QQ音乐OpenID接入指南

> OpenID授权方案可以将QQ音乐的登录态授权给合作方App，合作方App拿到授权后，可以访问QQ音乐的内容

修改记录

版本	日期	描述
V1.0	2018/12/28	OpenID接入指南

一. 业务接入流程

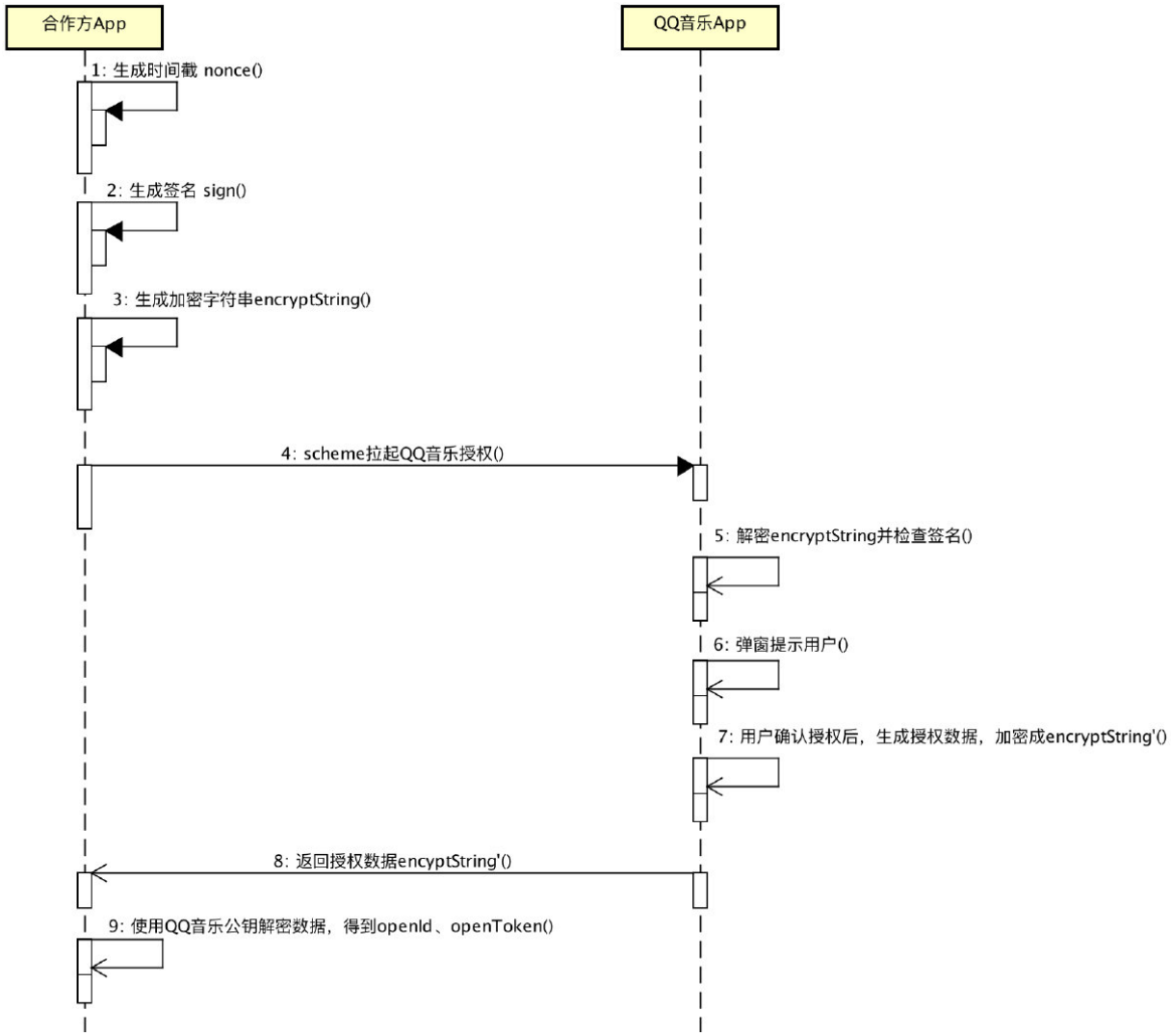
1. 合作方在接入前，先自行生成RSA公私钥
    - 合作方公钥 `AppPubKey` ,注册时提供给QQ音乐
    - 合作方私钥 `AppPrivateKey` ，合作方自己保管
    - 生成密钥参考 **6.3 RSA公私钥生成说明**
2. 将业务信息提交给QQ音乐进行注册。其中业务信息包含
    - 业务名称 `AppName`
    - 公司名称 `Company`
    - 程序包名 `PackageName`
    - 合作方公钥 `AppPubKey`
    - 合作方Logo `AppIcon`
3. 注册完成后，QQ音乐将向合作方提供以下信息：
    - 业务ID `AppId`
    - QQ音乐公钥 `QQMusicPubKey`
    - `OpenAPI AppId`
    - `OpenAPI AppKey`
    - `OpenAPI app_private_key`

    具体参数含义参考： **6.1 参数说明**

二. iOS接入指南

2.1 iOS授权流程说明

流程时序图



步骤说明

1. 业务方App，使用当前unix时间截作为随机字符串 `nonce`
2. 使用业务方私钥 `AppPrivateKey` 对nonce进行RSA签名，生成 `sign`
3. 将 `nonce` 、 `sign` 、 `callbackUrl` 组装成json

```
{
    "nonce":nonce,
    "sign":sign,
    "callbackUrl":"openiddemo://"
}
```

使用QQ音乐公钥 `QQMusicPubKey` 对json进行加密，得到 `encryptString`

4. 通过scheme方式拉起QQ音乐，其中scheme url规则如下

将 `cmd` 、 `appId` 、 `encryptString` 、 `callbackUrl` 组成json

```
{
    "cmd":auth,
    "appId":appId,
    "encryptString":encryptString,
    "callbackUrl":"openiddemo://"
}
```

将json使用urlencode编码得到字符串 `param`

生成完整scheme url:

`qqmusic://qq.com/other/openid?p=param`

5. QQ音乐会对scheme传过来的encryptString使用QQ音乐私钥进行解密，并使用合作方公钥 `AppPubKey` 对签名 `sign` 进行验证
6. 验证通过后，QQ音乐App会弹出授权框让用户进行确认
7. 用户确认授权后，QQ音乐会将 `nonce` 、 `sign` 、 `openId` 、 `openToken` 、 `expireTime` 组成json

```
{
    "nonce":auth,
    "sign":appId,
    "openId":18762394837293,
    "openToken":"2sxSws1EbEhiXYRfFImI9ZCQt8a6rWFbg",
    "expireTime":1545994007    //token过期时间，到期后需要重新授权
}
```

QQ音乐会使用合作方公钥 `AppPubKey` 对上述json进行加密，生成 `encryptString'`

8. 通过scheme方式将授权结果返回合作方App，回调scheme url规则如下

将 `encryptString'` 、 `ret` 组成json

```
{
    "ret":ret,//结果码，其中0：授权成功  -1：授权失败 -2：用户取消
    "encryptString":encryptString //如果授权成功则返回加密串
}
```

将json使用urlencode编码得到字符串 `param`

生成前面的 `callbackUrl` 和 `param` 完整scheme url:

`callbackUrl?p=param`

9. 合作方App根据结果码 `ret` 判断是否授权成功，成功则使用合作方私钥 `AppPrivateKey` 进行解密，最终得到 `nonce` 、 `sign` 、 `openId` 、 `openToken` 、 `expireTime` 。合作方使用QQ音乐公钥 `QQMusicPubKey` 对sign进行签名验证，以确保是来自QQ音乐的数据。

2.2 开发集成流程

> 为了方便合作方快速接入，iOS提供SDK和Demo两个源码，在SDK中封装了加解密以及签名的过程。
>
> 合作方可以在SDK源码基础上修改接入。

1. 集成SDK

   引用 `libQQMusicOpenIDSDK.a` `QQMusicOpenSDK.h` 两个文件

2. 在info.plist中LSApplicationQueriesSchemes添加 `qqmusic`

3. 实现 `QQMusicOpenSDKDelegate` 回调的接口

4. 向SDK注册App

```
  //AppId QQ音乐分配的业务ID
  //AppPrivateKey 自行生成的RSA私钥
  //callbackUrl 合作方App的scheme，授权结果后会通过callbackUrl跳转回合作方App
  [QQMusicOpenSDK registerAppID:AppId SecretKey:AppPrivateKey
callbackUrl:@"openiddemo://" delegate:self];
```

5. 开始授权

```
  [QQMusicOpenSDK startAuth];//将拉起QQ音乐进行授权，授权过程见上方"iOS授权流程说明"
```

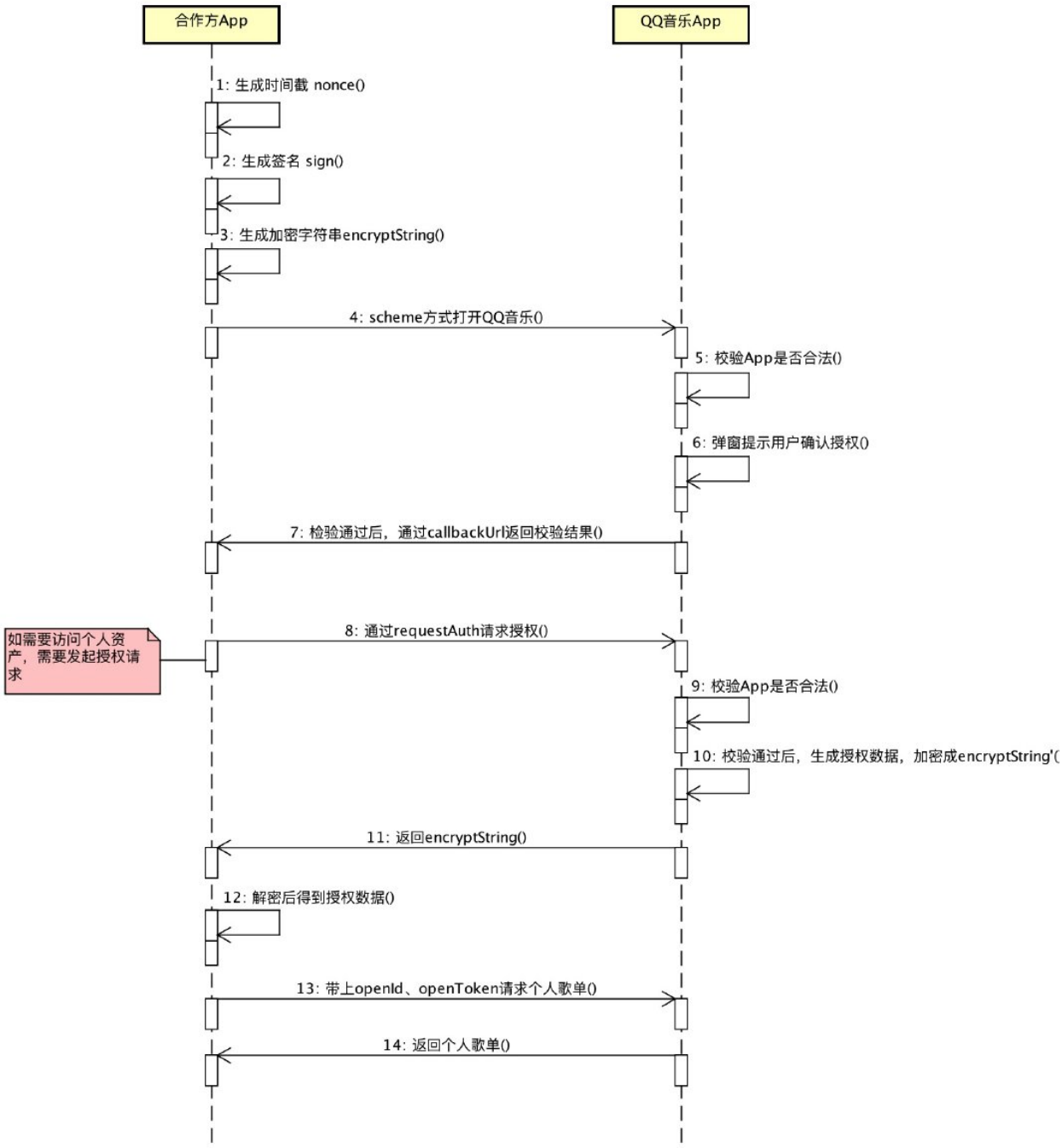
6. 在 `QQMusicOpenSDKDelegate` 回调中得到授权结果 `openID` 及`openToken

7. 调用 `OpenAPI` ,在url参数中加上 `openID` 及 `openToken`

三. Android接入指南

3.1 Android授权流程说明

流程时序图



步骤说明

1. 业务方App，使用当前unix时间截作为随机字符串 `nonce`
2. 使用业务方私钥 `AppPrivateKey` 对nonce进行RSA签名，生成 `sign`
3. 将 `nonce` 、 `sign` 、 `callbackUrl` 组装成json

```
{
    "nonce":nonce,
    "sign":sign,
    "callbackUrl":"openiddemo://"
}
```

使用QQ音乐公钥 `QQMusicPubKey` 对json进行加密，得到 `encryptString`

4. 通过scheme方式拉起QQ音乐，其中scheme url规则如下

将 `cmd` 、 `appId` 、 `encryptString` 、 `callbackUrl` 组成json

```
{
    "cmd":"start",
    "appId":appId,
    "packageName":"xxx",
    "encryptString":encryptString,
    "callbackUrl":"openiddemo://"
}
```

将json使用urlencode编码得到字符串 `param`

生成完整scheme url:

`qqmusic://qq.com/other/openid?p=param`

5. QQ音乐会对scheme传过来的encryptString使用QQ音乐私钥进行解密，并使用合作方公钥 `AppPubKey` 对签名 `sign` 进行验证
6. 验证通过后，QQ音乐App会弹出授权框让用户进行确认
7. 用户确认授权后，会通过 `callbackUrl` 跳转回合作方App，并通过 `ret` 告知授权结果
8. 如果需要访问用户资产，需要通过AIDL接口 `requestAuth` 请求授权。需要带上参数 `encryptString` ，生成规则同第3步。
9. QQ音乐会对scheme传过来的encryptString使用QQ音乐私钥进行解密，并使用合作方公钥 `AppPubKey` 对签名 `sign` 进行验证
10. 验证通过后，QQ音乐会将 `nonce` 、 `sign` 、 `openId` 、 `openToken` 、 `expireTime` 组成json

```
{
    "nonce":auth,
    "sign":appId,
    "openId":18762394837293,
    "openToken":"2sxSws1EbEhiXYRfFImI9ZCQt8a6rWFbg",
    "expireTime":1545994007    //token过期时间，到期后需要重新授权
}
```

QQ音乐会使用合作方公钥 `AppPubKey` 对上述json进行加密，生成 `encryptString'`

11. 通过AIDL回调方式将授权结果 `encryptString'` 返回合作方App
12. 合作方App使用合作方私钥 `AppPrivateKey` 进行解密，最终得到 `` `nonce `` 、 `sign` 、 `openId` 、 `openToken` 、 `expireTime` 。合作方使用QQ音乐公钥 `QQMusicPubKey` 对sign进行签名验证，以确保是来自QQ音乐的数据。
13. 访问个人资产接口时，带上 `openId` 、 `openToken` 。

3.2 开发集成流程

> Android的OpenID授权，是通过AIDL方式进行通讯，需要接入AIDL SDK，详情请参考QQ音乐AIDL文档

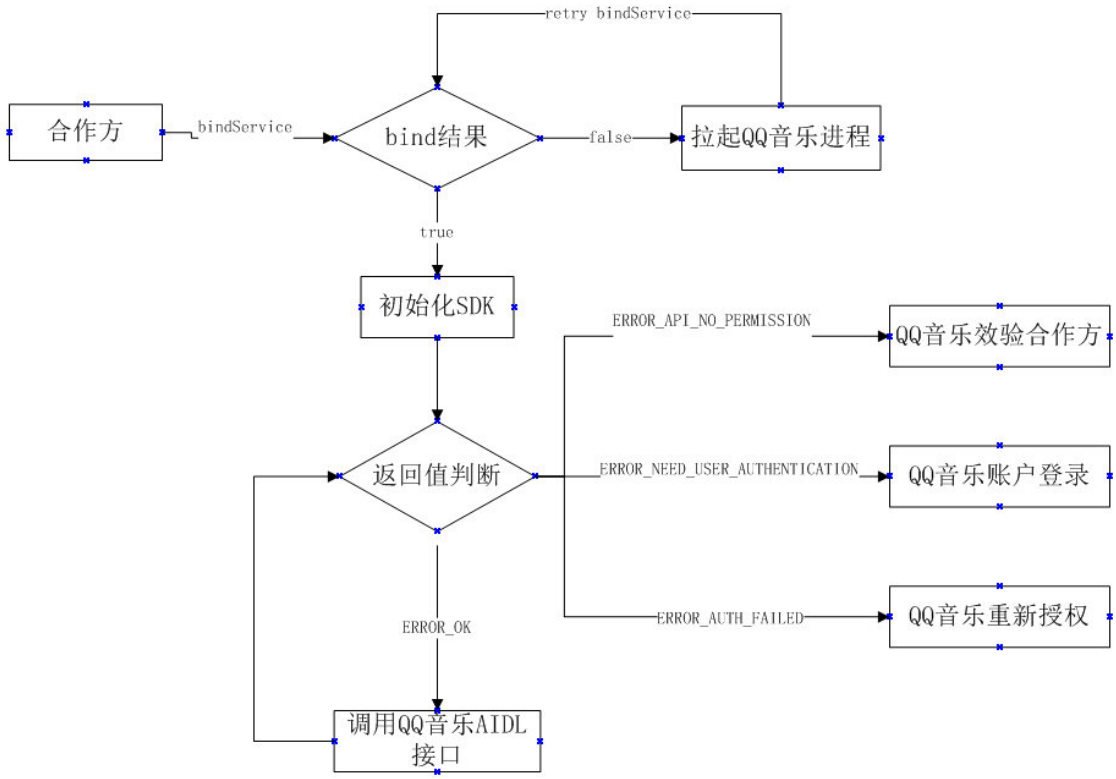
1. 集成QQ音乐AIDL SDK

Android Studio环境下，在build.gradle文件中，添加如下依赖即可：

```
dependencies {
  compile(name:'qqmusic-api-sdk-release', ext:'aar')
}
```

2. 业务方App通过AIDL 方式与QQ音乐进程通信，调用AIDL接口时会返回不同的返回值，合作方需要根据返回值做出不同的回应：

   返回值定义请参考： **6.2.2 AIDL接口返回值说明**



3. 绑定QQ音乐AIDL服务

```
// 必须显式绑定
val intent = Intent("com.tencent.qqmusic.third.api.QQMusicApiService")
intent.`package` = "com.tencent.qqmusic"
var ret = bindService(intent, this, Context.BIND_AUTO_CREATE)
if (!ret) {
    CommonCmd.startQQMusicProcess(this, packageName)
    //定时100ms再进行bindService
}
```

   特别说明：目前主流厂商Android高版本限制，后台应用不经过系统授权，bindService将无法启动后台QQ音乐服务。如果遇到这种情况，bindService失败后请通过QQ音乐SDK内置接口主动拉起QQ音乐应用，再进行bindService操作

4. 初始化SDK

```
var bundle = Bundle()
bundle.putInt(Keys.API_PARAM_KEY_SDK_VERSION, CommonCmd.SDK_VESION)//SDK版本号
var result = qqmusicApi?.execute("hi", bundle)
```

5. 拉起QQ音乐进行校验App

   根据上述流程步骤，使用 `nonce` 、 `sign` 、 `callbackUrl` 生成 `encryptString`

```
CommonCmd.verifyCallerIdentity(this, appId, packageName, encryptString,
"qqmusicapidemo://xxx")
```

6. QQ音乐校验App结束后，会通过上述的 `callbackUrl` 进行回调，

   返回 `qqmusicapidemo://xxx?cmd=verify&ret=0`

   `ret` 为结果码，表示0：授权成功 -1：授权失败 -2：用户取消

7. 校验成功后，可以通过AIDL对QQ音乐的播放控制及普通歌单、电台、排行榜等非用户个人资产数据进行访问。但如果需要访问用户的个人歌单，需要先通过AIDL接口 `requestAuth` 获取授权。

8. 调用 `OpenAPI` ,在url参数中加上 `openID` 及 `openToken`

```
val params = Bundle()
params.putString(Keys.API_RETURN_KEY_ENCRYPT_STRING, encryptString)
qqmusicApi?.executeAsync("requestAuth", params, object :
IQQMusicApiCallback.Stub() {
    if (code == ErrorCodes.ERROR_OK) {
        //解密
        ...
        //检查签名
        if (OpenIDHelper.checkQMSign(sign, nonce)) {
            //拿到openId、openToken授权
            openId = appDecryptJson.getString(Keys.API_RETURN_KEY_OPEN_ID)
            openToken = appDecryptJson.getString(Keys.API_RETURN_KEY_OPEN_TOKEN)
            ...
        }
        ....
    }
})
```

9. 访问个人资产，需要带上openId和openToken

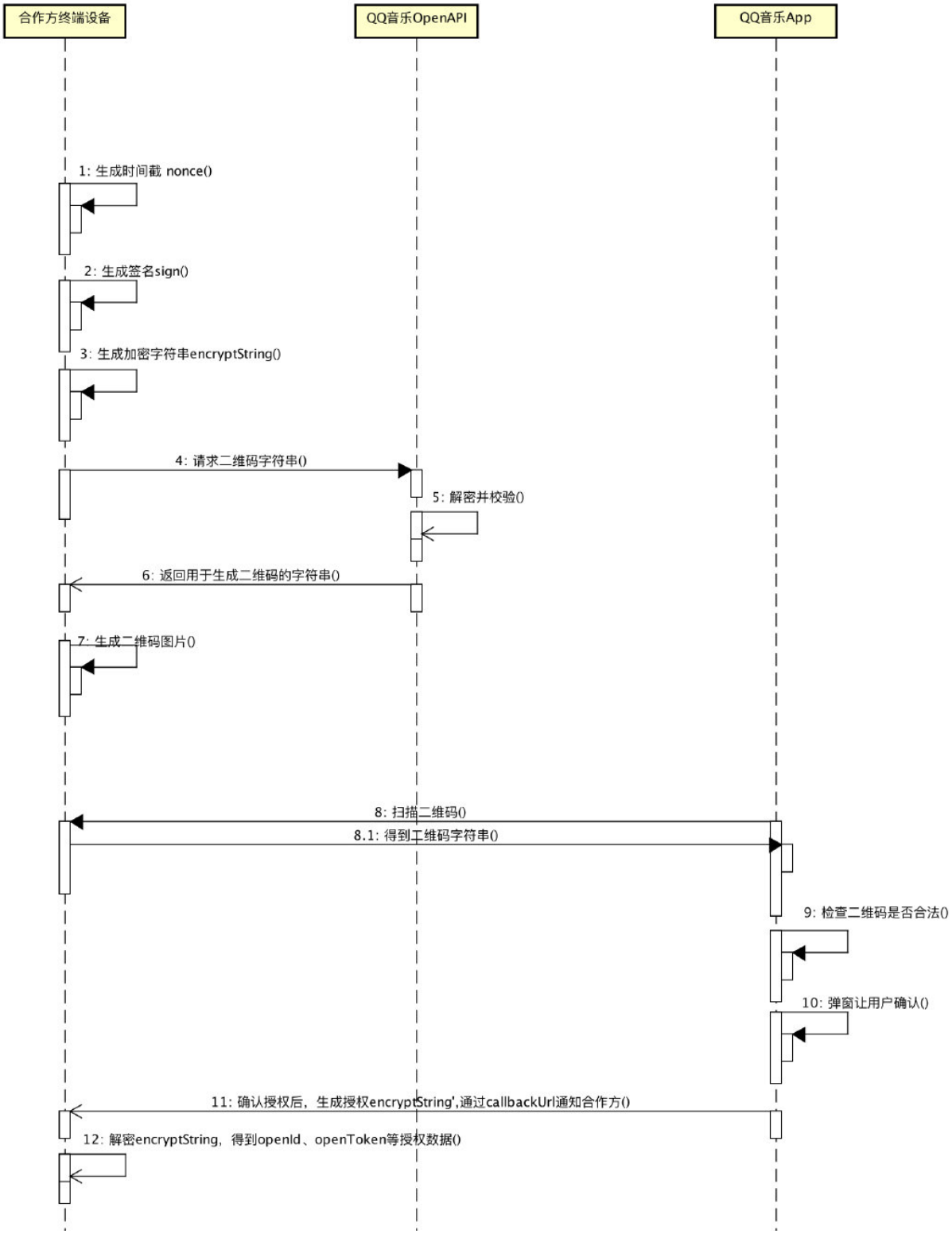
```
//访问个人歌单接口
void getUserFolderList(String openId,String openToken,folderId,int folderType,int
page,IQQMusicApiCallback callback);
//访问个人歌曲接口
void getUserSongList(String openId,String openToken,String folderId,int
folderType,int page,IQQMusicApiCallback callback);
```

四. 异构设备接入指南

除了手机上可以通过QQ音乐App完成OpenID授权，在有屏的异构终端设备上，比如TV、音箱等可以通过二维码方式完成授权

4.1 二维码授权流程说明

流程时序图



步骤说明

1. 合作方终端设备准备生成二维码，使用当前unix时间截作为随机字符串 `nonce`
2. 使用业务方私钥 `AppPrivateKey` 对nonce进行RSA签名，生成 `sign`
3. 将 `nonce` 、 `sign` 、 `callbackUrl` 组装成json

```
{
    "nonce":nonce,
    "sign":sign,
    "callbackUrl":"http://www.demo.com/qm/auth/set?clientid=xxxx"
}
```

其中 `callbackUrl` 为http url，由合作方自定义，但需要确保QQ音乐App可以访问。

使用QQ音乐公钥 `QQMusicPubKey` 对json进行加密，得到 `encryptString`

4. 向QQ音乐OpenAPI请求二维码字符串

Http Url: [https或http]://openrpc.music.qq.com/rpc_proxy/fcgi-bin/music_open_api.fcg?opi_cmd=fcg_music_custom_sdk_get_qr_code.fcg

Method:GET，POST

输入参数说明

输入参数	含义
qqmusic_open_appid	QQ音乐open SDK分配appid
qqmusic_package_name	接入方应用包名称
qqmusic_dev_name	接入方应用设备
qqmusic_encrypt_auth	接入方加密鉴权信息，即上述 `encryptString`

返回数据说明

参数名称	类型	描述
ret	int	返回码。
sub_ret	int	详情返回码，用于具体定位错误，一般为0
msg	string	如果错误，返回错误信息。
sdk_qr_code	string	生成二维码信息

示例： http://openrpc.music.qq.com/rpc_proxy/fcgi-bin/music_open_api.fcg?opi_cmd=fcg_music_custom_sdk_get_qr_code&app_id=xxx&app_key=xxx&device_id=xxx&client_ip=xxx×tamp=xx&sign=xx&qqmusic_open_appid=xxx&qqmusic_package_name=xx&qqmusic_dev_name=xx&qqmusic_encrypt_auth=x

5. QQ音乐对 `encryptString` 进行解密并校验
6. QQ音乐验证通过后，会返回 `qrcodeString` ，用以生成二维码图片

7. 合作方终端设备拿到 `qrcodeString` ,生成二维码图片
8. QQ音乐App(iOS或Android)打开『扫一扫』，对设备上的二维码进行扫描
9. QQ音乐检查二维码是否合法
10. QQ音乐App弹窗提示用户确认授权
11. 用户确认授权后，QQ音乐App会调用上面的 `callbackUrl` ，以通用合作方授权完成。QQ音乐发起http请求：

    Http Url: `callbackUrl`

    Method：POST

    Post Body如下

    ```
    {
       "ret": 0,
       "encryptString": encryptString
    }
    ```

12. 合作方收到回调后，使用合作方私钥 `AppPrivateKey` 对encryptString `进行解密，最终得到` ``nonce` 、 `sign` 、 `openId` 、 `openToken` 、`expireTime

4.2 开发集成流程

二维码授权一般在异构设备上，合作方和QQ音乐只有两个交互：

1. 向QQ音乐获取二维码字符串
2. 授权完成后QQ音乐通过 `callbackUrl` 通知合作方授权完成

只要保证上述两个请求正常，其它步骤合作方可以根据自己实际情况灵活开发。

QQ音乐针对二维码授权提供了Demo，模拟了在网页上显示二维码进行授权的过程。

其中后台服务采用golang开发，前端使用html展示，需要部署在linux或mac进行体验。

五. QQ音乐OpenAPI访问

另外参考 《open_api_custom_version.docx》

六. 系统说明

6.1 参数说明

名称	含义
`AppId`	合作方业务ID，QQ音乐分配
`PackageName`	程序包名，在Android是PackageName，iOS是BundleId。如果Android和iOS的包名不一样，需要都给到。
`AppIcon`	业务方logo
`AppPubKey`	合作方RSA公钥。QQ音乐会保存合作方的公钥，用来验证合作方数据的签名
`AppPrivateKey`	合作方RSA私钥，合作方自行妥善保管
`OpenAPI AppId`	QQ音乐分配的OpenId AppId，**访问QQ音乐OpenAPI时才需要用到**，注意与认证AppId区别
`OpenAPI AppKey`	QQ音乐分配的OpenAPI AppKey，**访问QQ音乐OpenAPI时才需要用到**
`OpenAPI app_private_key`	QQ音乐分配的OpenAPI私钥，**访问QQ音乐OpenAPI时才需要用到**，注意和 `AppPrivateKey` 的使用
`QQMusicPubKey`	QQ音乐RSA公钥，合作方将数据用它将数据加密成 `encryptString`
`nonce`	随机数，使用unix时间截，例如1546048533
`sign`	对随机数使用私钥加密后的签名，使用对方公钥验证签名
`callbackUrl`	在手机App环境中代表scheme url,例如qqmusic://。在二维码场景为http链接 http://xxx
`encryptString`	使用公钥加密后的字符串
`openId`	授权后拿到的用户标识，同一个业务下同一个用户的openId不会变
`openToken`	授权票据，有时效性
`expireTime`	`openToken` 的过期时间
`ret`	返回码，参考 6.2.1 ret 返回值
`qrcodeString`	二维码字符串，由QQ音乐生成，合作方通过OpenAPI向QQ音乐请求。

6.2 返回码说明

6.2.1 `ret` 返回码

授权操作结束后，QQ音乐会将操作结果 `ret` 放在 `callbackUrl` 中返回给合作方

数值	说明
0	授权成功
-1	授权失败
-2	用户取消

6.2.2 AIDL返回码

AIDL接口execute、executeAsync、registerEventListener、unregisterEventListener，除executeAsync以外其他接口均返回Bundle数据集

```
var errCode = result.getInt(Keys.API_RETURN_KEY_CODE)//获取错误码
```

错误码	说明
`ERROR_API_NO_PERMISSION`	表示没有权限，需要拉起QQ音乐校验，用户确认是否授权
`ERROR_NEED_USER_AUTHENTICATION`	表示需要用户登录态（比如获取用户歌单，歌曲收藏状态）,需要拉起QQ音乐进行登录。
`ERROR_AUTH_FAILED`	表示需要OpenID授权，需要请求AIDL重新授权。
`...`	其它错误码参考AIDL javadoc

6.3 RSA公私钥说明

QQ音乐OpenID授权方案使用的RSA密钥位数为**1024**位，密钥格式使用**PKCS#8**。

使用openssl来生成示例：

1. 命令生成原始 RSA私钥文件 `rsa_private_key.pem`

```
openssl genrsa -out rsa_private_key.pem 1024
```

2. 命令将原始 RSA私钥转换为 pkcs8格式，得私钥文件到 `private_key.pem`

```
openssl pkcs8 -topk8 -inform PEM -in rsa_private_key.pem -outform PEM -nocrypt -out
private_key.pem
```

3. 生成RSA公钥文件 `rsa_public_key.pem`

```
openssl rsa -in rsa_private_key.pem -pubout -out rsa_public_key.pem
```

4. 生成成功后，将公钥文件 `rsa_public_key.pem` 给QQ音乐，私钥 `private_key.pem` 合作方妥善保管。