QQ互联 iOS SDK 接入指南

背景

> QQ互联SDK封装了QQ的登录授权以及大部分OpenAPI，应用只需要修改相应参数，不需要理解验证授权流程，即可快速实现QQ登录功能。

集成

1. 下载并解压SDK: 下载链接
2. 拖拽 **TencentOpenAPI** 文件到Xcode工程内（请勾选 Copy items if needed）；或拷贝SDK文件到工程的物理目录下，选中工程Target -> General -> Linked Frameworks and Libraries -> Add Other 然后找到对应文件添加即可
3. 添加依赖库

   SystemConfiguration.framework

必要的配置

1. 新增一条URL scheme：选中工程Target -> Info -> URLTypes；新的scheme命名为：tencent+appid（ex: tencent123456）
2. 添加白名单：LSApplicationQueriesSchemes新增白名单，详见demo
3. Appdelegate的handleOpenURL代理方法中中添加处理回调的代码

```
[QQApiInterface handleOpenURL:url delegate:(id<QQApiInterfaceDelegate>)[QQApiShareEntry class]];

if (YES == [TencentOAuth CanHandleOpenURL:url])
{
    return [TencentOAuth HandleOpenURL:url];
}
```

API使用说明

初始化SDK

创建TencentOAuth并初始化其appid，demo示例为222222，delegate不能为空。

```
NSString *appid = __TencentDemoAppid_;
_oauth = [[TencentOAuth alloc] initWithAppId:appid andDelegate:self];
```

登录授权

登录时，调用TencentOAuth对象的authorize方法，permissions是应用需要用户授权API列表，详见sdkdef.h

```
NSMutableArray * permissions = [[NSMutableArray alloc] initWithObjects:kOPEN
_PERMISSION_GET_USER_INFO, kOPEN_PERMISSION_GET_SIMPLE_USER_INFO,nil];
//登录授权
[_oauth authorize:permissions inSafari:NO];
```

登录完成后，会调用TencentSessionDelegate中关于登录的协议方法：

```
//登录成功回调
- (void)tencentDidLogin
{
    _labelTitle.text = @"登录完成";

    if (_tencentOAuth.accessToken && 0 != [_tencentOAuth.accessToken length]
)
    {
        //记录登录用户的OpenID、Token以及过期时间
    }
    else
    {
        //登录不成功 没有获取到accesstoken
    }
}

//非网络错误导致登录失败:
-(void)tencentDidNotLogin:(BOOL)cancelled
{
    if (cancelled)
    //用户取消登录
      else
    //登录失败
}


//网络错误导致登录失败:
-(void)tencentDidNotNetWork
{
```

```
    //检查网络设置
}
```

登录成功后，即可获取到access token和openid。accessToken和 openid保存在TencentOAuth对象中，并且已经本地化存储。可以通过相应的属性方法直接获得。

```
[_oauth accessToken] ;
[_oauth openId] ;
[_oauth getCachedOpenID] ;
[_oauth getCachedToken] ;
```

特别提示：

1. 由于登录是异步过程，这里可能会由于用户的行为导致整个登录的的流程无法正常走完，即有可能由于用户行为导致登录完成后不会有任何登录回调被调用。开发者在使用SDK进行开发的时候需要考虑到这点，防止由于一直在同步等待登录的回调而造成应用的卡死，建议在登录的时候将这个实现做成一个异步过程。

2. access token过期后提示用户重新登录授权。

3. 第三方网站可存储access token信息，以便后续调用OpenAPI访问和修改用户信息时使用。如果需要保存授权信息，需要保存登录完成后返回的accessToken，openid 和expirationDate三个数据，下次登录的时候直接将这三个数据是设置到TencentOAuth对象中即可。

   ```
   //获得:
   [_tencentOAuth accessToken] ;
   [_tencentOAuth openId] ;
   [_tencentOAuth expirationDate] ;
   //设置:
   [_tencentOAuth setAccessToken:accessToken] ;
   [_tencentOAuth setOpenId:openId] ;
   [_tencentOAuth setExpirationDate:expirationDate] ;
   ```

4. 建议应用在用户登录后，即调用getUserInfo接口获得该用户的头像、昵称并显示在界面上，使用户体验统一。

增量授权

当第三方应用调用某个API接口时，如果服务器返回操作未被授权，则会触发增量授权逻辑。第三方应用需自行实现tencentNeedPerformIncrAuth:withPermissions:协议接口才能够进入增量授权逻辑，否则默认第三方应用放弃增量授权。在用户通过增量授权页重新授权登录后，第三方应用需更新自己维护的token及有效期限等信息。

这里需要注意的是：用户在增量授权时是可以更换帐号进行登录的，强烈要求第三方应用核对增量授权后的用户openid是否一致，以添加必要的处理逻辑（用户帐号变更需重新拉取用户的资料等信息）

示例如下：

```
- (BOOL)tencentNeedPerformIncrAuth:(TencentOAuth *)tencentOAuth
withPermissions:(NSArray *)permissions
{
// incrAuthWithPermissions是增量授权时需要调用的登录接口
// permissions是需要增量授权的权限列表
[tencentOAuth incrAuthWithPermissions:permissions];
// 返回NO表明不需要再回传未授权API接口的原始请求结果；否则可以返回YES
return NO;

}

//增量授权成功时，会通过tencentDidUpdate:协议接口通知第三方应用：
- (void)tencentDidUpdate:(TencentOAuth *)tencentOAuth
{
    _labelTitle.text = @"增量授权完成";
    if (tencentOAuth.accessToken
        && 0 != [tencentOAuth.accessToken length])
    {
        // 在这里第三方应用需要更新自己维护的token及有效期限等信息
        // **务必在这里检查用户的openid是否有变更，变更需重新拉取用户的资料等信息**
        _labelAccessToken.text = tencentOAuth.accessToken;
    }
    else
    {
        _labelAccessToken.text = @"增量授权不成功，没有获取accesstoken";
    }
}


//增量授权失败时，会通过tencentFailedUpdate:协议接口通知第三方应用：
- (void)tencentFailedUpdate:(UpdateFailType)reason
{
    switch (reason)
    {
```

```
        case kUpdateFailNetwork:
        {
            _labelTitle.text=@"增量授权失败，无网络连接，请设置网络";
            break;
        }
        case kUpdateFailUserCancel:
        {
            _labelTitle.text=@"增量授权失败，用户取消授权";
            break;
        }
        case kUpdateFailUnknown:
        default:
        {
            _labelTitle.text=@"增量授权失败，未知错误";
            break;
        }
    }
}
```

分享

分享主要分为分享到好友，分享到空间两种，同时还支持分享到我的收藏，我的电脑，支持轻应用消息对象的分享（后面几种使用方法详见SDKDemo）。

分享到好友支持以下几种消息类型：
* 纯文本消息（`QQApiTextObject`）
* 纯图片消息（`QQApiImageObject`）
* 链接类消息（`QQApiNewsObject`，`QQApiAudioObject`，`QQApiVideoObject`）

分享到空间支持以下几种消息类型：
* 纯文本消息（`QQApiImageArrayForQZoneObject` 见SDK注释）
* 纯图片消息（`QQApiImageArrayForQZoneObject`）
* 视频类消息（`QQApiVideoForQZoneObject`）
* 链接类消息（`QQApiNewsObject`，`QQApiAudioObject`，`QQApiVideoObject`）

纯文本分享

```
QQApiTextObject *txtObj = [QQApiTextObject objectWithText:@"QQ互联测试"];
SendMessageToQQReq *req = [SendMessageToQQReq reqWithContent:txtObj];
QQApiSendResultCode sent = [QQApiInterface sendReq:req];
```

纯图片分享

```
NSString *imgPath = [[[NSBundle mainBundle] resourcePath] stringByAppendingP
athComponent:@"test.gif"];
NSData *imgData = [NSData dataWithContentsOfFile:imgPath];
QQApiImageObject *imgObj = [QQApiImageObject objectWithData:imgData
                                                    previewImageData:imgData
                                                    title:@"QQ互联测试"
                                                    description:@"QQ互联测试分享"];
SendMessageToQQReq *req = [SendMessageToQQReq reqWithContent:imgObj];
QQApiSendResultCode sent = [QQApiInterface sendReq:req];
```

链接分享-新闻

链接分享的缩略图可以是本地图片也可以是网络图片

```
NSString *utf8String = @"http://www.163.com";
NSString *title = @"新闻标题";
NSString *description = @"新闻描述";
NSString *previewImageUrl = @"http://cdni.wired.co.uk/620x413/k_n/NewsForeca
st%20copy_620x413.jpg";

QQApiNewsObject *newsObj = [QQApiNewsObject objectWithURL:[NSURL URLWithStri
ng:utf8String]
                                                        title:title descript
ion:description
                                                    previewImageURL:[NSURL URLWith
String:previewImageUrl]];
SendMessageToQQReq *req = [SendMessageToQQReq reqWithContent:newsObj];

QQApiSendResultCode sent = [QQApiInterface SendReqToQZone:req];
```

链接分享-视频

链接分享的缩略图可以是本地图片也可以是网络图片

```
NSString *previewPath = [[[NSBundle mainBundle] resourcePath] stringByAppend
ingPathComponent:@"video.jpg"];
NSData* previewData = [NSData dataWithContentsOfFile:previewPath];
NSString *utf8String = @"http://www.163.com";
QQApiVideoObject *videoObj = [QQApiVideoObject objectWithURL:[NSURL URLWithS
tring:utf8String ? : @""]
                                                        title:@"QQ互联测试
"
                                                    description:@"QQ互联测试
分享"
                                                    previewImageData:previewData
];
```

```
    [videoObj setFlashURL:[NSURL URLWithString:@"http://v.qq.com/cover/5/53x
6bbyb07ebl3s/n0013r8esy6.html"]];
SendMessageToQQReq *req = [SendMessageToQQReq reqWithContent:videoObj];

QQApiSendResultCode sent = [QQApiInterface SendReqToQZone:req];
```

其他特性

UnionID

UnionID介绍：
此接口用于获取用户个人信息。开发者可通过OpenID来获取用户基本信息。特别需要注意的是，如果开发者拥有多个移动应用、网站应用，可通过获取用户的unionid来区分用户的唯一性，因为只要是同 一个QQ互联平台帐号下的移动应用、网站应用，用户的unionid是唯一的。换句话说，同一用户，对 同一个QQ互联平台下的不同应用，unionid是相同的。（Unionid机制暂未对外开放，开发者只能通过 申请获得权限，后续会开放给所有开发者）

UnionID的获取：

```
//登陆成功过后调用
[_oauth RequestUnionId];
```

回调：

```
//如果获取成功会在TencentLoginDelegate的didGetUnionID回调中回调，最后在oauth实例中
获取
[oauth unionid];
```

轻应用消息对象分享能力

分享流程如下：
A: 创建QQApiNewsObject(结构化消息类型的object，这里推荐使用news类型)
B: 生成ArkJSONString 和 创建ArkObject
C: 创建SendMessageToQQReq
D: QQInterface sendReq
如果ark分享失败会分享原来的分享消息类型到手Q，**使用实例详见SDKDemo（关键字：ArkObject）**

轻应用消息对象（JSON）格式定义及示例

app：

轻应用ID，本分享消息将使用哪个应用进行打开，必须是当前互联id对应的轻应用。

view：

视图ID，指定需要展示的轻应用具体视图名称，必须是本应用有效的视图id

meta：

数据对象，打开指定view时候传递的参数，view创建成功后，将通过对应view的lua脚本OnSetMetadata函数传递。

例一：

```
{
"app": "com.tencent.sample",
"view": "sample",
"meta": {}
}
```

这条消息将用id为com.tencent.sample的应用的id为sample的view打开，不传递任何参数

例二：

```
{
"app": "com.tencent.sample",
"view": "sample",
"meta": {
"data": "value"
}
}
```

这条消息将用id为com.tencent.sample的应用的id为sample的view打开，运行时通过OnSetMetadata传递meta对象，脚本可以通过参数用"data"作key获得value为"value "的值。

注意事项

- 以前申请过 APPID 类似“QQXXXXXXXX”的开发商，建议重新申请。为了兼容旧版本的手机QQ，需要增加 URL Scheme，QQ + 十六进制新AppId，不足八位在首部补0。（如 appid=222222 则 scheme=QQ0003640E）
- SDK目前已经支持QQ、TIM授权登录及分享功能， 会按照QQ>TIM的顺序进行调用。只要用户安装了QQ、TIM中任意一个应用，都可为第三方应用进行授权登录、分享功能。第三方应用在接入SDK时不需要判断是否安装QQ、TIM。若有判断安装QQ、TIM的逻辑建议移除。
- 登录 || 分享失败？请戳错误码说明

更新说明

SDK_V3.3.3

1. 未安装手机QQ的情况下，支持扫码登录
2. 修复bug