Cordova LivePerson Messaging Plugin

> Examples of how to interact with the LivePerson Messaging SDK

Versioning

Current Version: v2.5.0

Running with the following versions of the LivePerson Messaging SDK

- iOS SDK v2.5.x
- Android SDK v2.5.x

PLEASE NOTE -- XCODE UPGRADE REQUIRED!!

- Any version of the Cordova plugin v2.1.x and above has been compiled with the xcode version 8.3.1
- Any apps you build with these versions of the iOS SDK (v2.1.5 and above) will require the above xcode version to build and run - due to the change in Swift version numbers.
- Xcode v8.3.1 also requires macOS Sierra in order to install!

Installing / Getting started

"Where do I find the latest version of the plugin?"

Repo has been restructured. The various versions of the plugin live here:

`/plugins/`

The latest release v2.5.x is here

/plugins/v2.5.x/MessagingSDKPlugin

If you need to reinstall the plugin to your app, make sure you pull it from this folder.

e.g if you were working in the sampleapp01 example folder

```
cd apps/sampleapp01
cordova plugin remove com.liveperson.messagingSDK
cordova plugin add
../../plugins/v2.5.x/MessagingSDKPlugin
```

iOS

Prerequisites

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

- XCode 8.3 or later
- Swift 3.1 or later, or Objective-C

Note: For information on supported operating systems and devices, refer to System Requirements.

Add iOS SDK Frameworks - Recommended via CocoaPods

The SDK is also compatible with CocoaPods, a dependency manager for Swift and Objective-C Cocoa projects. CocoaPods has thousands of libraries and is used in over 2 million apps. It can help you scale your projects elegantly and provides a standard format for managing external libraries.

1. Install cocoapods using the following command:

```
$ gem install cocoapods
```

2. Navigate to your project folder and init new pod using the following command:

```
$ pod init
```

3. Podfile should be created under your project’s folder.

> To integrate Liveperson Messaging SDK into your Xcode project using CocoaPods, specify it in your Podfile:

```
source
'https://github.com/LivePersonInc/iOSPodSpecs.git'
platform :ios, '8.0'
use_frameworks!

target '<Your Target Name>' do
    pod 'LPMessagingSDK'
end
```

4. Run the following command in the terminal under your project folder:

```
$ pod install
```

5. Incase you wish to upgrade to the latest SDK version and you have already ran 'pod install', Run the following command:

```
$ pod update
```

Installation of iOS Libraries by Copying to Xcode Project

Copy all framework and bundle files into the project, including the bundle file In project settings, navigate to the General tab, and add all Framework files to the Embedded Binaries section.

Installation notes

In project settings, navigate to the Build Phases tab, and click the + button to add a New Run Script Phase. Add the script below in order to loop through the frameworks embedded in the application and remove unused architectures (used for simulator). This step is a workaround for known iOS issue http://www.openradar.me/radar?id=6409498411401216 and is necessary for archiving your app before publishing it to the App Store.

If frameworks installed using CocoaPods, use the following script:

```
bash
"${SRCROOT}/Pods/LPMessagingSDK/LPMessagingSDK/LPInfr
a.framework/frameworks-strip.sh"
```

If frameworks installed using copy to Xcode project, use the following script:

```
bash
"${BUILT_PRODUCTS_DIR}/${FRAMEWORKS_FOLDER_PATH}/LPIn
fra.framework/frameworks-strip.sh"
```

Manual Framework installation Notes

Follow the usual steps for adding the embedded binary .frameworks and

.bundle files into the iOS app via Xcode. Check the Video Link for examples of how this can be done.

- Add SDK plugin with latest version

---- DO NOT RUN cordova build ios on CLI until you have been into XCODE! ---

- Open project workspace in XCODE
    - accept all recommend project settings
    - select project
    - add embedded binaries for the SDK and frameworks
    - **CLEAN PROJECT!**
    - **BUILD PROJET!** == should pass & you will warnings about missing cordova libs as we have not "built in cordova" yet!

- go back to CLI

- run cordova build ios command which should now succeed

- run in xcode or CLI

> if you get a build/run error about missing simulators, then edit this file -- <your app folder>/platforms/ios/cordova/lib/start-emulator and change the default iOS emulator to run or add iPhone 5s to your list of emulated devices

---

Android

The plugin itself does NOT include any copies of the Android SDK aars libraries. You should download the latest version of the Android SDK from here on github.

https://developers.liveperson.com/android-quickstart.html

BUT ALWAYS CHECK GITHUB RELEASES LINK ABOVE FOR THE LATEST VERSION

Sample Apps Included

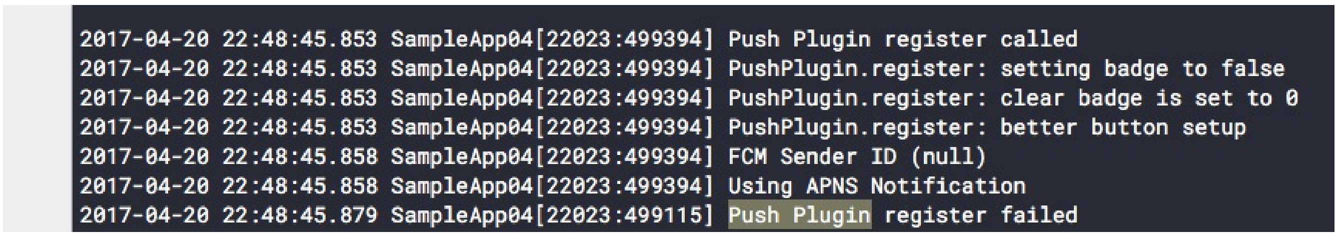
Within the apps/ folder at the root of this repo you will find some sample apps demoing the plugin. Here is a breakdown.

apps/sampleapp04 -- iOS Only SDK v2.1.5

Has been created as reference app using iOS frameworks.

This app does NOT include an Android application - refer to SampleApp03 for the latest Android example app.

Includes a basic iOS Cordova App with Messaging integration and the PhoneGap Push plugin installed - but not yet tested as Simulator restrictions prevent this...



```
2017-04-20 22:48:45.853 SampleApp04[22023:499394] Push Plugin register called
2017-04-20 22:48:45.853 SampleApp04[22023:499394] PushPlugin.register: setting badge to false
2017-04-20 22:48:45.853 SampleApp04[22023:499394] PushPlugin.register: clear badge is set to 0
2017-04-20 22:48:45.853 SampleApp04[22023:499394] PushPlugin.register: better button setup
2017-04-20 22:48:45.858 SampleApp04[22023:499394] FCM Sender ID (null)
2017-04-20 22:48:45.858 SampleApp04[22023:499394] Using APNS Notification
2017-04-20 22:48:45.879 SampleApp04[22023:499115] Push Plugin register failed
```

apps/sampleapp03 -- Android -- Authentication and Push Plugin

- includes authentication for both platforms
- phone gap push plugin RC 2.0 for Android and iOS
  - working on Android
  - not yet tested on iOS due to account limitations TBC.
- Android SDK v2.1.3
- iOS SDK v2.1.2
  - runs on xcode 8.2.1

Api Reference

filename: plugins/MessagingSDKPlugin/www/LPMessagingSDK.js

This is where we expose the different function names within the wrapper to call the native code for starting, configuring a messaging conversation. In the current version there is just one function exposed which takes in an "action" arg for telling the native code what to do. You must also supply a successCallback js func / errorCallback js func / account id (clone or prod) + any arguments needed by the function.

API Function definition

```
lp_conversation_api: function(action, args,
successCallback, errorCallback)
```

Supported values for action :

*************************** **NEW IN v2.5.x** *************************************

"close_conversation_screen"

sample call:

```
lpMessagingSDK.lp_conversation_api(
  'close_conversation_screen', [accountNumber],
  function (data) {
    var eventData = JSON.parse(data);
    console.log('@@@ js ... unique
close_conversation_screen SDK callback');
  },
  function (data) {
    var eventData = JSON.parse(data);
    console.log('@@@ js ... unique
close_conversation_screen SDK error callback');
  }
);
```

can be triggered if the callbacks you bind to trigger an error response and you want to close the conversation screen as a result and provide the user with instructions within the web layer.

"lp_sdk_init"

Must be called before trying to use any other methods. Requires the account number to be passed within args parameter

sample call...

```
var success = function(message) {
    console.log("OnEvent JS: " + message);
}

var failure = function() {
    console.log("Error calling lp_conversation_api
Plugin");
}

lpMessagingSDK.lp_conversation_api("lp_sdk_init",
["123456"], success, failure);
```

"register_pusher"

Used to register device tokens with the SDK for handling push notifications

- **api method** : lpMessagingSDK.lp_conversation_api
- **action** : 'register_pusher'
- **args** : [accountId,deviceToken]
    - deviceToken should be obtained by your app using relevant cordova plugin
- Reccomend this method is called within the success callback of lp_sdk_init to ensure SDK is ready to receive the token.

```
lpMessagingSDK.lp_conversation_api(
    "lp_sdk_init", [this.settings.accountId,
sdkConfig],
    function(data) {
        var eventData = JSON.parse(data);
        console.log("@@@ js ... unique lp_sdk_init
SDK callback");
        lpMessagingSDK.lp_conversation_api(
            "register_pusher",
[app.settings.accountId,app.deviceToken],
            function(data) {
                //var eventData = JSON.parse(data);
                console.log("@@@ js ... unique
register_pusher SDK callback .."+data);
            },
            function(data) {
                // var eventData = JSON.parse(data);
                console.log("@@@ js ... unique
register_pusher SDK error callback ..."+data);
            }
        );
    },
    function(data) {
        var eventData = JSON.parse(data);
        console.log("@@@ js ... unique lp_sdk_init
SDK error callback");
```

```
    }
);
```

"start_lp_conversation"

Used to open the Messaging conversation window and connect to LivePerson.

Includes support for authentication JWT token for implicit flow

```
var authenticationCode =
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIrOTc
yLTMtNTU1NS01NTUiLCJpc3MiOiJodHRwczovL2lkcC5saXZlcGVy
c29uLm5ldCIsImF1ZCI6ImFjYzpxYTU3MjIxNjc2IiwiZXhwIjoxN
TM0OTcxOTMwLCJpYXQiOjE0NzE4OTk5NDIsIm5hbWUiOiJFaXRhbi
J9.Fh0sG-iu-
VMZRFRbUNK0kEzb7Y1BXtQHOKaHL2y40y_c4mBvmQDCOYNWJ1ZEea
yTNuLboYx6L8xEoC5xZIFnVv2N4a36BBU88fNuhe9Em2b5qNdVbdB
tIJQoBY5ep5O2geAaCVA7A7oS8ysWVGn9CV4btH_D5sU2jGr3ml8y
fJA"

lpMessagingSDK.lp_conversation_api(
    "start_lp_conversation",
    [
        "123456",
        authenticationCode
    ],
    success,
    failure);
```

authenticationCode is optional arg parameter. If omitted then the conversation will be **unauthenticated**

"lp_clear_history_and_logout"

Used to clear the current user data and unregister the device from push notifications.

PLEASE NOTE: Impact on Push Notifications

Calling this method will mean the device is no longer going to receive push notifications until the next user logs back in and opens the conversation screen to send the latest JWT token to the SDK and establish a connection. After that point the device will receive push notifications again for that user when not viewing the conversation screen.

- When the customer logs out of your app, call this method to clear the local device SDK history and unregister the device from push notifications.
- Then once the next user logs in, **remember to call lp_sdk_init before starting a new conversation for the next user when you supply the updated and relevant JWT token**
- see example below for doing this in the callback for logout ready for the next user.
- You must supply the account id for your LivePerson account number into this method

```
lpMessagingSDK.lp_conversation_api(
    "lp_clear_history_and_logout",
[this.settings.accountId],
    function(data) {
        var eventData = JSON.parse(data);
        console.log("@@@ js ... unique
clearDeviceHistoryAndLogout SDK callback ...data =>
"+data);
        console.log("@@@ js ... post logout...now
auto reinitialise the SDK for next user to save
button press in demo!");
        app.lpMessagingSdkInit();
    },
    function(data) {
        var eventData = JSON.parse(data);
        console.log("@@@ js ... unique
clearDeviceHistoryAndLogout SDK error callback
...data => "+data);
    }
);
```

- callback event name : LPMessagingSDKClearHistoryAndLogout

"reconnect_with_new_token"

- **args** : [token]
    - ~~**in that specific order!**~~
    - ~~accountId is required for iOS version of the function~~

within your javascript successCallback function, you must listen for the specific eventName that tells you the SDK has detected that the customer JWT has expired and must be refreshed.

- Listen for the correct event in your successCallback method and call your relevant app function to get the new token

```
successCallback: function(data) {

    var eventData = JSON.parse(data);

    if (eventData.eventName ==
'LPMessagingSDKTokenExpired') {
        console.log("authenticated token has
expired...refreshing...");
        this.lpGenerateNewAuthenticationToken();
    }

```

```
  },
```

- generate your new token calling your IDP / JWT service via javascript

```
lpGenerateNewAuthenticationToken: function() {
    // code to generate new fresh JWT would go
here...
    // this example uses a hard coded JWT which has
the new expiry time.
    var jwt =
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJUQUx
LVEFMSy1JRC0xMjM0NTY3ODkwIiwiaXNzIjoiaHR0cHM6Ly93d3cu
dGFsa3RhbGsuY28udWsiLCJleHAiOjE1MTQ3MTg2NzEwMDAsImlhd
CI6MTQ4NzE1OTMzNzAwMCwicGhvbmVfbnVtYmVyIjoiKzEtMTAtMz
Q0LTM3NjUzMzMiLCJscF9zZGVzIjpbeyJ0eXBlIjoiY3RtcmluZm8
iLCJpbmZvIjp7ImNzdGF0dXMiOiJjYW5jZWxsZWQiLCJjdHlwZSI6
InZpcCIsImN1c3RvbWVySWQiOiIxMzg3NjZBQyIsImJhbGFuY2UiO
i00MDAuOTksInNvY2lhbElkIjoiMTEyNTYzMjQ3ODAiLCJpbWVpIj
oiMzU0MzU0NjU0MzU0NTY4OCIsInVzZXJOYW1lIjoidXNlcjAwMCI
sImNvbXBhbnlTaXplIjo1MDAsImFjY291bnROYW1lIjoiYmFuayBj
b3JwIiwicm9sZSI6ImJyb2tlciIsImxhc3RQYXltZW50RGF0ZSI6e
yJkYXkiOjE1LCJtb250aCI6MTAsInllYXIiOjIwMTR9LCJyZWdpc3
RyYXRpb25EYXRlIjp7ImRheSI6MjMsIm1vbnRoIjo1LCJ5ZWFyIjo
yMDEzfX19LHsidHlwZSI6InBlcnNvbmFsIiwicGVyc29uYWwiOnsi
Zmlyc3RuYW1lIjoiSm9objc3IiwibGFzdG5hbWUiOiJCZWFkbGU3N
yIsImFnZSI6eyJhZ2UiOjM0LCJ5ZWFyIjoxOTgwLCJtb250aCI6NC
wiZGF5IjoxNX0sImNvbnRhY3RzIjpbeyJlbWFpbCI6ImpiZWFkbGU
5OUBsaXZlcGVyc29uLmNvbSIsInBob25lIjoiKzEgMjEyLTc4OC04
ODc3In1dLCJnZW5kZXIiOiJNQUxFIn19XX0.i-PFEBjgXR-
rEM30iGJAV-
4l0P58wysMEZxyoYdwOCTpIkmfkXtnztfyYRNdaBkpaF1AmZVtgEB
IFEYWLcSmcRWkMvnSUAKV0dv9QhR9tDbILsdyd-
DEFB_RcmW8rXB7rWSoSJa4z3EMatpoC7CzaUrih8IycB2X4FuKuxL
9mOg";

    // pass the
    lpMessagingSDK.lp_conversation_api(
        "reconnect_with_new_token",
[jwt,app.settings.accountId],
        this.successCallback,
        this.errorCallback
    );
    console.log('lpGenerateNewAuthenticationToken
completed --> new jwt -->  ', jwt);
},
```

- once you have the new token pass it back down into the native application using the following cordova API call

```
lpMessagingSDK.lp_conversation_api(
    "reconnect_with_new_token",
    [jwt,app.settings.accountId],
    this.successCallback,
    this.errorCallback
);
```

- The SDK will pass the token back to LivePerson using the reconnect method of the SDK to refresh the token and continue authenticated conversations.

"set_lp_user_profile"

Used to send **unauthenticated** customer information to the agent where authenticated equivalents are not being sent.

PLEASE NOTE : unclear which of these values are still being used on server side so subject to change/deprecation in subsequent versions.

args array parameter mapping:

- 0 : accountId : "123456"

- 1 : first name : "John"

- 2 : last name : "Doe"

- 3 : nickname : "JD"

- 4 : profile image url : "https://s-media-cache-ak0.pinimg.com/564x/a2/c7/ee/a2c7ee8982de3bae503a730fe4562cf9.jpg"

(**Android Deprecated in v2.5** - value will be ignored - send null if not applicable)

(**iOS still supported in v2.5**)

- 5 : customer phone number : "555-444-12345"

(iOS Only)

- 6 : uid : "UID123145"
- 7 : employeeID : "employeeId123123123"

^ Both of the above seemingly not being read on server side for iOS so use with caution.

```
lpMessagingSDK.lp_conversation_api(
  'set_lp_user_profile',
  [
    accountId,
    'John',
    'Doe',
    'NickName:JD',
    'https://s-media-cache-
ak0.pinimg.com/564x/a2/c7/ee/a2c7ee8982de3bae503a730f
e4562cf9.jpg',
    'tel:555-444-12345',
    'uid_5678',
    'employeeId_1234'
  ],
  function(data) {
    var eventData = JSON.parse(data);
    console.log('@@@ js ... unique
set_lp_user_profile SDK callback');
  },
  function(data) {
    var eventData = JSON.parse(data);
    console.log('@@@ js ... unique
set_lp_user_profile SDK error callback');
  }
);
```

Sending Secure Information via JWT tokens

If you wish to send secure, authenticated information about the customer to your agent, it should be encoded and encrypted within your JWT token

https://s3-eu-west-1.amazonaws.com/ce-sr/CA/security/Authenticated+Interactions+with+oAuth+2.0.pdf

For a list of supported engagement attributes within a JWT token payload, see the following example JWT:

```
  {
    "sub": "MY_UNIQUE_CUSTOMER_IDENTIFIER_GOES_HERE",
    "iss": "https://www.talktalk.co.uk",
    "exp": 1514718671,
    "iat": 1487159337,
    "phone_number": "+1-10-344-3765333",
    "lp_sdes": [
      {
        "type": "ctmrinfo",
        "info": {
          "cstatus": "cancelled",
          "ctype": "vip",
          "customerId": "138766AC",
          "balance": -400.99,
          "socialId": "11256324780",
          "imei": "3543546543545688",
          "userName": "user000",
          "companySize": 500,
          "accountName": "bank corp",
          "role": "broker",
          "lastPaymentDate": {
            "day": 15,
            "month": 10,
            "year": 2014
          },
          "registrationDate": {
            "day": 23,
            "month": 5,
            "year": 2013
          }
        }
      },
      {
        "type": "personal",
        "personal": {
          "firstname": "AN",
          "lastname": "Other",
          "age": {
            "age": 34,
            "year": 1980,
            "month": 4,
            "day": 15
          },
          "contacts": [
            {
              "email": "another@liveperson.com",
              "phone": "+1 212-788-8877"
            }
          ],
          "gender": "MALE"
      }
    }
  ]
}
```

HOWTO : Update iOS Frameworks in your existing app

This video shows step by step how to replace the existing iOS frameworks and bundles in your app when their is a new version / hotfix released and you do not want to have to remove and add the plugin back into the app again...

Video Link

Cordova Plugin Callback Names

The following callbacks are fired from the iOS / Android apps back up into the Cordova plugin for processing / actions in Javascript

Register your Global Async Callback Handler

API Function definition

```
lpMessagingSDK.lp_register_event_callback:
function(args, successCallback, errorCallback)
```

- register your global async callback handler for monitoring these events.

```
lpMessagingSDK.lp_register_event_callback(
        [accountId],
        this.globalAsyncEventsSuccessCallback,
        this.globalAsyncEventsErrorCallback
    );
```

SuccessCallback Event Object

In your js successCallback function you will receive a data object that must be parsed from a string into JSON

```
var eventData = JSON.parse(data);
```

This object has an eventName property which matches the above callbacks we support.

```
eventData.eventName
```

There may be other additional data points depending on the callback and context.

---

iOS + Android Callbacks

Immediate responses

Certain events are now returning immediate responses by triggering the respective, unique success/error callbacks passed into certain API functions.

lpMessagingSDK.lp_conversation_api with the following actions will return success/failure ASAP.

- **action**: start_lp_conversation | **eventName**: LPMessagingSDKStartConversation
    - based on the SDK API call not failing – does not mean the conversation screen has loaded without errors etc, just means we called the method successfully.
- **action**: set_lp_user_profile | **eventName**: LPMessagingSDKSetUserProfile
    - called the API method and did/did not throw errors as a result.
- **action**: lp_sdk_init | **eventName**: LPMessagingSDKInit
    - SDK has initialised successfully based on calling the method and not throwing any errors.
- **action**: lp_clear_history_and_logout | **eventName**: LPMessagingSDKClearHistoryAndLogout
    - SDK has cleared device history and unregistered push notifications for the device.
- **action**: register_pusher | **eventName**: LPMessagingSDKRegisterLpPusher
    - successully registered the push token via the API call. **Should only be called once lp_sdk_init has succeeded**. Suggest triggering within the success callback of lp_sdk_init
    - Refer to register_pusher section for exact details, usage and implications

Triggering asynchronously

- This will be the single handler for all the following async events
- example function that JSON parses the data to get the eventData.eventName property listed below

```
globalAsyncEventsSuccessCallback: function(data) {
    var eventData = JSON.parse(data);
    console.log(
        '@@@ globalAsyncEventsSuccessCallback --> ' +
data
    );
    if (eventData.eventName ==
'LPMessagingSDKTokenExpired') {
        console.log("@@@ authenticated token has
expired...refreshing...");
        app.lpGenerateNewAuthenticationToken();
    }
},
```

eventData.eventName values listed below. additional eventData properties are indented

- "LPMessagingSDKCustomButtonTapped" (iOS Only)
- "LPMessagingSDKAgentDetails"
    - agentName : String
- "LPMessagingSDKActionsMenuToggled" (iOS Only)
    - toggled : true|false
- "LPMessagingSDKHasConnectionError" (iOS Only)
    - error : String
- "LPMessagingSDKCSATScoreSubmissionDidFinish"
- "LPMessagingSDKAuthenticationFailed"
    - error : String
- "LPMessagingSDKTokenExpired"
    - accountId : string
- "LPMessagingSDKError"
    - error : String
- "LPMessagingSDKAgentIsTypingStateChanged"
    - isTyping : true|false
- "LPMessagingSDKConversationStarted"
    - conversationID : string
- "LPMessagingSDKConversationEnded"
    - conversationID : string
- "LPMessagingSDKConversationCSATDismissedOnSubmission"

    - conversationID : string
- "LPMessagingSDKConnectionStateChanged"
    - isReady : true|false
- "LPMessagingSDKOffHoursStateChanged"
    - isOffHours : true|false
- "LPMessagingSDKConversationViewControllerDidDismiss" (iOS only)

Immediate response callback execution:

- "LPMessagingSDKInit"
- "LPMessagingSDKClearHistoryAndLogout"
- "LPMessagingSDKSetUserProfile"
- "LPMessagingSDKReconnectWithNewToken"
- "token" : new jwt
- "LPMessagingSDKRegisterLpPusher"
- "deviceToken" : device token supplied
- "LPMessagingSDKStartConversation"
    - "type" : "authenticated" | "unauthenticated"

Triggering asynchronously -- register your global async callback handler for monitoring these events.

- "LPMessagingSDKObseleteVersion" (iOS Only)
- "LPMessagingSDKCsatDismissed" (Android Only)

!!! Not currently implemented/applicable in callbacks!!!

- "LPMessagingSDKAgentAvatarTapped"
- "LPMessagingSDKCSATCustomTitleView"

Refer to native documentation and if you are missing a specific callback pleae let us know!