Weline Android SDK 开发手册

Weline.io

关于本手册

版权声明

信息更新

本文档仅用于为NDA用户提供信息，并且随时可由Weline.io更改或撤回。

免责条款

根据适用法律的许可范围，Weline.io按“原样”提供本文档而不承担任何形式的担保，包括（但不限于）任何隐含的适销性、特殊目的适用性或无侵害性。在任何情况下，Weline.io都不会对最终用户或任何第三方因根据说明文档使用SDK/后台接口造成的任何直接或间接损失或损坏负责，即使Weline.io明确得知这些损失或损坏，这些损坏包括（但不限于）利润损失、业务中断、信誉或数据丢失。

阅读对象

本文的读者对象为和 Weline.io 签署了 NDA（保密协议）的第三方厂商的研发人员。

密级

本文的密级为秘密，禁止向非 NDA 公司或个人开放此文件。

目录

修订记录

日期	修订版本	描述
2017/11/01	1.0.0	初稿完成
2017/11/03	1.0.1	补充状态信息
2018/03/21	1.0.6	SDK 整合成安装包
2019/03/25	2.1.6	重新修订
2019/04/04	2.1.7	新增根据当前连接状态执行监听操作的接口
2019/04/10	2.1.8	新增 FAQ
2019/04/12	2.1.9	新增测试验证场景
2019/04/22	2.1.10	排版优化,FAQ 补充了关于重连时发送通知的说明
2019/06/21	2.2.0	解决了华为网需等待的问题

1 概要

此文档描述了第三方厂商集成虚拟网时，在 Android 平台上所使用的 Weline.io SDK，其目的是让第三方厂商的研发人员能快速将虚拟网功能嵌入到第三方应用程序中。

此文档适合公司内部和签署了 NDA 的第三方研发人员阅读。

2 系统需求

运行的目标系统要求 Android 5.0 以上，Android 4.4 及以下的系统不提供技术支持。

3 工作准备

在使用 Weline.io SDK 之前，请联系 Weline.io Inc. 获取 SDK 使用授权，取得授权后，将得到一个 AppId、PartnerId。

随本文档一起分发的 Weline.io SDK 包，包括以下内容：

1. Android Demo 安卓项目
2. Cmapi_vX_X_X_cmg.aar 集成 sdk 的 aar 包(位于 Demo 的 libs 目录下)

3.1 配置环境

第一步：将 aar 包放入工程项目的 libs 目录下，如下图：

第二步：在工程项目的 build.gradle 文件中做如下配置：

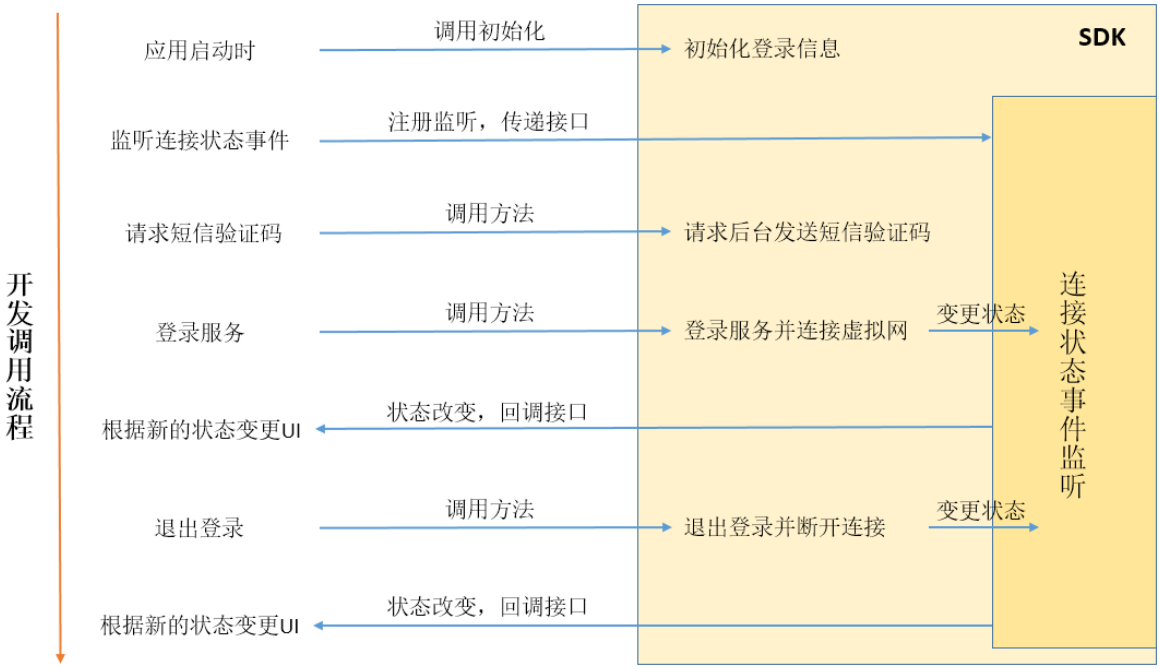
```
android {
    defaultConfig {...}
    buildTypes {...}

    repositories {
        flatDir {
            dirs 'libs'
        }
    }
}

dependencies {
    implementation(name: 'cmapi_v2_1_7_cmg', ext: 'aar')
}
```

此处依赖的 cmapi_v2_1_7_cmg 名称以实际分发的 aar 包名称为准。

4 开发流程



开发调用流程与 SDK 交互原理示意如上图。

4.1 初始化

在应用程序创建时初始化 SDK 引用的 Context 建议使用全局变量，推荐在 Application 中完成初始化。

appId 和 partnerId 是字符串，devicesClass 是整型，作为识别应用、设备的标识，需要向我们申请。

```
CMAPI.getInstance().init(Context context, String appId, String partnerId, String devicesClass);
```

4.2 状态事件监听

在登录之前还必须要对状态事件设置监听,可用于控制界面跳转和弹窗提示。

设置状态事件监听方法如下:

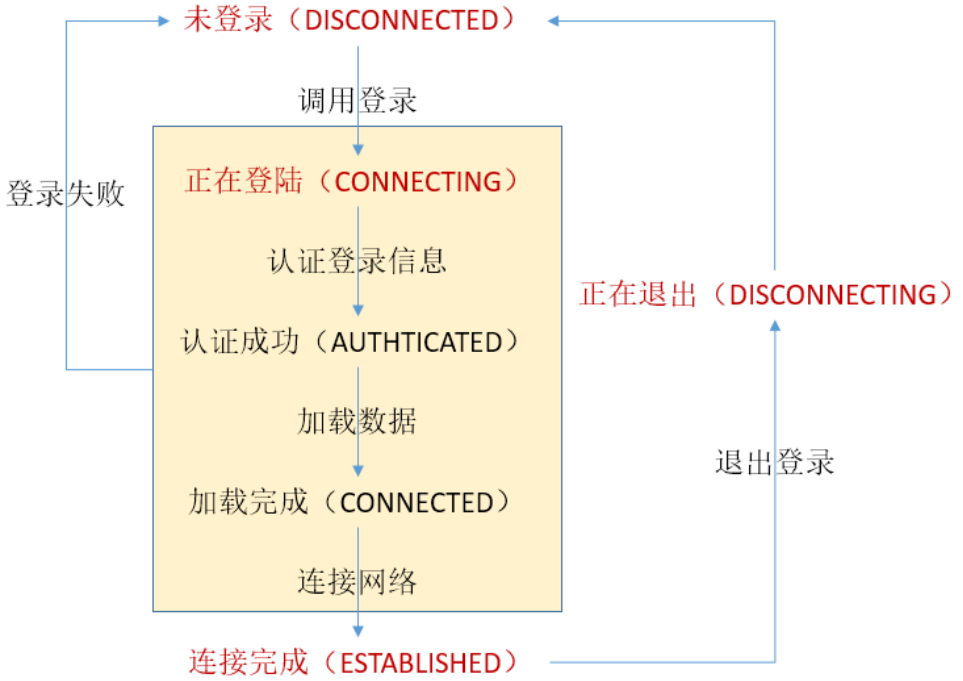
```
CMAPI.getInstance().addConnectionStatusListener(statusListener);
```

在注册接口的页面被销毁时要及时取消监听:

```
CMAPI.getInstance().removeConnectionStatusListener(statusListener);
```

监听接口内容参看 8.1。

状态变化流程图如下:



注册状态监听后，上述状态每次发生变化，都会回调监听接口，并携带当前相应的状态参数。

状态事件即：与服务器连接状态发生变更的事件，为如下 6 种：

(未连接-) onConecting (-等待认证);
(认证完成-) onAuthenticated (-等待数据);
(加载完成-) onConnected (-等待启动 VPN 通道);
(启动 VPN 通道完成-) onEstablished (-已连接);
(连接中-) onDisconnecting (-正在断开连接);
(正在断开连接-) onDisconnected (-未连接)。

状态事件分隔了 6 种对应的连接状态，其触发流程代表了的主程序操作的顺序。onDisconnecting 回调偶尔不会出现，由断开速度决定。

更详细使用方法请参看 Demo。

4.3 登录连接

使用如下方式调用 CMAPI 中的 login 登录方法，需要传入当前活动页实例、用户名、密码以及 ResultListener 实例。

```
CMAPI.login(account, password, resultListener);
```

如果当前要登录的账号已经被保存（登录过一次并且没有注销），则无需密码、验证码使用 loginByTicket()接口即可完成登陆。

错误回调接口如下:

```
private ResultListener resultListener = new ResultListener() {
    @Override
    public void onError(int result) {
        //错误回调,错误码含义参看 5.3
    }
};
```

值得注意的是，在调用登录接口前，必须保证当前状态是是未连接,获取当前状态的方法如下:

```
int status = CMAPI.getRealtimeInfo().getCurrentStatus();
```

状态值代表的含义参看 5.1。

断开连接的方法如下:

```
CMAPI.getInstance().disconnect();
```

终止登录操作的方法如下:

```
CMAPI.getInstance().cancelLogin();
```

首次登录须要请求 VPNService 的权限，这里会弹出系统的授权窗口

4.4 一般事件监听

一般事件可被多个观察者订阅，触发事件后对所有订阅了该事件的观察者做回调处理

订阅接口 observer 本质是抽象类，观察者在订阅时可为想关注的事件重写回调方法，对不关心的事件可不重写

订阅方式如下:

```
CMAPI.getInstance().subscribe(observer);
```

常用订阅接口 observer 如下:

```
private EventObserver observer =  new EventObserver() {
    @Override
    public void onRealTimeInfoChanged(RealtimeInfo info) {
    //实时信息变更，该方法订阅后每秒回调一次,用于刷新在线时长、时延等实时信息
```

```
    }
    @Override
    public void onTunnelRevoke(boolean isRevoked) {
    // isRevoked: VPN 隧道是否被占用，仅在其值改变时回调该方法
    回调 true 的情况:
        1.当检测出 VPN 隧道被占用时回调;
        2.隧道被抢占时直接回调;
        3.登录前检测是否有其他使用 VPN 的应用，有则回调.
    回调 false 的情况:
        在 VPN 隧道被占用后，每隔一定时间 sdk 会自动检测一次当前是否有其他应用正在使用 VPN 隧道，没有则回调，每次占用仅回调一次(在该回调返回 true 后的 5 秒内不会返回 false，保护期)
    }
};
```

在某一观察者不再使用，或者在其所依附的组件被销毁时，调用 unsubscribe 方法取消它的事件订阅，可有效控制内存泄漏

```
CMAPI.getInstance().unsubscribe(observer);
```

详细使用方法请参看 Demo MainActivity.java 文件。

5 常量

5.1 连接状态

CS_UNKNOWN = 0;	未初始化
CS_PREPARE = 1;	准备就绪
CS_CONNECTING = 2;	**正在连接**
CS_CONNECTED = 3;	**已连接成功**
CS_DISCONNECTED = 4;	**已断开**
CS_AUTHENTICATED = 5;	**已认证成功**
CS_DISCONNECTING = 8;	正在断开
CS_WAIT_RECONNECTING = 9;	等待重连
CS_ESTABLISHED = 200;	**虚拟网连接成功**

粗体标注了需要重点关注的状态

5.2 登录连接或异常断开的错误码

0	未知
1	主动退出
2	版本太低(被限制登录)

3	与后台服务器的网络连接中断(如后台升级)
4	未配置过任何账号信息或者设备账号信息丢失
5	无效的用户名
6	无效的密码，或者口令
9	设备被限制登入
10	设备数量超限
11	无可用的虚拟网络（一般不会发生）
15	系统无可用 tun 设备，或者 tun0 设备其他程序占用
20	认证模式不匹配(验证码错误或认证模式错误)
21	认证无效
2017	app 运行错误
2018	VPN 权限被拒绝，请设置允许该应用使用 VPN 权限
2019	連接超時
2020	VPN 通道被占用，请关闭其他 VPN

6 API 说明

6.1 初始化

在调用 SDK 内其他 API 之前，必须调用一次初始化函数，以便对 SDK 进行初始化工作。

```
函数原型
    public void  init(Context context, String appId, String partnerId, String devicesClass)

参数说明
    partnerId, appId, devicesClass: 申请的开发者账号信息
```

6.2 登录

```
函数原型
  public void login(String account, String password, ResultListener listener)

    参数说明
    account：登录账号
    password：登录密码，如果成功登录过，此值可传空串，表示使用上一次的密码登录
    listener：结果监听器，提供耗时操作执行完成后所使用的回调方法
```

说明：登录过程是异步的，即调用此方法成功后，仅表示成功开始了登录过

程，必须通过监听状态事件(6.10)来获知登录的结果，如从 connecting 到 connected，表示登录成功，如果从 connecting 到 disconnected，表示登录失败，具体的失败原因通过 listener 的回调来获取。

首次登录须要请求 VPNService 的权限，这里会弹出系统的授权窗口

如使用保存的历史登录账号进行登录，则 password、auxAuthValue 参数可以传空字符串。

6.3 登出

函数说明

```
public void disconnect()
```

说明: 和登录过程一样，登出的过程也是异步操作，需要等待状态从 connected 转换到 disconnected 才完成整个登出操作

6.4 终止登录过程

既取消正在进行的登录操作。

函数说明

```
public boolean cancelLogin()
```

说明:在 login() 调用成功后，状态转换成 CONNECTED 之前，都可以调用 cancelLogin 终止登录过程。

6.5 获取基本信息

包含登录的用户基本信息、用于校验的票据。

函数原型

```
public BaseInfo getBaseInfo()
```

返回值

BaseInfo 类型的对象

说明：BaseInfo 结构参看 7.1。

6.6 获取实时信息

包含当前连接的实时信息，如：在线时长、连接时延、连接状态等。

函数原型

```
public RealtimeInfo getRealtimeInfo()
```

返回值

RealtimeInfo 类型的对象

说明：RealtimeInfo 结构参看 7.2。

6.7 获取当前账号

```
函数原型
    public String getAccount()
返回值
    当前登录的或保存的历史账号
```

说明：返回的为最近登录成功的账号，可用于后续的自动登录。

6.8 移除保存的账号

```
函数原型
    public boolean removeUser(String id)

参数说明
    account: 登录账号
返回值
    true 表示成功，false 表示失败
```

说明：保存的账号移除后将不再保存账号的登录信息，再次登录此账号时必须传入密码及短信验证码。

6.9 订阅状态事件

```
函数原型
  public  void  addConnectionStatusListener(ConnectStatusListenerPlus
statusListener)
参数说明
    statusListener: 状态事件的回调接口
```

接口中任意方法被回调时均表示当前进入了对应的状态。

具体内容参看 8.1。

使用方式请参考 Demo。

6.10取消状态事件的订阅

```
函数原型
public void  removeConnectionStatusListener(ConnectStatusListenerPlus
statusListener)
参数说明
    statusListener: 状态事件的回调接口
```

6.11根据当前连接状态执行监听操作

```
函数原型
    public   void  toDoListenerByStatus   (ConnectStatusListenerPlus
statusListener)
参数说明
    statusListener: 状态事件的回调接口
```

说明：根据当前连接状态调用传入的状态事件监听中对应的接口，此接口可作为启动新页面时，错过监听状态变化时的初始化处理。

6.12订阅一般事件

```
函数原型
    public void  subscribe(EventObserver observer)
参数说明
    observer: 一般事件的回调接口
```

接口内容参看 8.4。

6.13取消一般事件的订阅

```
函数原型
    public void  unsubscribe (EventObserver observer)
参数说明
    observer: 一般事件的回调接口
```

6.14获取当前网络中的设备列表

```
函数原型
    public List<Device> getDevices ()
返回值
    Device 类型的对象的集合
```

6.15获取用户的网络列表

```
函数原型
    public List<Network> getNetworkList()
返回值
    Network 类型的对象的集合
```

7 实体类

7.1 BaseInfo

包含当前基本信息。

- String version
  - library 版本号
- String account
  - 当前缓存的账号(最近一次登录过的账号)
- List<String> userList
  - 登录成功并保存的账号集合
- String domain
  - 域名
- String vip
  - 虚拟 IP
- String vmask
  - 虚拟掩码
- String ticket
  - 登录认证的票据
- String snid
  - 当前 SmartNode 的 ID
- boolean dlt
  - DLT 是否可用

7.2 RealtimeInfo

包含当前实时信息。

- int currentStatus
  - 当前状态的标识
- int netLatency
  - 网络时延
- long onlineTime
  - 在线时长

7.3 Device

设备信息。

- String id
  - 设备 ID
- String name
  - 设备名

- String owner
  - 设备所有者
- String userid
  - 设备所有者的用户 ID
- String domain
  - 设备域名
- String vip
  - 设备虚拟 IP
- String priIp
  - 设备所在局域网分配的 IP

7.4 Network

网络信息。

- String name
  - 网络名
- String id
  - 网络 ID
- String owner
  - 网络所有者
- String uid
  - 网络所有者 ID
- boolean isCurrent;
  - 是否为当前网络

8 回调接口

8.1 ConnectStatusListenerPlus

状态事件监听的回调接口，在触发连接状态改变的时候回调相应的方法

- void onConecting()
  开始连接时回调。也可视作连接成功后（由于网络等因素造成的）的自动重连。一般而言，登录成功后除非调用退出登录的接口，SDK 不会主动断开并停止连接，异常断连后会尝试自动重连，并回调此方法，此时可自行判断是否终止重连(调用 cancelLogin())。

- void onAuthenticated()
  认证完成后回调，表示账号完成了认证。

- void onConnected()
  登录成功后回调，表示数据加载完成，已具备连接虚拟网的资质。

- void onEstablished()
  成功连接虚拟网后回调，此时可正常访问虚拟网。

- void onDisconnecting()
  连接正在断开时回调

- void onDisconnected(int reason)
  连接断开后回调，也对应首次登录前的状态，在登录失败后同样会回调此方法。一般而言，登录成功后除非调用退出登录的接口，SDK 不会主动断开并停止连接，异常断连后会尝试自动重连。
  - reason 断开原因标识

8.2 CodeResult

请求验证码的回调接口，会返回此次操作的处理结果。

- void onError (int errCode)
  执行错误时回调，参数含义参看 5.2

8.3 ResultListener

登录方法需要的执行结果回调接口，仅在登录失败时回调。

- void onError (int result)
  执行错误时回调，参数含义参看 5.3

8.4 EventObserver

一般事件监听的回调接口，在触发相应事件时回调相应的方法。

- void onRealTimeInfoChanged(RealtimeInfo info)
  实时信息变更，该方法订阅后每秒回调一次，可根据需要实时更新 UI。RealtimeInfo 内容参看 7.2。

- void onTunnelRevoke (boolean isRevoked)
  当 VPN 隧道在占用和空闲之间变化时回调
  isRevoked 为 true：
  1. 当检测出 VPN 隧道被占用时回调;
  2. 隧道被抢占时直接回调;
  3. 登录前检测是否有其他使用 VPN 的应用，有则回调。
  isRevoked 为 false：

在 VPN 隧道被占用后，每隔一定时间 sdk 会自动检测一次当前是否有其他应用正在使用 VPN 隧道，没有则回调，每次占用仅回调一次(在该回调返回 true 后的 5 秒内不会返回 false，保护期)。

9 FAQ

登录不上，提示 5——无效的用户名

检查用户名正确性，并注意用户名是否与当前连接的服务环境（正式或测试）匹配。

登录不上，提示 2018——VPN 权限被拒绝

1.请注意，Weline.io 服务是依托于 VPN 构建的，要使用 Weline.io 服务请先为应用授权 VPN。首次调用登录时会弹出系统的权限询问窗口，如若拒绝，则需要进入系统设置界面进行设置。

2.VPN 通道被其他应用占用，请关闭其他 VPN 应用后再尝试登录。如果已经登录后被其他 VPN 应用抢占通道，在关闭其他 VPN 应用后，SDK 会在回调一般事件 onTunnelRevoke(false)后自动重连。

如何使用状态监听来实现 UI 切换

先初始化一个监听对象，具体操作见注释

```
ConnectStatusListenerPlus listener = new ConnectStatusListenerPlus(){
        @Override
        public void onConnecting() {
          //此处可写入正在连接时的等待动画逻辑
          //可视作首次登录之外的自动重连，可写入系统通知的逻辑
        }

        @Override
        public void onDisconnecting() {
          //此处可写入正在退出连接时的等待动画逻辑
        }

        @Override
        public void onEstablished() {
          //此处表示登录后 VPN 隧道建立成功，可写入跳转完成连接后的界面的逻辑
        }

        @Override
```

```
        public void onDisconnected(int reason) {
          //此处表示登录失败或连接异常断开，可根据 4.1 中“登录的错误代码”对 reason 做出解释，并在此对用户进行提示
        }
    };
```

关于网络切换和掉线重连

SDK 保持的 VPN 连接状态会在断连时自动重连，常见的异常断连情况如下：

1. 网络波动
2. 切换了当前使用的网络，如更换 WIFI，4G、WIFI 之间的相互切换
3. VPN 隧道被占用（为避免不停争抢隧道，将会在隧道不被占用时重连）

在自动重连时，状态事件监听将会回调 onConnecting(),可在回调中写入等待动画的逻辑

后台运行时连接中断

部分安卓系统（EMUI）默认不支持长时间后台运行，需手动设置自启动权限。此外，还有部分安卓系统（MIUI）后台运行正常，但在连接中断后无法自动重连，需要手动设置省电策略不再限制本应用。

部分系统可以使用如下方式打开当前应用设置进行自启动权限、省电策略的修改，建议使用弹窗提示的方式引导用户点击跳转:

```
Intent intent = new Intent();
intent.setAction(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
Uri uri = Uri.fromParts("package", getActivity().getPackageName(), null);
intent.setData(uri);
startActivity(intent);
```

关于连接中断时的通知

建议在连接状态监听的 onConnecting()回调时发送一个系统通知，作为 SDK 连接断开并自动重连的提示。经实测，这个系统级的通知可以有效防止小米 MIUI10 在安卓 9.0 系统的省电策略对后台应用的使用限制。

具体通知方式可参看 Demo MyApplication.java 文件。

10 测试验证场景

01.登录、退出登录功能是否正常

调用登录、登出方法后，状态监听接口是否正确回调

02.登录成功后虚拟网是否连接、登出后虚拟网是否断开

观察状态栏中代表 VPN 的图标是否亮起，部分系统仅在通知栏提示

03.登录后能否正常访问互联网

登录后，使用其他应用或工具对互联站点进行访问或 ping 测试

04.登录后能否正常访问测试的接入网关

登录后，使用其他应用或工具对测试网关地址进行 ping 测试

05.登录后能否正常访问招商集团的应用页面

登录后，使用其他应用或工具对招商集团内网的应用页面进行访问

06.在后台运行的情况下，能否正常访问互联网

登录后，将 app 切换至后台，然后使用其他应用或工具对互联站点进行访问或 ping 测试（部分安卓系统需要后台权限）

07.在后台运行的情况下，能否正常访问测试的接入网关

登录后，将 app 切换至后台，然后使用其他应用或工具对测试网关地址进行 ping 测试（部分安卓系统需要后台权限）

08.在后台运行的情况下，能否正常访问招商集团应用页面

登录后，将 app 切换至后台，然后使用其他应用或工具对招商集团内网的应用页面进行访问（部分安卓系统需要后台权限）

09.登录后，关闭开启网络后能否正常自动重连

10.登录后，由 4G 切换至 WIFI 后能否正常自动重连

11.登录后，由 WIFI 切换至 4G 后能否正常自动重连

12.登录后，由 WIFI 切换至 WIFI 后能否正常自动重连

13.在线时长、时延是否正常

登录后，于一般事件监听中回调的 RealtimeInfo 数据中获取在线时长、网络食盐。观察在线时长是否真确累计，时延波动是否正常（自动重连时的大幅波动属正常现象）