Android LIP-Integration

Prerequisites

LIP Module's android minSdkVersion is 14 and targetSdkVersion is 23. Application can override these using application build.gradle without modifying SDK module build.gradle itself

Project Dependency

1. LIP Module uses Android Studio with Gradle
2.
```
android {
    compileSdkVersion 23
    buildToolsVersion “23.0.1”
}
```
3.
```
dependencies {
        classpath 'com.android.tools.build:gradle:2.1.2'
        classpath 'com.google.gms:google-services:3.0.0'
}
```

Application Gradle Script changes

1. settings.gradle file need to be changed to include dependency module (ex:include':sdk)
2. build.gradle need to be changed to compile dependency module under dependencies tag
   ex:compile project(':sdk')

Facebook client id addition

1. Define string `facebook_app_id` in array.xml with valid facebook application id. Refer below link to create application
   https://developers.facebook.com/docs/apps/register

Google client id addition

1. Create project and client ID for the application , refer below link
   https://developers.google.com/identity/sign-in/web/devconsole-project
   https://developers.google.com/maps/documentation/android-api/signup?hl=en

2. Create configration service from below link and add to application
   https://developers.google.com/mobile/add?platform=android&cntapi=signin&cnturl=https:%2F%2Fdevelopers.google.com%2Fidentity%2Fsign-in%2Fandroid%2Fsign-in%3Fconfigured%3Dtrue&cntlbl=Continue%20Adding%20Sign-In
   Downloaded google-services.json file need to overwrite which is present in lip-library folder. For more details refer
   https://developers.google.com/identity/sign-in/android/start-integrating
3. Define string `server_client_id` in array.xml, This id will be generated using google console as per step 1

Logitech client id addition

1. Define string `logitech_app_id` in array.xml with valid logi application id. This id will be unique for each application and will be provided by logi.

Note: SDK will expects all the above client id's else it will not work as expected , Refer Troubleshoot section

SDK Configuration

Application Specific Configuration

Each application can pass its static configuration through json file(Ex: Custom font, Application Internal Name). Resource specific can be overridden using android res folder concept

Static configuration liplibrary_config.json file can be put in res\raw folder so that SDK will read and initialize at startup

```
{
 "appName":harmony  /* This info used to validate TOU & Opt-In accepted by Identity
                                         Ex: harmony, brownie, circle,.... */
"keepScreenOn":true,  /*To keep screen on always */

 "fontSet":["font/BrownProTT-Regular.ttf",
                 "font/BrownProTT-Bold.ttf",
                 "font/BrownProTT-Light.ttf"] /*Application specific font, By Default SDK will use
                 BronProTT font as per spec*/
}
```

UI Customization

Application can change default Window background color , Button background , Text color and strings based on need except the initial screen (i.e Google/Facebook/Email with Logi logo screen).

Ex:

```
<string name="lip_signup_tou_url"
translatable="false">https://files.myharmony.com/Assets/legal/en/termsofuse.html</string>

<string name="lip_signup_privacy_policy_url"
translatable="false">https://files.myharmony.com/Assets/legal/en/privacypolicy.html</string>
<string name="lip_sign_up_create_id">SIGNUP/LOGIN</string>



<color name="lip_primary_white_button_pressedstate">#1bbc9b</color>
<color name="lip_window_background">#000000</color>
```

Note: Initial screen can't be customized

Locale enforcement on LIP

LIP supports multiple languages(5 languages). In case client application supports only few languages then application update resource configuration with default.

Ex:

```
private void updateLanguage(String language) {
        // en- English fr - French, de - German, nl - Dutch, es - Spanish
        if (!((language.equals("en")) || (language.equals("fr")) || (language.equals("de")) ||
                (language.equals("nl")) || (language.equals("es")))) {
            language = "en";
            Resources mResources = getResources();
            Configuration mConfig = mResources.getConfiguration();
            mConfig.locale = new Locale(language, "US");
            mResources.updateConfiguration(mConfig, mResources.getDisplayMetrics());
        }

    }
```

Terms of use & Privacy Policy

SDK expects Terms of use and Privacy policy in case application not accepted earlier. These are part of application claims. SDK expects URL from client application as below

```
LipConfiguration lipConfiguration = new LipConfiguration.Builder()
        .setIsVerifyEmail(false)    /*To indicate email verification Mandatory or Not*/
        .setAnalyticCallback(null) /* pass IAnalytics to log Analytics info by client app*/
        .setLoggerCallback(null)  /* pass ILogger to log info and error by client app*/
        .setServerUrl("https://XXX.XXX.XXX.logi.com/")
        .setTermsUseUrl("https://XXX.XXX.XXX.logi.com/xxx.html")
        .setPrivacyPolicyUrl("https://XXX.XXX.XXX.logi.com/pp.html")
        .build();
```

Client application can use update configuration in case it's already initialized

```
// Update Configuration when changed
LIPSdk.updateConfiguration(lipConfiguration);
```

SDK Dynamic Configuration

Application can pass certain configuration dynamically based on requirements.

```
LipConfiguration lipConfiguration = new LipConfiguration.Builder()
        .setIsVerifyEmail(false)    /*To indicate email verification Mandatory or Not*/
        .setAnalyticCallback(null) /* pass IAnalytics to log Analytics info by client app*/
        .setLoggerCallback(null)  /* pass ILogger to log info and error by client app*/
        .setServerUrl("https://XXX.XXX.XXX.logi.com/")
        .build();

// Initialize SDK as first time
LIPSdk.initialize(this.getApplication(),lipConfiguration);

// Update Configuration when changed
LIPSdk.updateConfiguration(lipConfiguration);
```

Troubleshooting

SDK supports social login like Google and Facebook, There might be problems in integration with application as we depends on certificates

Google Sign-In

1. Make sure we enable Google+ api in console and application packege registed in google console
2. Check the SHA1 key mentioned in google console and application signed SHA1 key is same or not
3. Pass empty `server_client_id` and check plan login works or not, In case login works but we are not getting acess token & id token then contact server team to get correct server client id

Facebook

4. Make sure we registered application in facebook console and it's MD5 key is same as the application signed

Sample Code

SDK Initialization

```
LipConfiguration lipConfiguration = new LipConfiguration.Builder()
      .setIsVerifyEmail(false)
      .setAnalyticCallback(null)/* pass IAnalytics to log Analytics info by client app*/
      .setLoggerCallback(null)  /* pass ILogger to log info and error by client app*/
      .setServerUrl("https://xxxx.xxx.logi.com/")
      .build();
LIPSdk.initialize(this.getApplication(),lipConfiguration);
```

Note: Initialize in application onCreate()

Check User logged in and Token validity

```
// Auto Refresh when Token Expired and return fresh token
AccountManager.getCurrentAccountToken(new ResponseListener<AccountToken>() {
                 @Override
                 public void onSuccess(AccountToken result) {
                 }

                 @Override
                 public void onError(ErrorCode errorCode, String errorMessage) {
})
```

```
// Fetch / Auto refresh Token when Expired based on autoRefresh flag
Boolean autoRefresh = false;
AccountManager.getCurrentAccountToken(autoRefresh, new ResponseListener<AccountToken>() {
                @Override
                public void onSuccess(AccountToken result) {

                }

                @Override
                public void onError(ErrorCode errorCode, String errorMessage) {

                      }
             });
```

Request Login or create account and register for callback

```
public class LoginCallbackListener implements LoginListener {
   @Override
   public void onLoginSuccess(UserInfo userInfo, AccountToken token) {
      Log.d(TAG , "onLoginSuccess AccessToken=" + token.getAccessToken());
   }

   @Override
   public void onLoginError(int errorCode, String errorMessage) {
      Log.d(TAG , "onLoginError errorMessage=" + errorMessage);
   }
}

LoginOptions loginOptions = new LoginOptions.Builder()
                      .setIsCreate(true)
                      .build();

LoginController instance = LoginController.getInstance();

instance.registerLoginListener(new CallbackListener())

instance.requestLogin(MainActivity.this, loginOptions);
```

Note: LoginController is a single point of contact for an application and to avoid Activity/Fragment leaks
In case the callback object holds activity/fragment reference then it must be unregistered to avoid memory leaks

UnRegister LoginListener when used in Activity/Fragment scope

LoginController designed to avoid context leaks, In case used with Activity/Fragment must be unregistered in onDestroy()

LoginController instance = LoginController.getInstance();
instance.registerLoginListener(null)

Account Token tracker when it refreshed / changed

Account Token tracker notifies when there is a change in token. This is helps to do background job with updated tokens

```
private class TokenTracker extends AccountTokenTracker {
  @Override
  protected void onAccountTokenChanged(AccountToken accountToken) {
                Log.d(TAG , "onAccountTokenChanged accountToken " +
        accountToken.getAccessToken());
  }
}
```

Token tracker must be started to receive data and stopped when out of scope else it will leak

```
Ex:     onCreate() {
                TokenTracker tracker = new TokenTracker();
                tracker.startTracking();
        }

        onDestroy() {
                tracker.stopTracking();
        }
```