Document Revision History

Revision	Date	Author	Description
0.1	2016-07-21	jojo	初版

SDK Revision History

Revision	Date	Modified
1.0.0	2016-07-23	初版
1.1.0	2016-09-01	新增 sdk 斷線重連更新 token 機制
1.1.1	2016-09-06	改 loadHistory 為每次 修10 筆
1.3.0	2016-10-06	*移除斷線重連自動更新token *修改sharedpreference name 為 “org_iii_sdk_im_prefs” *加入 setSDKTimestamp 可讓user自行設定離線訊息起始時間

Development Environment

平台	Android
開發工具	Android Studio 2.1.1 以上
SDK 版本	ShoaitSDKIM.android.v1.3.0.zip
API DOC 版本	ShoaitSDKIM.android.javadoc.v1.3.0.zip
Demo App 版本	ShoaitSDKIM.android.demo.v1.3.0.zip

Library Dependency	開發 app 需要在 app 的 build.gradle 中加入此Library
	
	dependencies {
	  compile 'com.koushikdutta.ion:ion:2.+'
	}

Demo App 使用

1. 修改程式欄位
    a. MainApplication.java 檔案中, 修改 mAppId , 此 id 為 app 使用的 id

```
private final String mAppId = "testapi"; // 測試用 App Id, 僅供測試, 正式使用請申請app id
```

    b. MainActivity.java 檔案中, 修改 mChannelId , 此 channel id 為公共聊天室 id

```
private final String mChannelId = "-KOtRHiOZG5iNVOfhflf"; // 測試用 Channel Id
```

    c. MainActivity.java 檔案中, 修改 mUserId, 此 userl id 為 IM server 登入的 id

```
private final String mUserId = "demo1@shoait.com"; // 測試用帳號, 僅供測試, 正式使用請申請app id
```

    d. MainActivity.java 檔案中, 修改 mPwd, 此密碼為 IM server 登入的 密碼

```
 private final String mPwd = "12345678"; // 測試用密碼
```

Shoait IM SDK Tutorial

SDK 初始化

```
ShoaitIM mIM= ShoaitIM.getInstance();
mIM.init(context, "appId");
```

登入 IM Server

```
請於網路連線成功後, 進行 login 動作, 並在每次網路斷線重連時, 重新 login IM server

mIM.login("email1@shoait.com", "password", new SCallback<SError, SAuthData>() {

            @Override
            public void onCb(SError err, SAuthData data) {
                if (err != null) return;
                mUid = data.getUid(); // 取得登入 user id
            }
});
```

送訊息

```
對 "user id" 送一則文字(SMessageType.TEXT)訊息

SMessage message = new SMessage("user id", "hello");
mIM.send(message);
```

```
對聊天室 channel 送一則文字(SMessageType.TEXT)訊息

SChannel channel = mIM.getChannel(channelId);
SMessage message = new SMessage(channel, SMessageType.TEXT, "hello");
mIM.send(message);
```

```
送一則圖片(SMessageType.IMAGE)/影片(SMessageType.VIDEO)/檔案訊息(SMessageType.FILE)

File file = new File("path");

mIM.uploadFile(file, new SCallback<SError, String>() {
       @Override
       public void onCb(SError err, String dataUrl) {
```

```
                //上傳完成, 傳回上傳資料的網址 dataUrl
                //將 dataUrl 包成對應訊息送出
                SMessage message = new SMessage(channel, SMessageType.IMAGE, dataUrl);
                mIM.send(message);


        }
}, new SProgressCallback() {
        @Override
        public void onProgress(final long progress, final long total) {
                //上傳進度
        }
});
```

```
送一則貼圖(SMessageType.STICKER)訊息

SMessage message = new SMessage(channel, SMessageType.STICKER, "stickerId")
```

收訊息, 加入訊息監聽

```
public class ChatActivity extends Activity implements SMessageListener
{
        mIM.addMessageListener(this);

        @Override
        public void onMessage(SMessage message) {
                //處理收訊息的地方
        }
}
```

收訊息狀態, 加入訊息狀態監聽

```
public class ChatActivity extends Activity implements SMessageListener
{
        mIM.addMessageListener(this);

        @Override
        public void onMessageStatus(SMessageStatus status) {
                //處理收訊息狀態的地方
        }
}
```

送訊息狀態

```
String status = "已讀" //可填入想要的訊息狀態

@Override
public void onMessage(SMessage message) {
        //對 message 送出狀態訊息
        mIM.setMessageStatus(status, message);
}
```

移除訊息監聽

```
mIM.removeMessageListener(messageListener);
```

建立聊天室

```
建立聊天室, 會預設 SChannelType 為 SChannelType.PUBLIC (多人聊天室)

//新增聊天室成員 user id
Set<String> uids = new HashSet<>();
uids.add("uid1");
uids.add("uid2");
...

//建立聊天室
SChannel.create(uids, new SCallback<SError, SChannel>() {
            @Override
            public void onCb(SError err, SChannel channel) {
                        //成功傳回聊天室 channel 資料
                }
            }
 });
```

邀請進入聊天室

```
SChannel channel = mIm.getChannel("channel Id");

//邀請聊天室成員 user id
```

```
Set<String> uids = new HashSet<>();
uids.add("uid1");
uids.add("uid2");
...

channel.invite(uids, new SCallback() {
           @Override
           public void onCb(SError err, Map<String, String> data) {
               //成功傳回邀請資訊 data
           }
});
```

踢出聊天室

```
SChannel channel = mIm.getChannel("channel Id");

//選擇踢出聊天室成員
Set<String> uids = new HashSet<>();
uids.add("uid1");
uids.add("uid2");
...

//將成員踢出聊天室
channel.ban(uids, new SCallback() {
            @Override
            public void onCb(SError err, Map<String, String> data) {
               //成功傳回踢出資訊 data
            }
 });
```

加入聊天室

```
SChannel.join("channel Id", new SCallback() {
       @Override
       public void onCb(SError err, Map<String, String> data) {
              //成功傳回加入資訊  data
       }
});
```

離開聊天室

```
SChannel channel = mIm.getChannel("channel Id");

channel.exit(new SCallback() {
       @Override
       public void onCb(SError err, Map<String, String> data) {
```

```
                //成功傳回離開資訊 data
        }
});
```

讀取歷史訊息和狀態

```
loadHistory 會傳回包含此 timestamp 之前的 10 筆 SMessage 歷史訊息, messageList 順序為時間由
舊到新, statusList 中為歷史訊息的狀態訊息, 每一筆歷史訊息可能會有多筆狀態,每筆狀態訊息由
message id 對應是哪一筆歷史訊息的狀態訊息

long timestamp = System.currentTimeMillis();

mIM.loadHistory("channel Id",timestamp , new SHistoryCallback() {
        @Override
        public void onLoadHistory(List<SMessage> messageList,
        List<SMessageStatus> statusList) {
                for(SMessage message : messageList) {
                        //讀取訊息
                        Log.d(TAG, message.toString());
                }

                for(SMessageStatus status : statusList( {
                        //讀取訊息狀態, 每則訊息狀態有 message id ,
                         對應到該訊息 id 的狀態
                        Log.d(TAG, status.toString())
                }
        }
});
```

讀取使用者聊天室(單聊, 群聊)

```
讀取使用者所擁有的所有聊天室

mIM.loadChannels(new SCallback<SError, List<SChannel>>() {
        @Override
        public void onCb(SError err, List<SChannel> channels) {
                for(SChannel channel : channels) {
                        //讀取使用者所有的聊天室
                        Log.d(TAG, channel.toString());
                }
        }
});
```

上傳檔案

```
File file = new File("path");

mIM.uploadFile(file, new SCallback<SError, String>() {
        @Override
        public void onCb(SError err, String data) {
                //上傳完成狀態
        }
}, new SProgressCallback() {
        @Override
        public void onProgress(final long progress, final long total) {
                //上傳進度
        }
});
```

備註說明

```
loadHistory 每次只會傳回 10 筆紀錄

mIM.loadHistory(final String channelId, long timestamp, final SHistoryCallback history)
```

```
uploadFile 目前上傳檔案限制 5M

mIM.uploadFile(...)
```

```
1.使用 getChannel 取得 SChannel 資料, 目前只會記錄約 50 筆使用者名單
2.請勿於 SChannel.create 立刻呼叫此 API 取得 SChannel, 會有資料同步的問題, 請於
SChannel.create 的 callback 取得 SChannel

SChannel channel = mIM.getChannel(channelId);

3.當 create 或 join Channel 後, 立即調用 send 傳送 message 時, 若使用到 getChannel 取得
SChannel 並用 SMessage(SChannel channel, SMessageType type, String message) 建立訊息時,
有時候 getChannel 會是 null 而導致失敗, 建議使用下列方式解決

SMessage msg = new Message();
msg.setChannelId("channelId");
msg.setMessage("hello world");
mIM.send(msg);
```