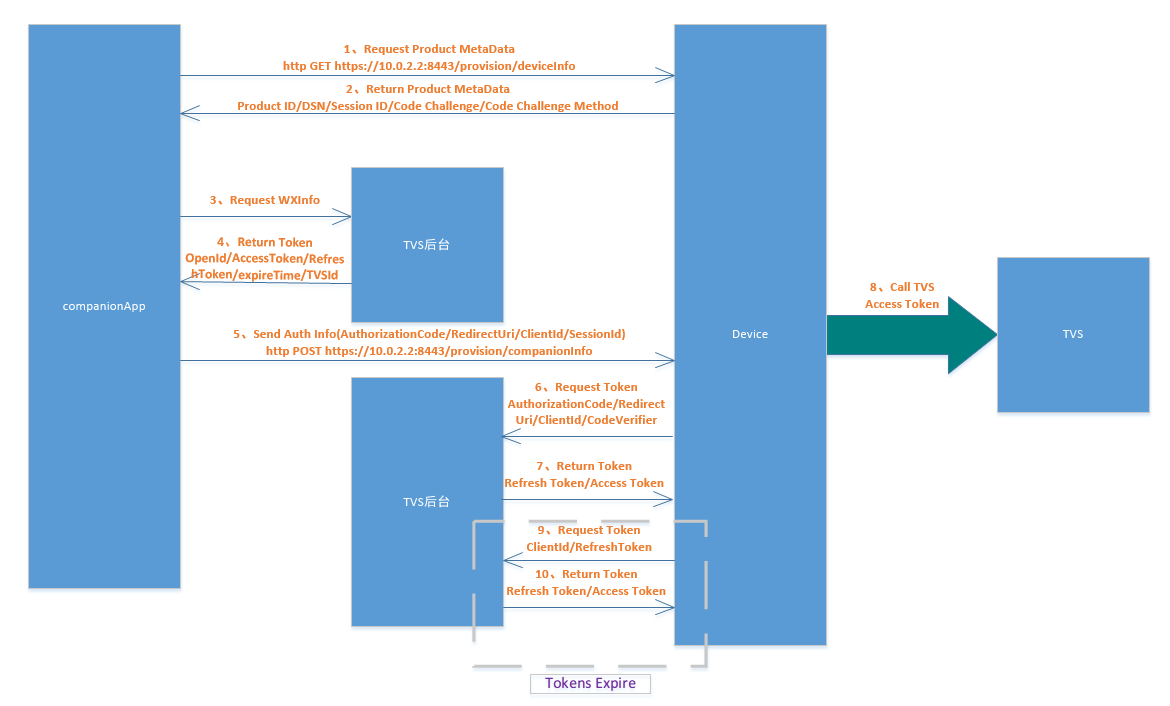
**Android厂商APP接入**

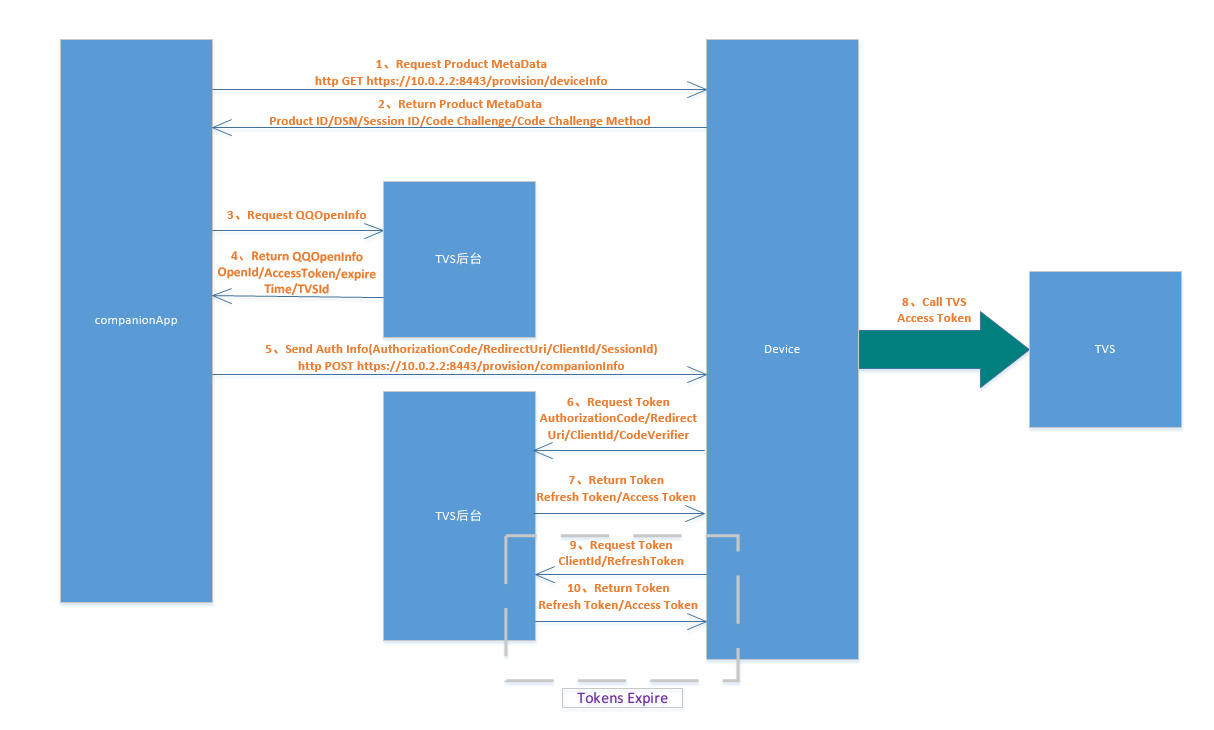
1. **登录验证流程**
2. 厂商APP-LWA-AVS音箱-AVS验证流程



1. 厂商APP-微信开放平台-TVS后台-TVS音箱-TVS验证流程



3、厂商APP-QQ互联平台-TVS后台-TVS音箱-TVS验证流程



**二、厂商APP接入TVS登录系统配置步骤**

1. 打开工程app目录下的build.gradle，

（1）确保defaultConfig下的applicationId为微信开放平台下注册的包名；

（2）确保signingConfigs目录下storeFile.file参数路径正确，keyAlias、keyPassword、storePassword均与微信开放平台下签名参数一致。

1. 将tvsloginlib-XXX-release.aar放入app\libs目录下，打开build.gradle将以下参数配置在dependencies属性下

compile(name: tvsloginlib-XXX-release, ext:'aar')

compile 'com.android.support:design:xx.x.0'

compile 'com.tencent.mm.opensdk:wechat-sdk-android-with-mta:1.0.2'

compile 'com.google.code.gson:gson:2.8.1'

compile 'org.jbundle.util.osgi.wrapped:org.jbundle.util.osgi.wrapped.org.apache.http.client:4.1.2'

repositories {

flatDir {

dirs 'libs'

}

}

确保minSdkVersion大于等于15

如有别的module需要引用这个aar所在的module，需要在另一个module的gradle里配置如下

repositories {

flatDir {

dirs project(':[aarmodulename]').file('libs')

}

}

1. 打开工程AndroidManifest.xml，

（1）确保package名称与微信开放平台下注册的包名一致；

（2）确保以下权限被添加

<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"/>

<uses-permission android:name="android.permission.WRITE\_SETTINGS"/>

（3）将application的name改为com.tencent.ai.tvs.LoginApplication，添加android:exported=”true”属性（若Application使用自定义的话，实例化LoginProxy需传入Application的Context实例）

（4）在application标签内新添加一个Activity

<activity

android:name="com.tencent.tauth.AuthActivity"

android:noHistory="true"

android:launchMode="singleTask" >

<intent-filter>

<action android:name="android.intent.action.VIEW" />

<category android:name="android.intent.category.DEFAULT" />

<category android:name="android.intent.category.BROWSABLE" />

<data android:scheme="[QQAPPID]" />

</intent-filter>

</activity>

其中[QQAPPID]为QQ互联平台注册的AppId

1. 登录Activity继承AuthorizeListener接口实现其中两个方法，具体说明见第三部分
2. 登录Activity重写onActivityResult回调，用于接收QQ登录的票据信息
3. **TVSSDK接口及使用说明**
4. **LoginProxy**

TVS登录功能代理类

**getInstance**

获取LoginProxy实例

static LoginProxy getInstance(String appIdWx, String appIdQQOpen)

参数

appIdWx：传入微信公众平台申请的appId

appIdQQOpen：传入QQ互联平台申请的appId

**getInstance**

获取LoginProxy实例，自定义Application方式

static LoginProxy getInstance(String appIdWx, String appIdQQOpen, Context context)

参数

appIdWx：传入微信公众平台申请的appId

appIdQQOpen：传入QQ互联平台申请的appId

context：传入Application Context

**getInfoManager**

获取LoginInfoManager实例

LoginInfoManager getInfoManager(ELoginPlatform platform)

参数

platform：传入需要获取哪个帐号平台的登录信息

**requestLogin**

请求TVS登录

void requestLogin(ELoginPlatform platform, String productId, String dsn, Activity activity)

参数

platform：传入需要登录哪个帐号平台

productId：设备的productId

dsn：设备的序列号

activity：登录Activity实例

**requestTokenVerify**

请求验票刷票

void requestTokenVerify(ELoginPlatform platform, String productId, String dsn)

参数

platform：传入在哪个帐号平台验票刷票

productId：设备的productId

dsn：设备的序列号

**isTokenExist**

判断Token信息存在与否

boolean isTokenExist(ELoginPlatform platform, Context context)

参数

platform：传入需要判断哪个帐号平台票据信息是否存在

context：当前应用上下文

**clearToken**

清除Token信息，用于注销

void clearToken(ELoginPlatform platform, Context context)

参数

platform：传入需要清除哪个帐号平台的票据信息

context：当前应用上下文

**setOwnActivity**

设置Wup接口调用所属的Activity，用于WUPResp与UI线程进行同步

void setOwnActivity(Activity activity)

参数

activity：传入用于授权后回调的Activity

**setAuthorizeListener**

设置票据验证监听器

void setAuthorizeListener (AuthorizeListener listener)

参数

listener：AuthorizeListener实例

**handleQQOpenIntent**

QQ登录帐号信息回调

void handleQQOpenIntent(int requestCode, int resultCode, Intent data)

参数

requestCode：Activity跳转请求Code，确认返回数据是哪个Activity的

resultCode：子Activity通过setResult返回的code

data：Intent对象，带有返回的数据

**handleQQOpenUserInfo**

QQ登录用户信息回调

void handleQQOpenIntent()

**getClientId**

获取用于对齐AVS厂商信息的ClientId

String getClientId(ELoginPlatform platform)

参数

platform：获取哪个登录平台的信息

**getUserId**

获取用户Id

String getUserId()

**setLoginEnv**

设置帐号后台环境

void setLoginEnv(ELoginEnv env)

参数

env：帐号后台环境类型

**setUserCenterEnv**

设置用户中心网络环境

void setUserCenterEnv(ELoginEnv env)

参数

env：用户中心网络环境类型

**requestGetCaptcha**

获取手机号短信验证码

void requestGetCaptcha(ELoginPlatform platform, String phoneNumber)

参数

platform：帐号类型

phoneNumber：需要绑定的手机号

**requestBindPhoneNumber**

绑定手机号

void requestBindPhoneNumber(ELoginPlatform platform, String phoneNumber, String captcha)

参数

platform：帐号类型

phoneNumber：需要绑定的手机号

captcha：手机验证码

**isLocationOpValid**

地址操作是否合法

boolean isLocationOpValid(ELoginPlatform platform)

参数

platform：帐号类型

返回值

true：合法，可以进行地址操作

false：非法，不可进行地址操作

**requestBindLocation**

常用地址绑定操作

void requestBindLocation(ELoginPlatform platform, LocationInfo homeLocationInfo, LocationInfo companyLocationInfo)

参数

platform：帐号类型

homeLocationInfo：常用地址（家庭）信息

companyLocationInfo：常用地址（公司）信息

**requestQueryLocation**

常用地址查询操作

void requestQueryLocation(ELoginPlatform platform)

参数

platform：帐号类型

**setBindingListener**

设置绑定信息监听器

void setBindingListener(BindingListener listener)

参数

listener：绑定信息监听器

**toUserCenter**

跳转用户中心页面

void toUserCenter(EUserAttrType type, DeviceManager deviceManager)

参数

type：H5页面类型

deviceManager：设备信息类型

**requestSetPushMapInfoEx**

设置推送绑定接口

void requestSetPushMapInfoEx(ELoginPlatform platform, PushInfoManager pushInfoManager, DeviceManager deviceManager)

参数

platform：帐号类型

pushInfoManager：推送信息管理器

deviceManager：设备信息管理器

**requestDelPushMapInfo**

解除设备推送绑定接口

void requestDelPushMapInfo(ELoginPlatform platform, PushInfoManager pushInfoManager, DeviceManager deviceManager)

参数

platform：帐号类型

pushInfoManager：推送信息管理器

deviceManager：设备信息管理器

**requestGetPushDeviceInfo**

void requestGetPushDeviceInfo(ELoginPlatform platform)

参数

platform：帐号类型

**getMemberStatus**

获取设备领取会员信息接口

void getMemberStatus(ELoginPlatform platform, DeviceManager deviceManager)

参数

platform：帐号类型

deviceManager：设备信息管理器

**isWXAppInstalled**

判断手机是否安装微信

boolean isWXAppInstalled()

**isWXAppSupportAPI**

判断手机微信App版本是否支持登录

boolean isWXAppSupportAPI()

1. **LoginInfoManager**

微信登录信息管理类，用户信息基类，被WxInfoManager和QQOpenInfoManager继承。

**accessToken**

String授权AccessToken

**refreshToken**

String 授权RefreshToken

**openID**

String 用户OpenID

**expireTime**

long 刷票过期时间

**tvsID**

String 授权后请求到的tvsID

1. **AuthorizeListener**

微信票据验证监听器

void **onSuccess**(int type);

微信接口调用成功回调，用于UI更新

参数

type:调用类型

AUTH\_TYPE：授权票据类型

REFRESH\_TYPE：刷新票据类型

WX\_TVSIDRECV\_TYPE：微信TVSID接收类型

QQOPEN\_TVSIDRECV\_TYPE：QQ TVSID接收类型

TOKENVERIFY\_TYPE：QQ帐号验证类型

USERINFORECV\_TYPE：用户信息获取类型

void **onError**(int type)

微信接口调用失败回调，用于UI更新

参数

type:调用类型

AUTH\_TYPE：授权票据类型

REFRESH\_TYPE：刷新票据类型

TVSIDRECV\_TYPE：TVSID接收类型

WX\_TVSIDRECV\_TYPE：微信TVSID接收类型

QQOPEN\_TVSIDRECV\_TYPE：QQ TVSID接收类型

TOKENVERIFY\_TYPE：QQ帐号验证类型

USERINFORECV\_TYPE：用户信息获取类型

1. **EloginPlatform**

登录帐号类型枚举

WX：微信帐号

QQOpen：QQ互联帐号

QQ：QQ帐号（内部）

1. **EloginEnv**

帐号后台环境配置

FORMAL：正式环境

TEST：测试环境

1. **UserInfoManager**

用户信息管理类，包含

idType 帐号类型 0：微信 1：QQ

nickName 昵称

headImgUrl 头像URL地址

sex 性别

unionId 微信UnionId

**getInstance**

获取UserInfoManager实例

static UserInfoManager getInstance()

1. **LocationInfo**

地址信息结构体类，包含

longitube：经度信息

latitube：纬度信息

addr：详细地址

name：地点名称

cabLongitube：打车点经度信息

cabLatitube：打车点纬度信息

cabAddr：打车点详细地址

cabName：打车点地点名称

1. **LocManager**

地址信息管理类

**getInstance**

获取LocManager实例

static LocManager getInstance()

**getLocation**

获取地址信息结构体

LocationInfo getLocation(ELocationType type)

参数

type：地址信息类型

1. **ElocationType**

常用地址类型枚举

HOME：家庭类型

COMPANY：公司类型

1. **BindingListener**

用户信息绑定监听器

void onSuccess(int type);

用户信息绑定接口调用成功回调，用于UI更新

参数

type:调用类型

GET\_CAPTCHA\_TYPE：获取短信验证码类型

BIND\_PHONENUMBER\_TYPE：绑定手机号类型

BIND\_LOCATION\_TYPE：绑定常用地址类型

QUERY\_LOCATION\_TYPE：查询常用地址类型

SET\_PUSH\_MAP\_INFOEX\_TYPE：设备绑定类型

DEL\_PUSH\_MAP\_INFO\_TYPE：设备解绑类型

GET\_PUSH\_DEVICE\_INFO\_TYPE：获取绑定设备列表类型

BIND\_GET\_MEMBER\_STATUS\_TYPE：绑定设备领取会员状态查询类型

void onError(int type)

用户信息绑定接口调用失败回调，用于UI更新

参数

type:调用类型

GET\_CAPTCHA\_TYPE：获取短信验证码类型

BIND\_PHONENUMBER\_TYPE：绑定手机号类型

BIND\_LOCATION\_TYPE：绑定常用地址类型

QUERY\_LOCATION\_TYPE：查询常用地址类型

SET\_PUSH\_MAP\_INFOEX\_TYPE：设备绑定类型

DEL\_PUSH\_MAP\_INFO\_TYPE：设备解绑类型

GET\_PUSH\_DEVICE\_INFO\_TYPE：获取绑定设备列表类型

BIND\_GET\_MEMBER\_STATUS\_TYPE：绑定设备领取会员状态查询类型

1. **PushInfoManager**

推送信息管理类

**getInstance**

获取PushInfoManager实例

static PushInfoManager getInstance()

id：推送ID

idExtra：推送额外信息

idType：推送类型

1. **DeviceManager**

设备信息管理类

DeviceManager构造函数

public **DeviceManager**()：用于构造手机端设备信息

public **DeviceManager**(String ip, String data)：用于构造设备端设备信息

参数

ip：设备目标

data：设备信息JSON数据

qua：设备QUA

guid：设备GUID

imei：设备IMEI

licence：设备licence

mac：设备mac地址

qimei：设备qimei

dsn：设备序列号

ip：设备ip

os：设备操作系统

deviceOEMUrl：品牌Icon

deviceOEM：品牌名称

deviceType：设备类型

deviceSerial：设备序列号

deviceMark：设备备注

deviceId：设备ID

deviceName：设备名称

manufacturer：设备制造商

packageName：设备主包名

isBinded：是否绑定过

bindOpenId：绑定的帐号OpenID

bindAccountType：绑定的帐号类型

bindTime：绑定时间

enrollTime：注册时间

1. **BindManager**

设备绑定接口管理类

**getInstance**

获取BindManager实例

static BindManager getInstance()

**pushDeviceInfos**

返回绑定设备列表

**bind**

设备绑定操作

void bind(String srcIP, String dstIP, String data, final BindManager.Callback callback)

参数

srcIP：源IP

dstIP：绑定设备IP

data：设备信息JSON数据

callback：绑定状态回调

**BindManager.Callback**接口

void **onSuccess**(String msg)

绑定成功回调，用于UI更新

void **onFail**(String msg)

绑定失败回调，用于UI更新

1. **UDPClient**

UDP接口类，用于设备扫描操作（设备与手机在同一局域网内）

**start**

开始设备扫描，并注册扫描回调

void start(UDPClient.Callback callback);

**requestAll**

请求扫描设备信息

void requestAll(int req)

参数

req：扫描类型

**UDPClient.Callback**接口

void **onUDPResponse** (String ip, int req, String data)

设备扫描回调

参数

ip：目标设备ip

req：请求类型

data：设备JSON数据

UDP\_CLIENT\_STARTED：扫描开始回调标志

REQ\_DEVICE\_INFO ：扫描到设备信息回调标志

SET\_ACCOUNT\_INFO：帐号绑定回调标志

1. **EUserAttrType**

用户中心类型枚举

HOMEPAGE：主页

CPOPERATION：运营页面

INFOSETTING：常用信息设置页面

AGREEMENT：声明页面

PRIVACY：隐私协议页面

FEEDBACK：反馈页面

1. **使用示例**

