Cordova Messaging Plugin

"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 is here

/plugins/iOS-hotfix-v2.1.2/MessagingSDKPlugin

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

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

```
cd apps/sampleapp01
cordova plugin remove com.liveperson.messagingSDK
cordova plugin add ../../plugins/iOS-hotfix-v2.1.2/MessagingSDKPlugin
```

iOS Install

You will still need to 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.

SDK Frameworks and Bundle Install summary

- 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. Once downloaded, follow the instructions in the `/docs` folder for adding the Messaging SDK to your Android Cordova app in Android studio.

Android v2.1.0 `aars` files are located in `/sdk-libs/android/v2.1.0` for your convenience **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/sampleapp01`

- Combined Android and iOS Cordova app – running SDK 2.1.2 on iOS and 2.1.0 on Android
- reference the apps/sampleapp01/www/js/index.js for examples of configuring the app in javascript to call the Cordova plugin and wrapper APIs

apps/sampleapp02-ios

- iOS only Cordova app – running SDK 2.1.2 on iOS
- reference the apps/sampleapp02-ios/www/js/index.js for examples of configuring the app in javascript to call the Cordova plugin and wrapper APIs

---

Cordova API Wrapper file:

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 :

"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);
```

"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.eyJzdWIiOiIrOTcyLTMt
NTU1NS01NTUiLCJpc3MiOiJodHRwczovL2lkcC5saXZlcGVyc29uLm5ldC
IsImF1ZCI6ImFjYzpxYTU3MjIxNjc2IiwiZXhwIjoxNTM0OTcxOTMwLCJp
YXQiOjE0NzE4OTk5NDIsIm5hbWUiOiJFaXRhbiJ9.Fh0sG-iu-
VMZRFRbUNK0kEzb7Y1BXtQHOKaHL2y40y_c4mBvmQDCOYNWJ1ZEeayTNuL
boYx6L8xEoC5xZIFnVv2N4a36BBU88fNuhe9Em2b5qNdVbdBtIJQoBY5ep
5O2geAaCVA7A7oS8ysWVGn9CV4btH_D5sU2jGr3ml8yfJA"

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

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

"reconnect_with_new_token"

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.eyJzdWIiOiJUQUxLVEFM
Sy1JRC0xMjM0NTY3ODkwIiwiaXNzIjoiaHR0cHM6Ly93d3cudGFsa3RhbG
suY28udWsiLCJleHAiOjE1MTQ3MTg2NzEwMDAsImlhdCI6MTQ4NzE1OTMz
NzAwMCwicGhvbmVfbnVtYmVyIjoiKzEtMTAtMzQ0LTM3NjUzMzMiLCJscF
9zZGVzIjpbeyJ0eXBlIjoiY3RtcmluZm8iLCJpbmZvIjp7ImNzdGF0dXMi
OiJjYW5jZWxsZWQiLCJjdHlwZSI6InZpcCIsImN1c3RvbWVySWQiOiIxMz
g3NjZBQyIsImJhbGFuY2UiOi00MDAuOTksInNvY2lhbElkIjoiMTEyNTYz
MjQ3ODAiLCJpbWVpIjoiMzU0MzU0NjU0MzU0NTY4OCIsInVzZXJOYW1lIj
oidXNlcjAwMCIsImNvbXBhbnlTaXplIjo1MDAsImFjY291bnROYW1lIjoi
YmFuayBjb3JwIiwicm9sZSI6ImJyb2tlciIsImxhc3RQYXltZW50RGF0ZS
I6eyJkYXkiOjE1LCJtb250aCI6MTAsInllYXIiOjIwMTR9LCJyZWdpc3Ry
YXRpb25EYXRlIjp7ImRheSI6MjMsIm1vbnRoIjo1LCJ5ZWFyIjoyMDEzfX
19LHsidHlwZSI6InBlcnNvbmFsIiwicGVyc29uYWwiOnsiZmlyc3RuYW1l
IjoiSm9objc3IiwibGFzdG5hbWUiOiJCZWFkbGU3NyIsImFnZSI6eyJhZ2
UiOjM0LCJ5ZWFyIjoxOTgwLCJtb250aCI6NCwiZGF5IjoxNX0sImNvbnRh
Y3RzIjpbeyJlbWFpbCI6ImpiZWFkbGU5OUBsaXZlcGVyc29uLmNvbSIsIn
Bob25lIjoiKzEgMjEyLTc4OC04ODc3In1dLCJnZW5kZXIiOiJNQUxFIn19
XX0.i-PFEBjgXR-rEM30iGJAV-
4l0P58wysMEZxyoYdwOCTpIkmfkXtnztfyYRNdaBkpaF1AmZVtgEBIFEYW
LcSmcRWkMvnSUAKV0dv9QhR9tDbILsdyd-
DEFB_RcmW8rXB7rWSoSJa4z3EMatpoC7CzaUrih8IycB2X4FuKuxL9mOg"
;

    // pass the
    lpMessagingSDK.lp_conversation_api(
        "reconnect_with_new_token", [jwt],
        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],
    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

args array parameter mapping:

- 1 : accountId : "123456"
- 2 : first name : "John"
- 3 : last name : "Doe"
- 4 : nickname : "JD"
- 5 : profile image url : "https://s-media-cache-ak0.pinimg.com/564x/a2/c7/ee/a2c7ee8982de3bae503a730fe4562c
- 6 : customer phone number : "555-444-12345"

```
lpMessagingSDK.lp_conversation_api(
    "set_lp_user_profile",
    [
            "123456",
            "John",
            "Doe",
            "JD",
            "https://s-media-cache-ak0.pinimg.com/564x/a2/c7/ee/a2c7ee8982de3bae503a730fe4562cf9.jpg",
            "555-444-12345"
    ],
    success,
    failure
);
```

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": "TALKTALK-ID-1234567890",
  "iss": "https://www.talktalk.co.uk",
  "exp": 1514718671000,
  "iat": 1487159337000,
  "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": "John99",
        "lastname": "Beadle99",
        "age": {
          "age": 34,
          "year": 1980,
          "month": 4,
          "day": 15
        },
        "contacts": [
          {
            "email": "jbeadle99@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

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

The following is a list of the expected eventName property passed to the successCallback function when these events are raised. Where applicable additional data points are passed and noted below...

- "LPMessagingSDKCustomButtonTapped"
- "LPMessagingSDKAgentDetails"
  - agentName : String
- "LPMessagingSDKActionsMenuToggled"
  - toggled : true|false
- "LPMessagingSDKHasConnectionError"
  - error : String
- "LPMessagingSDKCSATScoreSubmissionDidFinish"
- "LPMessagingSDKObseleteVersion"
  - error : String
- "LPMessagingSDKAuthenticationFailed"
  - error : String
- "LPMessagingSDKTokenExpired"
  - accountId : string
- "LPMessagingSDKError"
  - error : String
- "LPMessagingSDKAgentIsTypingStateChanged"
  - agentIsTyping : true|false
- "LPMessagingSDKConversationStarted"
  - conversationID : string
- "LPMessagingSDKConversationEnded"
  - conversationID : string

- "LPMessagingSDKConversationCSATDismissedOnSubmittion"
    - conversationID : string
- "LPMessagingSDKConnectionStateChanged"
    - isReady : true|false
- "LPMessagingSDKOffHoursStateChanged"
    - isOffHours : true|false
    - accountId : string

- "LPMessagingSDKConversationViewControllerDidDismiss"

!!! Not currently implemented in callbacks!!!

- "LPMessagingSDKAgentAvatarTapped"
- "LPMessagingSDKCSATCustomTitleView"
- "LPMessagingSDKObseleteVersion"

These will be added in upcoming versions. Refer to native documentation and if you are missing a specific callback pleae let us know!