will be back online *January 12, 2017 at 15:30*”

In case you want special **date/hour** format, you can use

```
<string name="lp_date_time_format"></string>
```

With any **date & time** format. For ex. - “MMM d, yyyy hh:mm a”, “EEEE dd/mm/yy HH:mm” etc..

## Timezone

Off hours can appear in different time zone with this resource ID:

```
<string name="lp_ttr_message_off_hours_time_zone_id"></string>
```

Can find list of timezones id [here](#)

For ex. - “US/Pacific”, “Europe/Berlin”.

## Bubble timestamp

Bubbles contains only time in **SHORT time** format, according to the locale (default or custom) and to device setting.

If device is set to 12 hours format : “3:30pm”

If device is set to 24 hours format : “15:30”

If you wish to configure this time format, override this resource ID:

```
<string name="lp_time_format"></string>
```

With any **time** format. For ex. - “hh:mm a”, “HH:mm” etc..

This will apply to all bubble’s timestamp.

## Separator timestamp

Separator contains only date in **SHORT date** format, according to the locale (default or custom) and to device setting.

“9/25/16” for US locale / “2016/9/25” for JP locale

If you wish to configure this time format - override this resource id :

```
<string name="lp_date_format"></string>
```

With any **date** format. For ex. - “MMM d, yyyy”, “EEEE dd/mm/yy” etc.

## Resolve message

Resolve message use default **SHORT date** and **SHORT time** according to the locale (default or custom) and to device setting.

If device is set to 12 hours format (US locale):

“Conversation resolved by [agent name] \n 9/25/16 , 3:30pm”

If device is set to 24 hours format (US locale):

“Conversation resolved by [agent name] \n 9/25/16 , 15:30”

In case you want special **date/hour** format, you can use

```
<string name="lp_date_time_format"></string>
```

With any **date & time** format. For ex. - “MMM d, yyyy hh:mm a”, “EEEE dd/mm/yy HH:mm” etc..

## CSAT Behavior

### Overview

This document describes the CSAT behaviour and configurations in the Messaging SDK.

You can find all the related configurations in the resources ID table, under [Survey Screen](#).

### Show CSAT flow

Show if:

1. CSAT configured to appear according to <bool name="show\_feedback">
2. Conversation has an assigned agent.
3. Conversation’s CSAT wasn’t previously submitted.

## Dismiss CSAT

The CSAT view is dismissed in one of four cases:

1. User pressed the submit button (answers are sent to the survey).
2. User choose to skip the CSAT (skipped button is pressed).
3. The CSAT is automatically dismissed if it was filled in any other device.
4. If agent resumed the conversation while csat is visible - it will automatically dismissed.

## CSAT UI content

CSAT screen includes several content containers:

### agentView (avatar and agent name)

1. Could be hidden or not according to `<bool name="show_agent_details_csat">`
2. Contains agent avatar:
  - a. If conversation has assigned agent and its image was downloaded previously using `profileUrl`, this image will be presented in the view.
  - b. If no image available, default avatar is presented. It's background and tint color is according to agent bubble with `lp_messaging_ui_ic_agent_avatar` and `agent_avatar_background_color`. More info in ['Configuring the SDK'](#)
3. Contains agent name:
  - a. By default it's an empty label.
  - b. If conversation has assigned agent, the agent's `nickName` will be used.

### ratingQuestionView (stars)

1. Always visible - can't configure its visibility.
2. Stars color is defined by `<color name="feedback_fragment_star">`
3. Rating question includes 'Agent' by default in the text. If conversation has assigned agent and the agent's `nickName` is not empty, this `nickName` will be used instead.

### resolutionConfirmationView (yes/no)

1. Could be hidden or not according to `<bool name="show_yes_no_question">`
2. If `agentView` is shown (`"show_agent_details_csat"`), this view will be always hidden (even if `"show_yes_no_question"` is set to true)
3. The question text color is defined with `<color name="feedback_fragment_title_yesno">` all the configuration related to Yes/No buttons explained in [Survey Screen](#) resources table and starting with the prefix : `"feedback_fragment_yesno_btn_"`.

# Photo Sharing - Beta

## Overview

This section describes the photo sharing behaviour and configurations in the Messaging SDK.

You can find all the related configurations in the resources ID table, under [Photo Sharing](#).

Notes:

- *This feature is available only for the In-App Messaging SDK.*
- *This feature enables photo sharing only (not video/files).*
- *Photo-sharing is one-way only: Photos can be sent from consumer to agent, but not vice versa.*
- *Device storage includes up to 20 images - this is configurable.*
- *Supported formats: .png, .jpg, .gif (non-animated).*
- *Photo size reduction: Thumbnail - 30 KB, Preview -3 MB.*
- The SDK version contains a beta version of the Photo sharing feature. For now the SDK doesn't support continuous uploading photos outside the conversation screen. Full-blown solution is under construction.

## Enable Photo Sharing

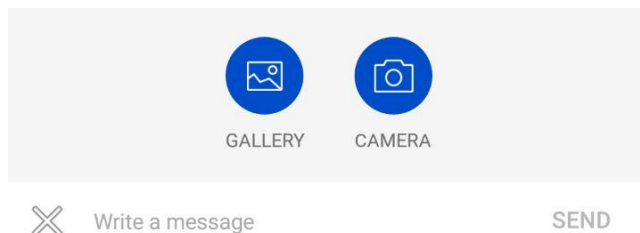
To enable/disable photo sharing you can change the boolean value `<bool name="enable_photo_sharing">` By default this value is set to false.

## Upload Photo

To upload a photo, press on the “attach” button next to “enter message” edit text



A menu will open with 2 options: Gallery and Camera. If the user had set a default app for any of those action- it will be open by default. Otherwise Android OS will open a popup menu with all the available apps for the relevant category (gallery or camera).



Changing the background color of attachment menu is available with configuration :

`<color name="attachment_menu_item_background_color">`

Changing the text of Gallery/Camera:



```
<string name="lp_accessibility_gallery">
<string name="lp_accessibility_camera">
```

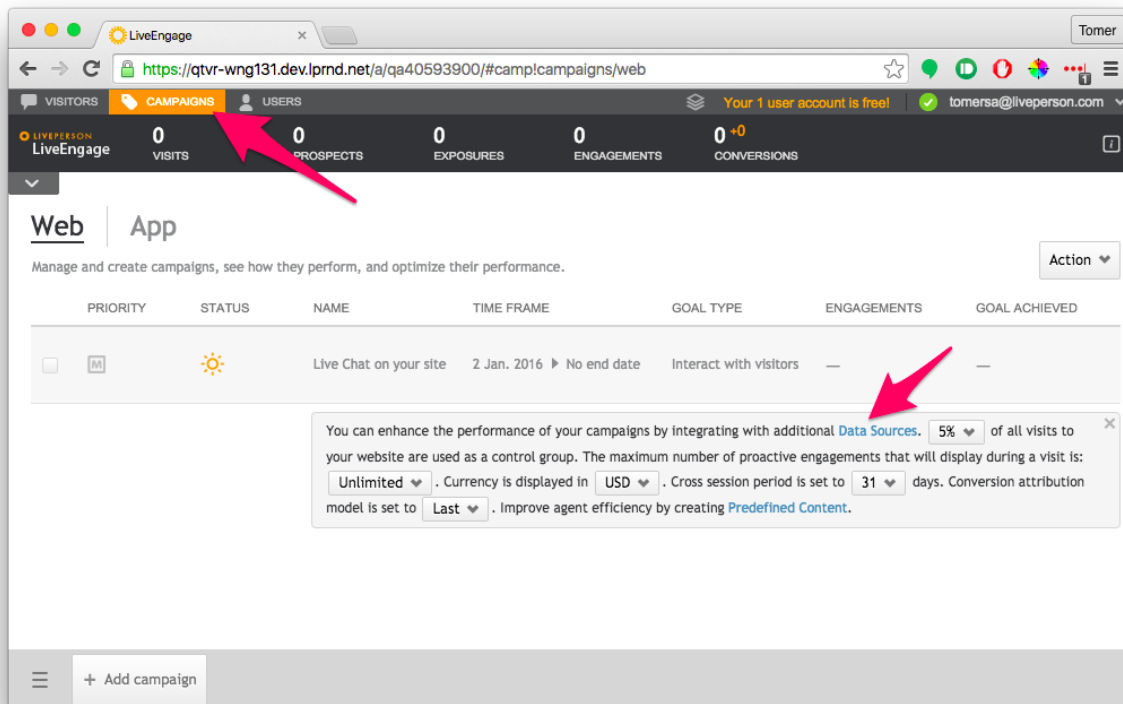
## Advanced features

More advanced configurations (image size, compression rate, etc..) [here](#)

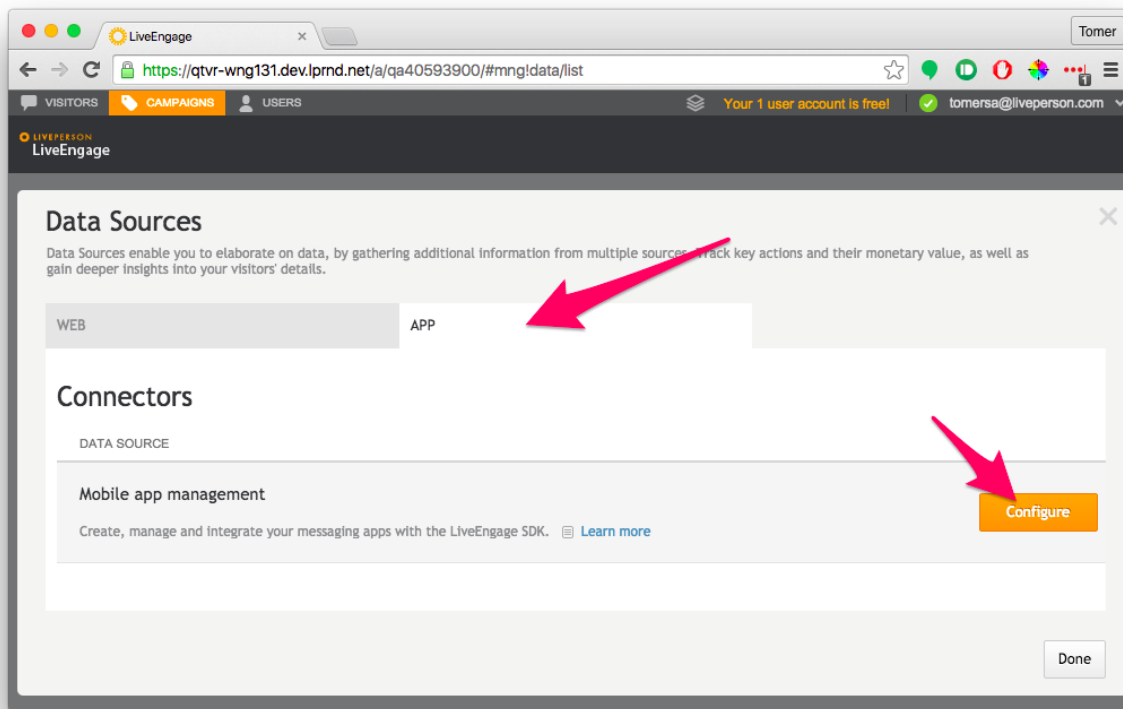
## LiveEngage Configuration

### Enable Push Notifications

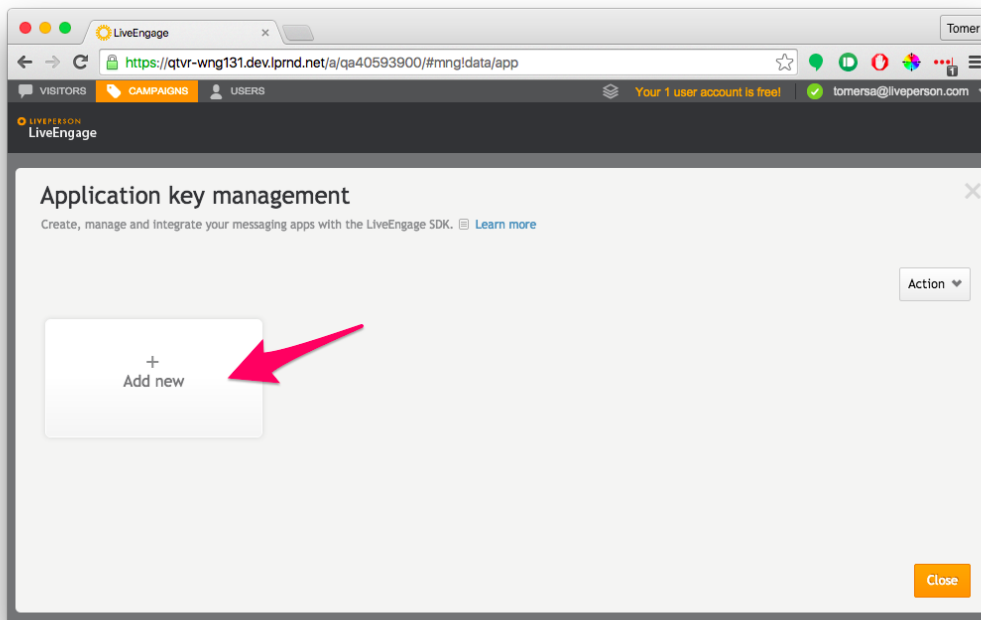
Log into your LiveEngage account using an administrator's credentials and navigate to **Campaigns**.



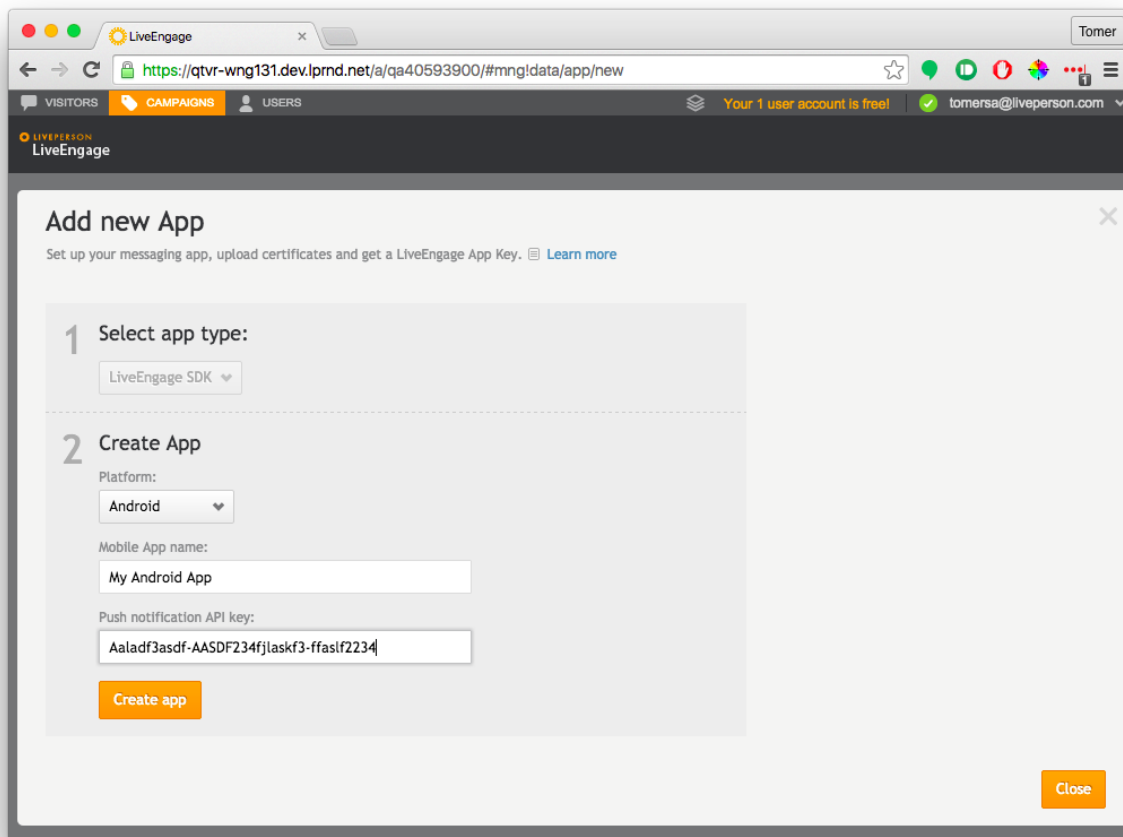
1. Click **Data Sources**, and then select **App**.



2. Click **Configure**.



3. Click **Add new** to associate your app with the LiveEngage account.
4. Select your platform as Android, enter your app's name and your push notification API key, and then click **Create app**.  
Refer to the [Notifications](#) section on how to get the notification API key.



The screenshot shows a web browser window with the LiveEngage interface. The main content area is titled 'Add new App' and includes a sub-header: 'Set up your messaging app, upload certificates and get a LiveEngage App Key. [Learn more](#)'. The form is divided into two sections: '1 Select app type:' with a dropdown menu set to 'LiveEngage SDK', and '2 Create App'. The '2 Create App' section contains three input fields: 'Platform:' with a dropdown set to 'Android', 'Mobile App name:' with the text 'My Android App', and 'Push notification API key:' with the text 'Aaladf3asdf-AASDF234fjlaskf3-ffaslf2234'. An orange 'Create app' button is located below these fields. A 'Close' button is in the bottom right corner of the form area. The browser's address bar shows the URL 'https://qtrv-wng131.dev.lpmc.net/a/qa40593900/#mngldata/app/new'. The top navigation bar includes links for 'VISITORS', 'CAMPAIGNS', and 'USERS', along with a status message 'Your 1 user account is free!' and a user profile for 'tomersa@liveperson.com'.

4. Click **Close** to finish the process.

# Appendix

## Security

Security is a top priority and key for enabling trusted, meaningful engagements.

LivePerson's comprehensive security model and practices were developed based on years of experience in SaaS operations, close relationships with Enterprise customers' security teams, frequent assessments with independent auditors, and active involvement in the security community.

LivePerson has a comprehensive security compliance program to help ensure adherence to internationally recognized standards and exceed market expectations. Among the standards LivePerson complies with are: SSAE16 SOC2, ISO27001, PCI-DSS via Secure Widget, Japan's FISC, SafeHarbor, SOX, and more.

Our applications are developed under a strict and controlled Secure Development Life-Cycle: Developers undergo secure development training, and security architects are involved in all major projects and influence the design process. Static and Dynamic Code Analysis is an inherent part of the development process and, upon maturity, the application is tested for vulnerabilities by an independent penetration testing vendor. On average, LivePerson undergoes 30 penetration tests each year.

## Dependencies

### **com.squareup.okhttp3:okhttp:3.4.1**

An HTTP+HTTP/2 client for Android and Java applications

### **com.neovisionaries:nv-websocket-client:1.30**

High-quality WebSocket client implementation in Java.

### **com.squareup.picasso:picasso:2.5.2**

An Android library for managing images and the memory they use.

**\*\*** If you already use one of the libraries above (either in the same library version or any other version), best practice is to keep the Inapp Messaging SDK in a separate module to avoid build gradle collision

## Open Source List

| Name                | Site  | License   |
|---------------------|---|---|
| Picasso             | <a href="http://square.github.io/picasso/">http://square.github.io/picasso/</a>   | <a href="http://square.github.io/picasso/#license">http://square.github.io/picasso/#license</a>   |
| OKHTTP              | <a href="http://square.github.io/okhttp/">http://square.github.io/okhttp/</a>   | <a href="https://github.com/square/okhttp/blob/master/LICENSE.txt">https://github.com/square/okhttp/blob/master/LICENSE.txt</a>                                       |
| nv-websocket-client | <a href="https://github.com/TakahikoKawasaki/nv-websocket-client">https://github.com/TakahikoKawasaki/nv-websocket-client</a> | <a href="https://github.com/TakahikoKawasaki/nv-websocket-client/blob/master/LICENSE">https://github.com/TakahikoKawasaki/nv-websocket-client/blob/master/LICENSE</a> |

## Localization Strings

### Android resources introduction:

Android resources are: Strings, drawables, layouts etc. During compile time, all resources are moved to the same location. App resources receive higher priority, and, due to this, in case the SDK and the App share the same resource name, the value of the App will be used. This is under OS responsibility.

### Language implementation:

SDK language support is split into two scenarios:

- **Device settings:** Uses device settings language → App's language is identical to the device language.
- **Host app settings:** App sets its own language regardless of device settings language → language may be different from device language.

*Note: The SDK language will be the same as the app language. The SDK cannot work with a language that is different from the app language. If the SDK does not support the app language, it will use the default language instead.*

The SDK contains a *values* folders for each supported language. For a list of supported languages, see [LiveEngage System Requirements and Language Support](#). Each folder contains

a strings file, where all strings are located for a specific language. Learn more about supporting different languages [here](#).

The SDK allows you to override the string localization of any supported language in LiveEngage. To apply a custom localization files with your own strings, create a strings file in the app's values folder (specific values folder for the required language). This option gives the ability to change strings, and to support languages that the SDK currently does not support.

*Note: In order to avoid collisions, each [SDK resource](#) starts with a prefix of “lp”. This is to avoid cases where the SDK and the host app use the same ID for a specific string, for example, dialog done button.*

*Example: <string name="lp\_resend\_failed\_masked\_message">Message failed to send. Please re-enter message and send again.</string>*

## Demo Project

The SDK is provided with a sample application called “SampleApp” that demonstrate the use of the SDK in a host app.

### Project structure explained

#### **MainActivity class**

This is the main class of the application. It gets the user data (first name, last name, the phone), the account number and authentication code (if required by the account) in order to set them to the SDK.

The main screen has two buttons to optionally open the conversation in Activity mode or in Fragment mode.

The Language and Region controls are used to test localization.

Sample Application

Account ID: \_\_\_\_\_

Auth Code: \_\_\_\_\_

OPEN ACTIVITY

OPEN FRAGMENT

First Name: \_\_\_\_\_

Last Name: \_\_\_\_\_

Phone Number: \_\_\_\_\_

▼ language region

UPDATE LANGUAGE CLEAR ☐ Set Callback

January 22, 2017  
1:07:30 PM  
SDK version 2.0.0.17

### **setCallBack method**

Sets the host app implementation of [LivePersonCallback](#) to the SDK. This implementation simply display a toast message on every callback received.

### **initActivityConversation method**

Opens the conversation view in Activity mode (see [Quick Start](#))

### **openFragmentManager method**

Opens the conversation view in Fragment mode (see [Quick Start](#)).

This method starts the [FragmentManagerActivity](#) that is simply the fragment container for the conversation fragment obtained from the SDK.

### **FragmentManagerActivity class**

This is an activity class that has a fragment container. It gets the conversation fragment from the SDK (*LivePerson.getConversationFragment()*) and attach it to the container.

## Push package

The *push* package contains sample code for implementing push notification in both [Google GCM](#) or [Google FCM](#). This is a simple code taken from Google documentation and integrated to be used in SampleApp and the Messaging SDK.

## branding.xml

Demonstrate the overriding of parameters (e.g. colors, dimensions etc.) used by the SDK. Please refer to the [Configuring the SDK](#) section for more details.

This document, materials or presentation, whether offered online or presented in hard copy ("LivePerson Informational Tools") is for informational purposes only. LIVEPERSON, INC. PROVIDES THESE LIVEPERSON INFORMATIONAL TOOLS "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

The LivePerson Informational Tools contain LivePerson proprietary and confidential materials. No part of the LivePerson Informational Tools may be modified, altered, reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), without the prior written permission of LivePerson, Inc., except as otherwise permitted by law. Prior to publication, reasonable effort was made to validate this information. The LivePerson Information Tools may include technical inaccuracies or typographical errors. Actual savings or results achieved may be different from those outlined in the LivePerson Informational Tools. The recipient shall not alter or remove any part of this statement.

Trademarks or service marks of LivePerson may not be used in any manner without LivePerson's express written consent. All other company and product names mentioned are used only for identification purposes and may be trademarks or registered trademarks of their respective companies.

LivePerson shall not be liable for any direct, indirect, incidental, special, consequential or exemplary damages, including but not limited to, damages for loss of profits, goodwill, use, data or other intangible losses resulting from the use or the inability to use the LivePerson Information Tools, including any information contained herein.

© 2017 LivePerson, Inc. All rights reserved.

**Copyright notice** – ©2017 LivePerson, Inc. All rights reserved.