## iOS\_SDK\_API\_使用说明

**1. SSO安全登录QQ帐号**

iOS SDK支持应用跳转到手机QQ进行登录，给用户提供更加安全、快捷的体验 。如果用户没有安装手机QQ，将显示web页面提示未安装手机QQ。

**2. 分享到QQ和QQ空间**

**2.1 分享到QQ好友**

分享到QQ好友支持发送：

* – 纯文本消息（QQApiTextObject）
* – 纯图片消息（QQApiImageObject）
* – 新闻类消息(QQApiNewsObject)
* – 音频类消息(QQApiAudioObject)
* – 视频类消息(QQApiVideoObject)

在用户安装了手机QQ时通过手机QQ进行分享，否则调用浏览器页面进行分享。  
其中文本消息，图文消息和音频消息的title是必须的，summary可以不填，具体调用请参考[2.3 分享示例代码](http://wiki.connect.qq.com/ios_sandbox1#2.3.E5.88.86.E4.BA.AB.E7.A4.BA.E4.BE.8B.E4.BB.A3.E7.A0.81)。  
使用分享到QQ好友功能需要设置QQ业务回调，请参考[4. 处理QQ业务的回调](http://wiki.connect.qq.com/ios_sandbox1#6..E5.A4.84.E7.90.86QQ.E4.B8.9A.E5.8A.A1.E7.9A.84.E5.9B.9E.E8.B0.83)。

**2.2 分享到QQ空间**  
持分享链接、纯 本、图 、本地视频到空间。 需要在 程的plist 件中 LSApplicationQueriesSchemes 字段中添加 mqqop ensdkapiV4

分享到QQ空间支持发送：

QQApiImageArrayForQZoneObject

QQApiVideoForQZoneObject  
带有URL的消息：  
– 新闻类消息(QQApiNewsObject)  
– 音频类消息(QQApiAudioObject)  
– 视频类消息(QQApiVideoObject)  
使用分享到QQ空间功能需要设置QQ业务回调，请参考[4. 处理QQ业务的回调](http://wiki.connect.qq.com/ios_sandbox1#6..E5.A4.84.E7.90.86QQ.E4.B8.9A.E5.8A.A1.E7.9A.84.E5.9B.9E.E8.B0.83)。

**2.3 分享示例代码**

下面是各种分享消息的实例代码，作为开发者调用QQ好友分享和QQ空间分享的参考：  
**纯文本分享:**

|  |  |
| --- | --- |
| 1  2  3  4 | QQApiTextObject \*txtObj = [QQApiTextObject objectWithText:@"QQ互联测试"];  SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:txtObj];  //将内容分享到qq  QQApiSendResultCode sent = [QQApiInterface sendReq:req]; |

**纯图片分享:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | NSString \*imgPath = [[[NSBundle mainBundle] resourcePath] stringByAppendingPathComponent:@"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];  //将内容分享到qq  QQApiSendResultCode sent = [QQApiInterface sendReq:req]; |

新闻分享:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 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/NewsForecast%20copy\_620x413.jpg";  QQApiNewsObject \*newsObj = [QQApiNewsObject  objectWithURL:[NSURL URLWithString:utf8String]  title:title  description:description  previewImageURL:[NSURL URLWithString:previewImageUrl]];  SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:newsObj];  //将内容分享到qq  //QQApiSendResultCode sent = [QQApiInterface sendReq:req];  //将内容分享到qzone  QQApiSendResultCode sent = [QQApiInterface SendReqToQZone:req]; |

**音乐分享:**

**视频分享:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | NSString \*previewPath = [[[NSBundle mainBundle] resourcePath] stringByAppendingPathComponent:@"video.jpg"];  NSData\* previewData = [NSData dataWithContentsOfFile:previewPath];  NSString \*utf8String = @"http://www.163.com";  QQApiVideoObject \*videoObj = [QQApiVideoObject objectWithURL:[NSURL URLWithString: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/5/53x6bbyb07ebl3s/n0013r8esy6.html"]];  SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:videoObj];  //将内容分享到qq  //QQApiSendResultCode sent = [QQApiInterface sendReq:req];  //将被容分享到qzone  QQApiSendResultCode sent = [QQApiInterface SendReqToQZone:req]; |

**3. 调用OpenAPI**

SDK中具体支持的API种类和每条API的参数说明，请参照[API列表](http://wiki.connect.qq.com/api%e5%88%97%e8%a1%a8)。这里用设置用户头像举例说明。

**3.1 OpenAPI参数字典封装**

在封装各接口的参数字典时，推荐使用为每个接口新增的参数封装辅助类，如：  
接口(BOOL)addShareWithParams:(NSMutableDictionary \*)params  
对应辅助类TCAddShareDic

TCAddShareDic辅助类中属性：  
@property (nonatomic, retain) TCRequiredStr paramTitle;  
对应于CGI请求中参数”title”

TCRequiredStr 表示这是一个必填参数，类型是字符串  
TCOptionalStr 表示这是一个可选参数，类型是字符串

**3.2 使用增量授权**

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。示例如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | - (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];  return NO; // 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  } |

**注意：**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。  
\*\*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）\*\*  
增量授权成功时，会通过tencentDidUpdate:协议接口通知第三方应用：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | - (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：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | - (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;          }      }  } |

**3.3 返回数据说明**

APIResponse属性：  
retCode – 网络请求返回码，主要表示服务器是否成功返回数据  
seq – 请求的序列号，依次递增，方便内部管理  
errorMsg – 错误消息  
jsonResponse – 由服务器返回的json格式字符串转换而来的json字典数据（具体参数字段请参见对应API说明文档）  
message – 服务器返回的原始字符串数据  
detailRetCode – 新增的详细错误码，以区分不同的错误原因（v1.2以及之前的SDK接口无此参数）

**3.4 返回码说明**

retCode网络请求返回码说明：  
0 表示成功，请求成功发送到服务器，并且服务器返回的数据格式正确  
1 表示失败，可能原因有网络异常，或服务器返回的数据格式错误，无法解析

detailRetCode详细错误码说明：  
kOpenSDKInvalid -无效的错误码

**[公共错误码]**  
kOpenSDKErrorSuccess – 成功  
kOpenSDKErrorUnknown – 未知错误  
kOpenSDKErrorUserCancel – 用户取消  
kOpenSDKErrorReLogin – token无效或用户未授权相应权限需要重新登录  
kOpenSDKErrorOperationDeny – 第三方应用没有该api操作的权限

**[网络相关错误码]**  
kOpenSDKErrorNetwork – 网络错误，网络不通或连接不到服务器  
kOpenSDKErrorURL – URL格式或协议错误  
kOpenSDKErrorDataParse – 数据解析错误，服务器返回的数据解析出错  
kOpenSDKErrorParam – 传入参数错误  
kOpenSDKErrorConnTimeout – http连接超时  
kOpenSDKErrorSecurity – 安全问题  
kOpenSDKErrorIO – 下载和文件IO错误  
kOpenSDKErrorServer – 服务器端错误

**[webview中特有错误]**  
kOpenSDKErrorWebPage – 页面错误

**[设置头像 自定义错误码段]**  
kOpenSDKErrorUserHeadPicLarge – 图片过大 设置头像自定义错误码

**4. 处理QQ业务的回调**

在使用QQApiInterface 的方法时需要设置回调才能正确调用。设置方法如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | - (BOOL)application:(UIApplication \*)application handleOpenURL:(NSURL \*)url  {  #if \_\_QQAPI\_ENABLE\_\_  [QQApiInterface handleOpenURL:url delegate:(id)[QQAPIDemoEntry class]];  #endif  if (YES == [TencentOAuth CanHandleOpenURL:url])  {  return [TencentOAuth HandleOpenURL:url];  }  return YES;  } |

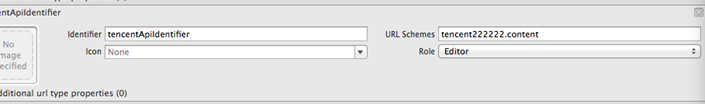
在handleOpenURL 中添加[QQApiInterface handleOpenURL:url delegate:(id)[QQAPIDemoEntry class]]代码，可以在QQAPIDemoEntry类中实现QQApiInterfaceDelegate的回调方法。更完整的示例请参考SDKDemo。

**5. iOS SDK 腾讯平台开放能力**

iOS SDK支持第三方应用成为腾讯业务内容的提供商，要想成为腾讯业务内容提供商需要[在QQ互联的官网进行注册](http://connect.qq.com/)。  
使用SDK响应腾讯业务发起的内容请求可以参考下面的操作：

**5.1 修改工程配置文件**

在XCode中，选择你的工程设置项，选中“TARGETS”一栏，在“info”标签栏的“URL type”添加一条新的“URL scheme”，新的scheme = tencent + appid+‘.’+ content，identifier“tencentApiIdentifier”为。如果您使用的是XCode3或者更低的版本，则需要在plist文件中添加。Demo中我们注册的appid是222222。如下图



**5.2 重写AppDelegate 的handleOpenURL和openURL方法**

openURL:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | - (BOOL)application:(UIApplication \*)application openURL:(NSURL \*)url sourceApplication:(NSString \*)sourceApplication annotation:(id)annotation  {      if (YES == [TencentApiInterface canOpenURL:url delegate:self])      {          [TencentApiInterface handleOpenURL:url delegate:self];      }  } |

handleOpenURL

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | - (BOOL)application:(UIApplication \*)application handleOpenURL:(NSURL \*)url  {      if (YES == [TencentApiInterface canOpenURL:url delegate:self])      {          [TencentApiInterface handleOpenURL:url delegate:self];      }  } |

**5.3 实现TencentApiInterfaceDelegate协议**

响应腾讯业务的请求，需要实现TencentApiInterfaceDelegate协议的两个方法:

|  |  |
| --- | --- |
| 1  2  3  4 | @protocol TencentApiInterfaceDelegate  //onTencentReq是腾讯业务向第三方应用主动发起请求，要求第三方应用响应提供内容，第三方应用再响应完成后需要调用TencentApiInterface的sendRespMessageToTencentApp返回腾讯业务。  - (BOOL)onTencentReq(TencentApiReq \*)req;  //onTencentResp是第三方应用主动向腾讯业务发起请求后，腾讯业务返回请求结果。 |

可以参考以下代码实现对腾讯业务的请求的响应，或参考我们的sample中的代码实现

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | - (BOOL)onTencentReq:(TencentApiReq \*)req  {      NSArray \*array = [req arrMessage];      for (id obj in array)      {          if ([obj isKindOfClass:[TencentTextMessageObjV1 class]])          {              obj = (TencentTextMessageObjV1 \*)obj;              [obj setSText:@"test"];          }          if ([obj isKindOfClass:[TencentImageMessageObjV1 class]])          {              obj = (TencentImageMessageObjV1 \*)obj;              NSString \*path = [NSString stringWithFormat:@"%@/qzone0.jpg",              [[NSBundle mainBundle] resourcePath]];              UIImage \*image = [[UIImage alloc] initWithContentsOfFile:path];              NSData \*data = UIImageJPEGRepresentation(image, 1.0f);              [obj setDataImage:data];          }          if ([obj isKindOfClass:[TencentVideoMessageV1 class]])          {              //请加入一段视频URL              obj = (TencentVideoMessageV1 \*)obj;              [obj setSUrl:@" http://www.tudou.com/programs/view/\_cVM3aAp270/"];          }      }      TencentApiResp \*resp = [TencentApiResp respFromReq:req];      [TencentOAuth sendRespMessageToTencentApp:resp];  } |

**5.4 新增特性**

1·本次新增加UnionID的获取接口

**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：**

//获取

BOOL bRet = [[[sdkCall getinstance] oauth] RequestUnionId];  
if (!bRet) {  
UIAlertView \*alertView = [[UIAlertView alloc] initWithTitle:@"提示" message:@"信息不足，请先登录。" delegate:nil cancelButtonTitle:@"好的" otherButtonTitles:nil, nil];  
[alertView show];  
\_\_RELEASE(alertView);  
}

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

NSString \*result = [NSString stringWithFormat:@"获取成功 appid = %@,\n openid = %@,\n unionid = %@",  
[[sdkCall getinstance].oauth appId],  
[sdkCall getinstance].oauth.openId,  
[sdkCall getinstance].oauth.unionid];

2· 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: 创建ArkObject

C: 创建SendMessageToQQReq

D: QQInterface sendReq

NSString \*jsonString = @"见轻应用消息对象（JSON）格式定义";  
QQApiNewsObject \*newsObj = [QQApiNewsObject objectWithURL:[NSURL URLWithString:utf8String ? : @""]  
title:self.binding\_title ? : @""  
description:self.binding\_description ? : @""  
previewImageURL:[NSURL URLWithString:self.binding\_previewImageUrl ? : @""]];  
ArkObject \*arkObj = [ArkObject objectWithData:jsonString qqApiObject:newsObj];  
SendMessageToQQReq \*req = [SendMessageToQQReq reqWithArkContent:arkObj];  
QQApiSendResultCode ret = [QQApiInterface sendReq:req];

特别注意：如果ark分享失败会分享原来的分享消息类型到手Q。

**轻应用消息对象（JSON）格式定义**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 类型 | 描述 | 备注 |
| app | string | 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 | string | 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 | object | 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：**

**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。**