**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，将打开登录页面。

**2. 分享到QQ和QQ空间**

**2.1 分享到QQ好友**

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[6. 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**

分享到QQ空间的接口用于取代老的分享接口addShareWithParams（该接口已经废弃）。

在用户安装了手机QQ（4.6版本以上）时通过手机QQ中的QZone结合版进行分享，否则调用浏览器页面进行分享。  
分享时调用浏览器页面进行分享。其中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空间功能需要设置QQ业务回调，请参考[6. 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)。

在分享到QQ好友和QQ空间的时候，根据是本地分享还是浏览器中的分享，支持分享的消息类型不同。

因为webQQ好友分享和web QQ空间的分享都不支持非URL类型的分享，所以这里建议在分享到QQ好友或者QQ空间的时候尽量避免这两种类型的调用，避免发生不支持的错误。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 分享消息类型 | QQ好友 | QQ空间 | web QQ好友 | web QQ空间 |
| QQApiTextObject | 支持 | 不支持 | 不支持 | 不支持 |
| QQApiImageObject | 支持 | 不支持 | 不支持 | 不支持 |
| QQApiNewsObject | 支持 | 支持 | 支持 | 支持 |
| QQApiImageObject | 支持 | 支持 | 支持 | 支持 |
| QQApiVideoObject | 支持 | 支持 | 支持 | 支持 |

**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:**

//开发者分享的文本内容

QQApiTextObject \*txtObj = [QQApiTextObject objectWithText:@"text"];

SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:txtObj];

//将内容分享到qq

QQApiSendResultCode sent = [QQApiInterface sendReq:req];

**纯图片分享:**

//开发者分享图片数据

NSData \*imgData = [NSData dataWithContentsOfFile:path];

//

QQApiImageObject \*imgObj = [QQApiImageObject objectWithData:imgData

previewImageData:imgData

title:@"title"

description:@"description"];

SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:imgObj];

//将内容分享到qq

QQApiSendResultCode sent = [QQApiInterface sendReq:req];

**新闻分享:**

//分享跳转URL

NSString \*url = @"http://xxx.xxx.xxx/";

//分享图预览图URL地址

NSString \*previewImageUrl = @"preImageUrl.png";

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:**

//分享跳转URL

NSString \*url = @"http://xxx.xxx.xxx/";

//分享图预览图URL地址

NSString \*previewImageUrl = @"preImageUrl.png";

//音乐播放的网络流媒体地址

NSString \*flashURL = @"xxx.mp3 ";

QQApiAudioObject \*audioObj =[QQApiAudioObject objectWithURL:[NSURL URLWithString:url]

title:@"title" description:@"description" previewImageURL:[NSURL URLWithString:previewImageUrl]];

//设置播放流媒体地址

[audioObj setFlashUrl:flashURL];

SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:audioObj]

//将内容分享到qq

//QQApiSendResultCode sent = [QQApiInterface sendReq:req];

//将被容分享到qzone

QQApiSendResultCode sent = [QQApiInterface SendReqToQZone:req];

**注意:**  
分享到QQ空间接口暂时不支持发送多张图片的能力，若开发者传入多张图片，则会自动选入第一张图片作为预览图。多图的能力将在以后支持。

**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 设置用户头像调用示例**

设置QQ头像时，调用TencetnOAuth对象的setUserHeadpic方法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | TCSetUserHeadpic \*params = [TCSetUserHeadpic dictionary];  params.paramImage = image;  params.paramFileName = @"make";  UIViewController \*headController = nil;  [\_tencentOAuth setUserHeadpic:params andViewController:&headController];  UIViewController \*rootController = [[[app delegate] window] rootViewController];  [rootController dismissModalViewControllerAnimated:NO];  [rootController presentModalViewController:headController animated:YES]; |

设置头像完成后，会调用TencentSessionDelegate中的tencentOAuth:doCloseViewController通知应用界面需要关闭:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | @protocol TencentSessionDelegate  - (void)setUserHeadpicResponse:(APIResponse\*) response  - (void)tencentOAuth:(TencentOAuth \*)tencentOAuth doCloseViewController:(UIViewController \*)viewController  {  if (tencentOAuth == \_tencentOAuth)  {  UIApplication \*app = [UIApplication sharedApplication];  UIViewController \*rootController = [[[app delegate] window] rootViewController];  [rootController dismissModalViewControllerAnimated:YES];  }  } |

设置头像完成后，会调用TencentSessionDelegate中的setUserHeadpicResponse返回调用结果：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | @protocol TencentSessionDelegate  - (void)setUserHeadpicResponse:(APIResponse\*) response  {      if (nil == response)      {          return;      }      if (URLREQUEST\_FAILED == response.retCode          && kOpenSDKErrorUserHeadPicLarge == response.detailRetCode)      {          UIAlertView \*alert = [[UIAlertView alloc] initWithTitle:@"操作失败" message:[NSString stringWithFormat:@"您的图片大小超标啦，请更换一张试试呢:)"]          delegate:self cancelButtonTitle:@"我知道啦" otherButtonTitles: nil];          [alert show];          [alert release];      }  } |

**3.3 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;          }      }  } |

**3.4 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）

**3.5 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. 调用微云接口**

iOS SDK支持调用腾讯微云的接口，支持对照片，音频，视频，结构化数据的上传，下载，删除等能力。下面以微云照片上传为例，向开发者介绍如何向腾讯微云上传照片。

**4.1创建微云request对象，调用TencentOAuth实例的统一发送接口**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | - (void)weiyunUpload:(NSNumber \*)weiyunType  {      WeiYun\_upload\_photo\_GET \*request = [[WeiYun\_upload\_photo\_GET alloc] init];      //data是图片的二进制数据流      request.param\_sha = [data digest];      request.param\_md5 = [data md5];      request.param\_name = @"test";      request.param\_size = [NSString stringWithFormat:@"%u", [data length]];      request.param\_upload\_type = @"control";      request.paramUploadData = data;      [\_tencentOAuth sendAPIRequest:request callback:self];  } |

**4.2 实现TCAPIRequestDelegate协议，响应结果**

-(void)cgiRquest:didResponse用来响应统一api的调用结果，request是请求的消息，response是响应的结果。

示例:

- (void)cgiRequest:(TCAPIRequest \*)request didResponse:(APIResponse \*)response

{

if (URLREQUEST\_SUCCEED == response.retCode && kOpenSDKErrorSuccess == response.detailRetCode)

{

NSString \*str = [self responseDataProcess:request didResponse:response];

UIAlertView \*alert = [[UIAlertView alloc] initWithTitle:@"操作成功"

message:[NSString stringWithFormat:@"%@",str]

delegate:self cancelButtonTitle:@"我知道啦"

otherButtonTitles:nil];

[alert show];

｝

｝

**注意:**  
上传图片是两步的过程。需要先获得上传参数，再根据上传参数拼装HTTP协议上传到微云的服务器。两步是分开进行的，所以需要注意两个问题。  
1.第三方需要将上传对象进行保存，否则将无法进行默认的大数据上传。  
2.iOS SDK 1.8实现了拉取上传参数和数据的上传。但是上传没有实现断点续传，所以对大数据的上传支持有限，可以通过实现 TCAPIRequestUploadDelegate 和 TCAPIRequestDownloadDelegate 来取消内部的自动上传和下载的逻辑，获得上传下载参数后自己拼装请求参数。  
TCAPIRequestUploadDelegate 协议：

|  |  |
| --- | --- |
| 1 | - (BOOL)cgiUploadRequest:(TCAPIRequest \*)uploadRequest shouldBeginUploadingStorageRequest:(NSURLRequest \*)storageRequest; |

TCAPIRequestDownloadDelegate 协议：

|  |  |
| --- | --- |
| 1 | - (BOOL)cgiDownloadRequest:(TCAPIRequest \*)downloadRequest shouldBeginDownloadingStorageRequest:(NSURLRequest \*)storageRequest; |

如果不实现上传、下载协议，则按默认方式进行数据的上传下载。第三方实现协议若返回YES，则默认进行数据的上传下载。返回NO则停止上传下载。由第三方获取当前数据后自己上传下载数据。

其他API统一接口的调用参照上传照片的调用方式。目前统一接口主要支持微云所有API接口的调用和微博上传带有图片微博的调用，具体可以参考WeiyunAPI.h和WeiboAPI.h中有关微云和微博TCAPIRequest对象的定义。

**5. WPA临时会话**

iOS SDK支持发起QQ临时会话，获取指定QQ帐号在线状态。使用WPA功能需要设置QQ业务回调，请参考[6. 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)。

**5.1 发起QQ临时会话**

下面是向指定QQ号码发起临时会话的示例代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | - (void)onOpenWPA:(QElement \*)sender  {  [self.view endEditing:YES];  [self.root fetchValueUsingBindingsIntoObject:self];  QQApiWPAObject \*wpaObj = [QQApiWPAObject objectWithUin:self.binding\_uin];  SendMessageToQQReq \*req = [SendMessageToQQReq reqWithContent:wpaObj];  QQApiSendResultCode sent = [QQApiInterface sendReq:req];  [self handleSendResult:sent];  } |

**5.2 获取指定QQ号码的在线状态**

下面是获取指定QQ号码在线状态的示例代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | - (void)getQQUinOnlineStatues:(QElement \*)sender  {  [self.view endEditing:YES];  [self.root fetchValueUsingBindingsIntoObject:self];  NSArray \*ARR = [NSArray arrayWithObjects:self.binding\_uin, nil];  [QQApiInterface getQQUinOnlineStatues:ARR delegate:self];  ｝ |

**6. 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：

- (BOOL)application:(UIApplication \*)application handleOpenURL:(NSURL \*)url

{

[QQApiInterface handleOpenURL:url delegate:qqApiDelegate];

return YES;

}

在handleOpenURL 中添加[QQApiInterface handleOpenURL:url delegate: qqApiDelegate]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。