智营网优

SDK文档

产品文档

腾讯云

文档目录

SDK文档
- 概述
- Android SDK
- iOS SDK
- Unity SDK
- 签名与鉴权

SDK文档

概述

最近更新时间：2019-07-17 17:24:44

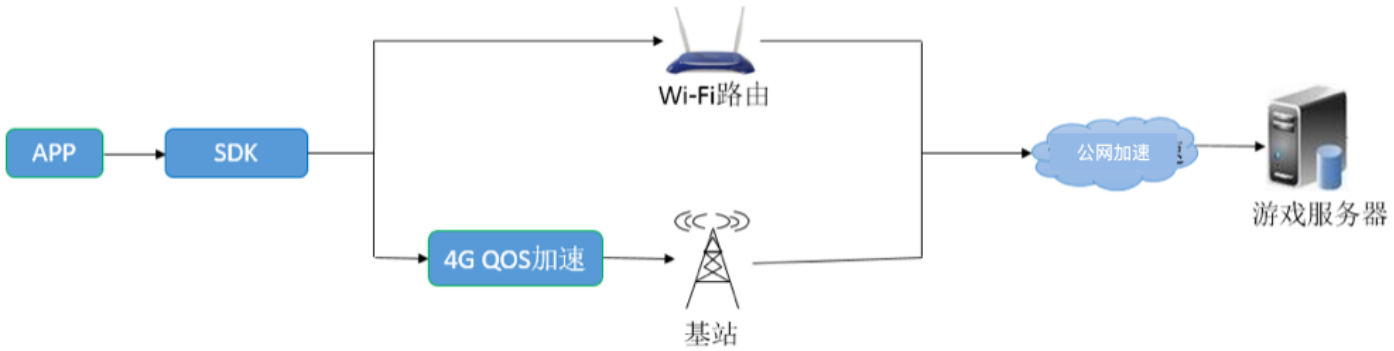
SDK 简介

智营网优 SDK（以下简称 SDK）是手机应用加速和网络诊断组件，应用集成 SDK 后，在应用启动后，调用 SDK 加速接口，SDK 可智能选择加速路径，降低通信网络时延；调用 SDK 诊断接口，可检测网络卡顿原因，并给出相应解决方法的提示，从而提升应用品质，给用户一个更畅快的 App 体验。

架构图

SDK 分为加速和诊断两个部分，并提供相应的 API 接口。

加速

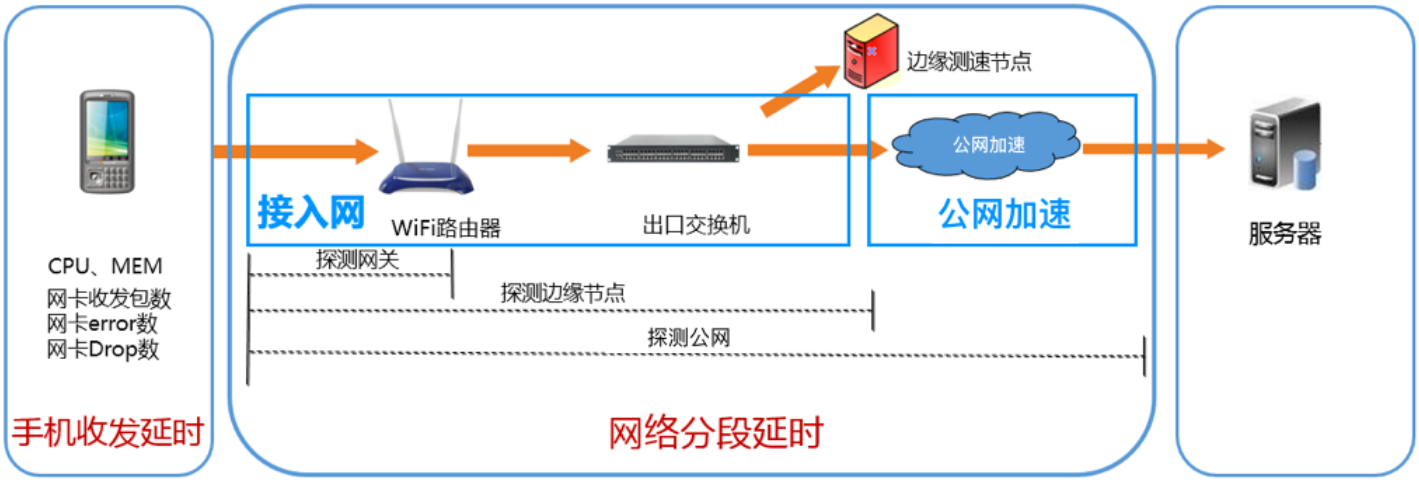


1. SDK 加速分为接入网加速和公网加速。
2. 接入网加速主要针对 4G 网络。
3. 公网采用 CDN 提供的最佳网络加速路径。

网络诊断

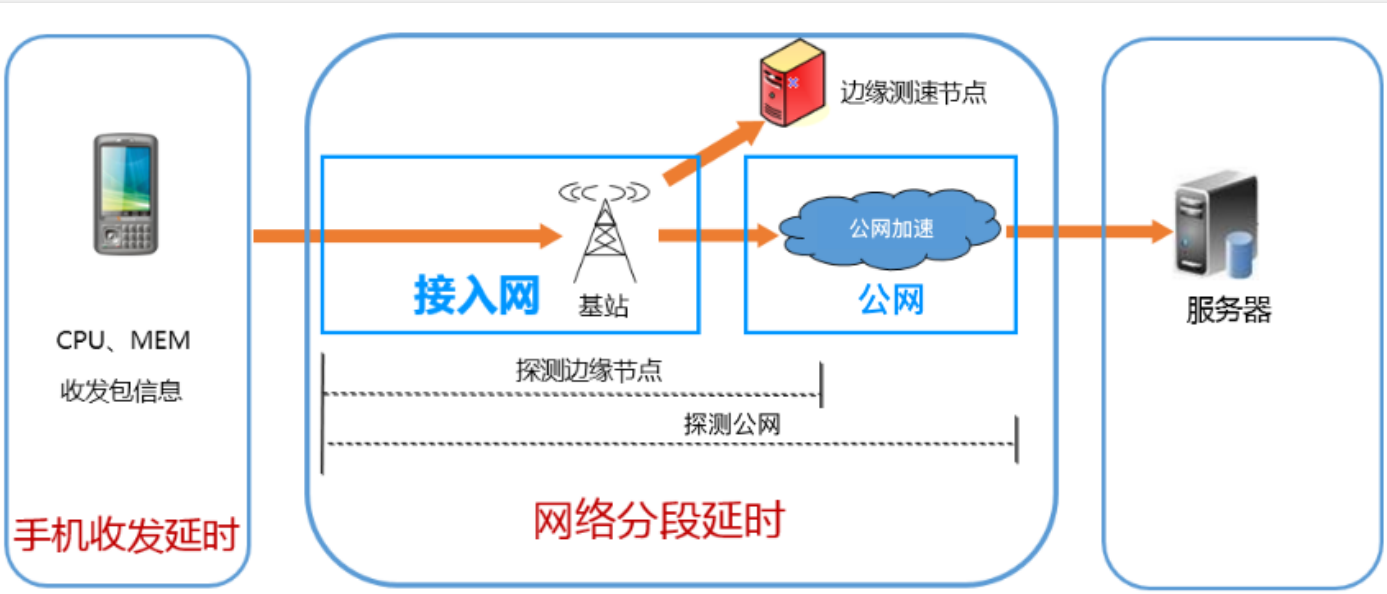
网络诊断分为 Wi-Fi 网络诊断和移动网络诊断。

Wi-Fi 网络诊断过程



Wi-Fi 下的网络诊断主要通过对网关、边缘节点服务器、公网的探测速率来判断网络质量。王者荣耀 Wi-Fi 网络诊断示例如下：

移动网络诊断过程



移动网络诊断主要通过对边缘节点服务器、公网的探测速率来判断网络质量。王者荣耀 4G 移动网络诊断示例如下：

Android SDK

最近更新时间：2019-11-25 18:51:44

注意：
请先单击 智营网优申请页 进行申请。

接入流程

获取 SDK

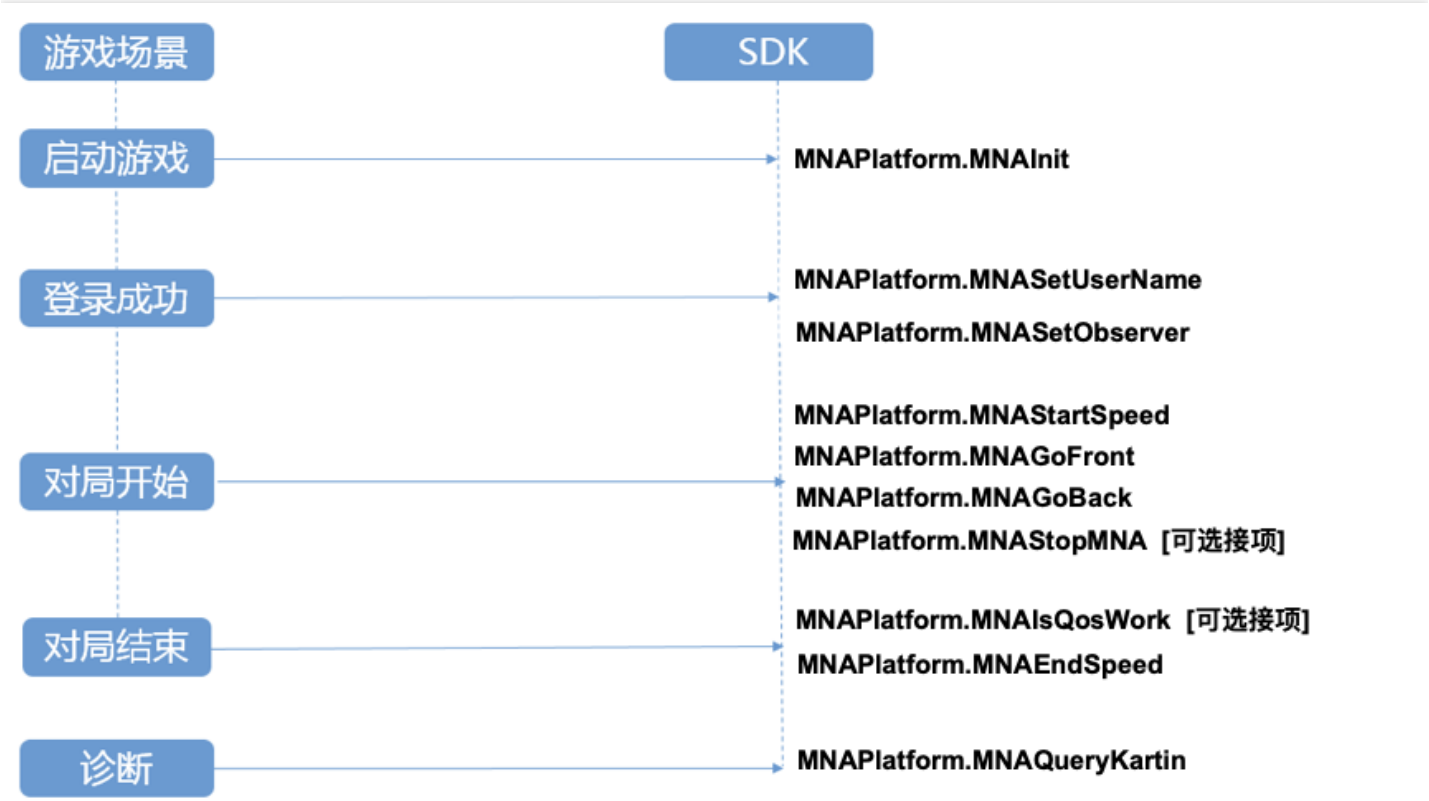
1. 登录 智营网优控制台，单击左侧菜单【SDK 下载】。
2. 请按系统平台选择对应版本的 SDK，并单击【下载】。

配置 SDK

1. 将下列文件导入到项目中：
   - MNA_ANDROID.jar
   - beacongsdk_xxx.jar
   - libgsdk.so
2. 修改项目中的 AndroidManifest.xml 文件，添加以下权限：

```
android:permission.INTERNET
android:permission:ACCESS_NETWORK_STATE
android:permission:READ_PHONE_STATE
android:permission:ACCESS_WIFI_STATE
android.permission.ACCESS_FINE_LOCATION
```

接入步骤



1. 应用启动时，调用初始化 MNAInit 。
2. 应用登录成功后，需要调用 MNASetUserName 设置 openid 。
3. 应用登录成功后，尽可能早地调用 MNAStartSpeed ，用户因网络或者其他异常无法连接到服务器，重新进行连接时，为保证加速效果，需要再次调用 MNAStartSpeed ；当应用切换到前台时，调 MNAGoFront ; 当应用切换到后台时，调用 MNAGoBack ；当应用不想进行加速，但依然想保留 SDK 的网络探测功能时可调用 MNAStopMNA 函数。
4. 应用退出时，先调用 MNAIsQosWork 函数获取 Qos 保障标识，再调用 MNAEndSpeed 结束当局加速，并上报网络质量数据。
5. 进入游戏大厅后可通过调用 MNAQueryKartin 函数来对网络进行诊断。

API 介绍

API 接口由 com.tencent.mna 包下的 MNAPlatform 类提供，该类将所有方法都设计为静态函数，开发者可用类名直接进行调用。

初始化

初始化 SDK

```
public static void MNAInit(Context context, string qqappid, bool debug, int zoneid, boolean env, boolean useBattery, String tcloudkey)
```

参数	含义
context	上下文环境或者当前 activity
qqappid	惟一标识该应用，对应腾讯云加速服务的“游戏 ID”
debug	控制 log 的输出方便联调
zoneid	玩家大区 ID
env	云控正式环境，默认直接填 true 即可
useBattery	电量统计信息，默认直接填 false 即可
tCloudKey	腾讯云申请的 key 值，即“密钥 KEY”

设置用户信息

```
public static void MNASetUserName(int userType, string openid)
//userType：此参数为 int 型，与 MSDK 中定义账号类型一致
//openid：用户的 openid
```

类型	取值
ePlatform_None	0
ePlatform_Weixin	1
ePlatform_QQ	2
ePlatfrom_WTLogin	3
ePlatform_QQHall	4
ePlatform_Guest	5

开启加速异步回调

```
public static void MNASetObserver(MNAObserver d)
```

需要在初始化后设置 `MNAObserver` 参数，参数为委托类型，定义如下：

```
public interface MNAObserver {
public abstract void OnStartSpeedNotify(int type, int flag, String desc);
public abstract void OnQueryKartinNotify(String _tag, int _flag, String _desc,
int _jump_network, int _jump_signal, int _jump_router, int _router_status, String _router_desc, int _jump_export, int _export_status, String _export_desc, int _jump_terminal, int _terminal_status, String _terminal_desc, int _jump_proxy, int _jump_edge, String _signal_desc, int _signal_status, int _jump_direct, int _direct_status, String _direct_desc,int _network_status, String _network_desc, int _wifi_num);
public abstract void OnBatteryChangedNotify(int charging, int level);
}
```

设置 MNAObserver 类 OnStartSpeedNotify 函数（网络加速回调函数）：

```
void OnStartSpeedNotify(int type, int flag, String desc);
```

参数	含义
type	hook 类型
flag	启动加速结果返回值
desc	启动加速结果描述

其中 flag 值及对应描述如下：

错误码	具体含义
2	无需开启加速
1	正在执行 startspeed
0	成功
-1	初始化时获取 JVM 失败
-2	unknown/2G 或者无网络下
-3	请求中控失败
-4	未达到加速条件
-5	hook 失败
-6	GameServer DNS 请求失败
-7	没有成功加载 so 库

错误码	具体含义
-8	中控域名 DNS 解析失败
-9	StartSpeed 请求阶段超时
-10	StartSpeed 测速阶段超时
-11	请求调度失败
-12	请求协商失败

设置 MNAObserver 类 OnQueryKartinNotify 函数（网络诊断结果回调函数）：

void OnQueryKartinNotify(String tag, **int** flag, String desc, **int** jump_network, **int** jump_signal, **int** jump_router, **int** router_status, String router_desc, **int** jump_export, **int** export_status, String export_desc, **int** jump_terminal, **int** terminal_status, String terminal_desc, **int** jump_proxy, **int** jump_edge, String signal_desc, **int** signal_status, **int** jump_direct, **int** direct_status, String direct_desc,**int** network_status, String network_desc, **int** wifi_num);

参数	含义
type	游戏传入的 Tag
flag	查询成功标识，若为0则成功
desc	查询flag的具体描述
jump_network	当时网络类型0：无网络、1：2G、2：3G、 3：4G、 4： Wi-Fi
jump_signal	信号强度
jump_router ping	路由时延
router_status ping	状态，0表示绿色，时延低；2表示红色，时延高
router_desc ping	描述
jump_export	宽带或基站出口时延
export_status	宽带出口和基站出口状态，0表示绿色，时延低；2表示红色，时延高
export_desc	宽带出口和基站出口描述
jump_terminal	手机终端数
terminal_status wifi	终端数状态，0表示绿色终端数少；2表示红色，时延高

参数	含义
terminal_desc wifi	终端数描述
jump_proxy ping	代理时延
jump_edge ping	边缘时延
signal_desc	信号强度描述
signal_status	信号状态，0表示绿色，信号强；1表示黄色，信号弱；2表示红色，信号极弱
jump_direct	直连测速时延
direct_status	直连测速时延状态，0表示绿色时延低；2表示红色，时延高
direct_desc	直连状态的描述
network_status	网卡状态
network_desc	网卡情况具体描述
wifi_num wifi	终端数

设置 `MNAObserver` 类 `OnBatteryChangedNotify` 电量监控回调函数

```
void OnBatteryChangedNotify(int charging, int level);
```

参数	含义
charging	电池充放电状态
level	电池电量

加速

开启加速器引擎 StartSpeed

```
public static void MNAStartSpeed(string vip, int vport, int htype, string hookModules, int zoneid, int stopMNA, int timeout, String pvpid)
```

本函数被调用后将开始异步对所有加速节点进行测速，判断是否执行加速。整个过程需要5 - 6秒。完成后会回调 `MNAObserver` 函数。

参数	含义

参数	含义
vip	游戏服务器地址（IP 或域名，强烈建议使用 IP）。
vport	游戏服务器端口
htype	本参数决定 HOOK 的函数种类，取值有如下： • htype = 1：表示只处理 sendto() 和 recvfrom() ，用于核心协议是 UD 且使用这两个函数的游戏。 • htype = 3：表示只处理 connect 和 send 函数，用于核心协议时 UDP 且使用这两个函数的游戏。
hookModules	hookModules 指定要 HOOK 的动态链接库，多个库名用英文（半角）逗号分开。若使用 apollo 的网络通信模块，则填 libapollo.so；若使用 C# 的网络通信模块，则填 libmono.so。
stopMNA	默认值为 0 ； 1 表示强制关闭加速功能，保留网络诊断功能
timeout	默认值为 0 ；设置启动阶段超时时间，单位为毫秒，当 timeout<=0 时，表示不设置启动超时
pvpid	游戏对局唯一 ID，应用直接填 "UNKNOWN"

通知加速引擎：游戏目前在前台

```
public static void MNAGoFront()
```

当游戏切换到前台时,必须调用此函数，通知加速引擎。

通知加速引擎：游戏目前切换到后台

```
public static void MNAGoBack()
```

当游戏切换到后台（例如被其它应用遮挡），必须调用此函数，通知加速引擎。

强行关闭加速

```
public static void MNAStopMNA (string vip, int vport)
//参数： vip、vport 游戏服务器地址,跟 startspeed 保持一致
```

强行关闭加速功能，对局过程中的网络探测功能。

正常结束加速器引擎

```
public static void MNAEndSpeed(string vip, int vport)
//参数： vip 游戏服务器地址,跟 startspeed 保持一致
```

```
//参数： vport 游戏服务器端口
```

查询 4G QoS 是否保障

```
public static int MNAIsQosWork()
```

返回 QoS 是否有保障成功：
需要把此状态信息在游戏体验数据中记录并上报。4G QOS 在移动 4G 网络下才会生效，一般在结束加速 `EndSpeed` 时调用，返回值具体含义如下，这个可以用于游戏上线后核对 4G QoS 加速效果。

返回值	具体含义
1	保障成功
0	保障失败（默认值）
-1	非 4G
-2	不满足保障条件
-3	云控配置关闭
-4	云控返回 errno 非 0
-5	解析云控数据失败
-6	获取本地 IP 失败
-7	运营商返回 errno 非 0
-8	解析运营商数据失败

网络诊断

设置实时网络诊断

1. 需要在初始化后设置 `MNAKartinObserver`

```
public static void MNASetObserver(MNAObserver d)
```

2. 调用实时网络诊断

```
public static void MNAQueryKartin(string tag)
//参数：tag，作为标识每一次查询的 ID 。
```

网络诊断是一个耗时的过程，大概有4 - 8秒钟，因此 MNA 使用异步返回来实现。调用此函数后，查询结果会通过回调 KartinObserver 返回。

附录

OnQueryKartinNotify 回调结果参数取值及其说明如下：

- 字段：flag。
- 关键名称：查询结果标识。

取值	含义
0	查询成功
-1	表明无网络
-2	表明请求云控失败，重新调用一次
-3	表明初始化 SDK 失败
-4	当前网络类型发生了变化，请稍后再试
-5	2G 网络不适合游戏，无法检测

- 字段：jump_network。
- 关键名称：网络类型。

取值	含义
0	无网络或网络类型无法识别
1	2G
2	3G
3	4G
4	Wi-Fi

- 字段：jump_signal。
- 关键名称：信号强度（仅 Wi-Fi）。

取值	含义
-1	获取强度失败，请稍后再试
0..4	Wi-Fi 信号强度

- 字段：jump_router。
- 关键名称：路由器时延（仅 Wi-Fi）。

取值	含义
-1	不支持路由延迟查询
-2	获取路由器延迟失败，请稍后再试
0..1000	当前路由器的延迟值

- 字段：jump_terminal。
- 关键名称：共享 Wi-Fi 设备数（仅 Wi-Fi）。

取值	含义
-1	仅在 Wi-Fi 模式下支持共享 Wi-Fi 设备查询
0..254	链接相同 Wi-Fi 的设备数

- 字段：jump_export。
- 关键名称：宽带出口时延。

取值	含义
-1	获取社区延迟失败，请稍后再试
-2	抱歉，当前所在区域网络不支持社区宽带延迟查询
0..500	带宽出口时延值

- 字段：jump_direct。
- 关键名称：直连时延。

取值	含义
-1	获取直连延迟失败，请稍后再试
0..800	网络时延值

- 字段：netinfo_desc。
- 关键名称：直连时延。

取值	含义
String	当前网卡有丢包或错包，不适合游戏。

具体设置请参考王者荣耀的示例：（ 红色字体为备注 ）
Wi-Fi 直连环境下图示如下：

4G 直连环境下图示如下：

iOS SDK

最近更新时间：2019-12-05 19:23:04

> 注意：
> 请先单击 智营网优申请页 进行申请。

接入流程

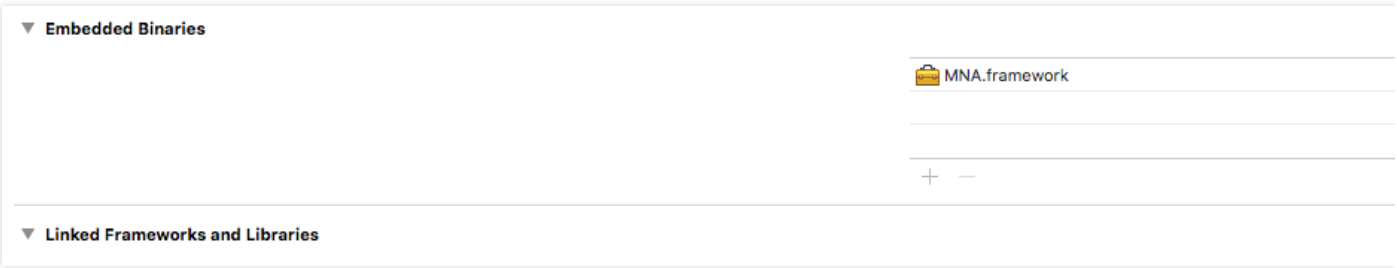
获取 SDK

1. 登录 智营网优控制台，单击左侧菜单【SDK 下载】。
2. 请按系统平台选择对应版本的 SDK，并单击【下载】。

配置 SDK

> 注意：
> iOS SDK 为动态库，仅支持 iOS8 以上。

1. 在 Embedded Binaries 中引入 SDK：



2. 在 Linked Frameworks and Libraries 中引入 GBeaconAPI_Base.framework。
3. 添加系统库：
   - SystemConfiguration.framework
   - libz.dylib
   - libstdc++.dylib（或 libc++.dylib ）
   - libsqlite3.dylib
   - CoreTelephoney.framework
   - Security.framework

- AdSupport.framework
- CoreLocation.framework
- MobileCoreService.framework
- UIKit.framework

4. 引入文件：
   - MyMNAObserver.h
   - MyMNAObserver.mm
5. 在 info.plist 中添加如下设置：

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>
```

注意：
在 NSAppTransportSecurity 下，只需设置 NSAllowsArbitraryLoads 为 YES 即可，请勿再添加其他配置，例如 NSAllowsArbitraryLoadsForMedia

6. 在 AppDelegate.mm 文件中，引入头文件：

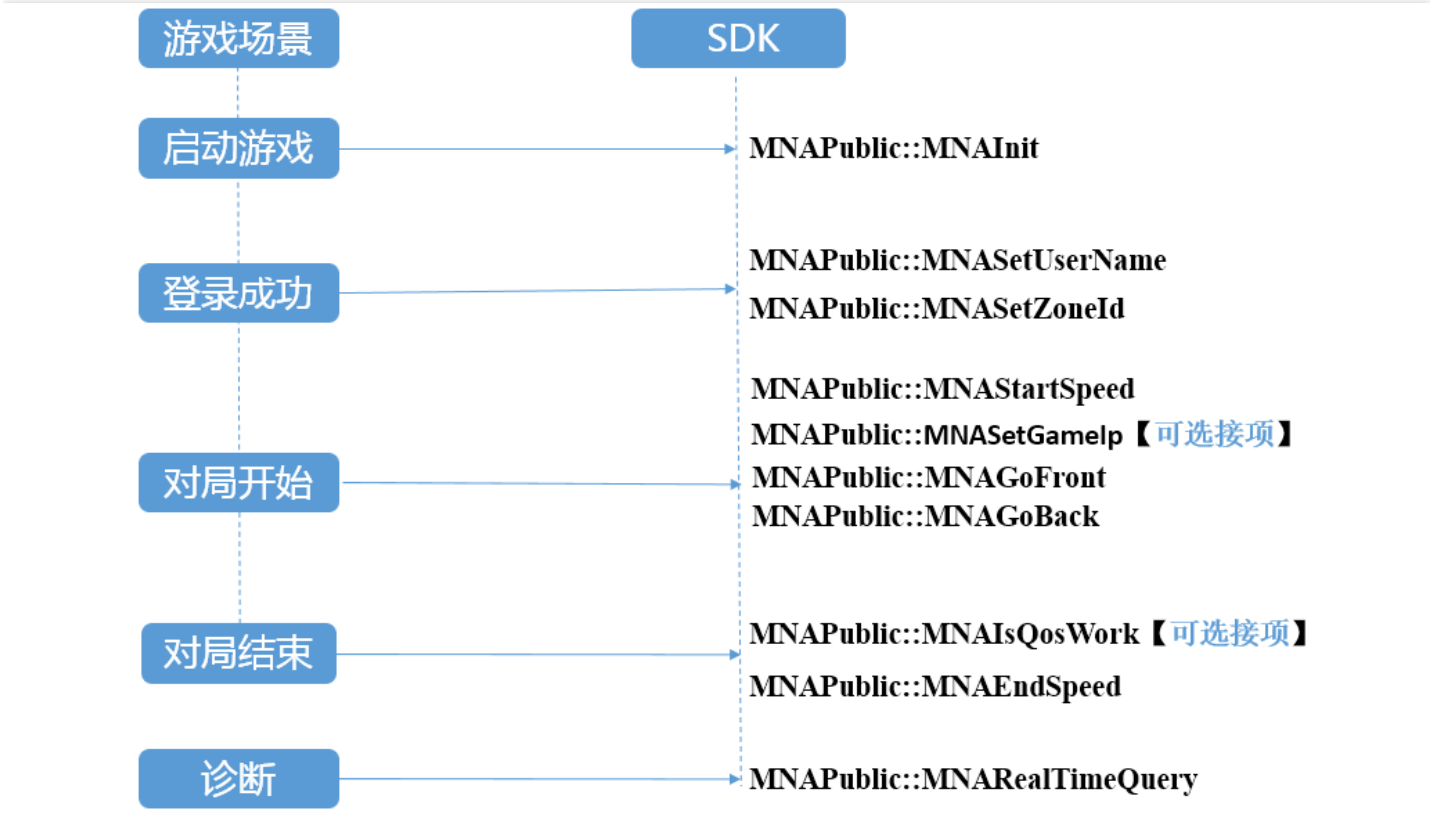
```
#import <MNA/MNAPublic.h>
#import “MyMNAObserver.h”
```

并在 application:didFinishLaunchingWithOptions 函数中设置回调函数，即粘帖如下代码：

```
MNAPublic * mna = MNAPublic::GetInstance();
MyMNAObserver * observer = MyMNAObserver::GetInstance();
mna->MNASetObserver(observer);
```

7. 完成后，按照所需调用相应接口即可。

接入步骤



1. 应用启动时，调用初始化 MNAInit ，并设置回调函数 MNASetObserver 。
2. 应用登录成功后，需要调用 MNASetUserName 设置 openid ，可通过 MNASetZoneId 设置 zone id 。
3. 对局开始前调用 MNAStartDiagnose 当游戏服务器 IP 发生变化时可调用 MNASetGameIp 来设置游戏服务器 IP，不发生变化时不需要调用；当应用切换到前台时，调用 MNAGoFront ;当应用切换到后台时，调用 MNAGoBack 。
4. 对局结束时，先调用 MNAIsQosWork 函数获取 Qos 保障标识，再调用 MNAEndSpeed 结束当局加速，并上报网络质量数据。
5. 进入游戏大厅后可通过调用 MNARealTimeQuery 函数来对网络进行诊断。

API 介绍

使用单例模式调用，调用示例：

```
MNAPublic* mna = MNAPublic::GetInstance();
mna->MNAInit("appid_test", true, 0, true, false, "");
```

初始化

初始化 SDK

```
void MNAInit(const char * appid, bool isDebug, int zoneid, bool isReleaseEnv, bool isBatteryNotify, const char * tCloudKey);
```

参数	含义
appid	惟一标识该应用,即腾讯云控制台加速服务的游戏 ID
isDebug	控制 log 的输出方便联调
zoneid	玩家大区 ID
isReleaseEnv	云控正式环境，默认直接填 true 即可
isBatteryNotify	电量统计信息，默认直接填 false 即可
tCloudKey	腾讯云申请的 key 值

设置回调函数

```
void MNASetObserver(MNAObserver * observer);
//参数：observer，回调函数实例。
```

设置用户信息

```
void MNASetUserName(int platform, const char * openId);
//platform：此参数为 int 型，与 MSDK 中定义账号类型一致
//openId：用户的 openid
```

类型	取值
ePlatform_None	0
ePlatform_Weixin	1
ePlatform_QQ	2
ePlatfrom_WTLogin	3
ePlatform_QQHall	4
ePlatform_Guest	5

加速

开启加速器引擎 MNAStartSpeed

```
void MNAStartSpeed(const char * vip, int vport, int htype, const char * module, int zoneid, int stop MNA, const char * pvpid, int timeout);
```

参数	含义
vip	游戏服务器地址（IP 或域名，强烈建议使用 IP）
vport	游戏服务器端口
htype	本参数决定 HOOK 的函数种类，取值有如下：htype = 1：表示只处理 sendto() 和 recvfrom()，用于核心协议是 UD，且使用这两个函数的游戏。
hookModules	指定要 HOOK 的动态链接库，填""即可
stopMNA	默认值为0；为1表示强制关闭加速功能，保留网络诊断功能。
pvpid	对局唯一 ID，用来定位对局的网络问题
timeout	默认值为0；设置启动阶段超时时间，单位为毫秒，当 timeout<=0 时，表示不设置启动超时。

本函数被调用后将开始异步对所有加速节点进行测速，判断是否执行加速。整个过程需要5 - 6秒。完成后会回调 MNAObserver 函数。

设置游戏服务器 IP（选接项）

```
void MNASetGameIp(const char *gameip);
```

参数	含义
gameIp	游戏服务器地址

当游戏在对局中因为网络或者别的原因导致游戏服务器 IP 发生改变时，直接调用该接口设置游戏服务器 IP 即可，当对局中 IP 不会发生变化时不需要接入，目前主要是境外业务需要调用该接口，中国内地（大陆）业务不用接。

通知加速引擎：游戏目前在前台

```
(void)MNAGoFront;
```

当游戏切换到前台时,调用此函数。

通知加速引擎：游戏目前切换到后台

```
(void)MNAGoBack;
```

当游戏切换到后台（例如被其它应用遮挡），必须调用此函数，通知加速引擎。

正常结束加速器引擎

```
void MNAEndSpeed(const char * vip, int vport);
//参数： vip 游戏服务器地址,跟 startspeed 保持一致
//参数： vport 游戏服务器端口
```

查询 4G QoS 是否保障

```
(int)MNAIsQOSWork;
```

返回 QoS 是否有保障成功：需要把此状态信息在游戏体验数据中记录并上报。4G QoS 在移动 4G 网络下才会生效，一般在结束加速 `MNAEndSpeed` 时调用，返回值具体含义如下，这个可以用于游戏上线后核对 4G QOS 加速效果。

返回值	具体含义
1	保障成功
0	保障失败（默认值）
-1	非 4G
-2	不满足保障条件
-3	云控配置关闭
-4	云控返回 errno 非 0
-5	解析云控数据失败
-6	获取本地 IP 失败
-7	运营商返回 errno 非 0
-8	解析运营商数据失败

网络诊断

设置实时网络诊断

```
void MNARealTimeQuery(const char * tag);
//参数`tag`，作为标识每一次查询的 ID。
```

结果由 MNAObserver 函数的 OnQueryKartinNotify 返回。

返回结果以分号拼接，含义依次如下：

```
tag; // 游戏传入的Tag
flag; // 查询成功标识，若为0则成功
desc; // 查询flag的具体描述
jump_network; // 当时网络类型0: 无网络,1: 2G, 2: 3G, 3: 4G, 4: wifi
jump_signal; // 信号强度
jump_router; // ping路由时延
router_status; // ping状态,0表示绿色，时延低；2表示红色，时延高
router_desc; // ping描述
jump_export; // 宽带或基站出口时延
export_status; // export状态,0表示绿色，时延低；2表示红色，时延高
export_desc; //宽带出口和基站出口描述
jump_terminal; // wifi终端数
terminal_status; // terminal状态,0表示绿色终端数少；2表示红色，时延高
terminal_desc; //wifi终端数描述
jump_proxy; // ping代理时延
jump_edge; // ping边缘时延
signal_desc; //信号强度描述
signal_status; //信号强度状态
jump_direct; // 直连测速时延
direct_status; // 直连状态,0表示绿色时延低；2表示红色，时延高
direct_desc; // 直连状态的描述
netinfo_status; // // 网卡状态, 0表示绿色网卡无问题；2表示红色网卡有问题
```

附录

MNAKartinRet 关键参数取值及其说明如下：

- 字段：flag。
- 关键名称：查询结果标识。

取值	含义
0	查询成功
-1	表明无网络
-2	表明请求云控失败，重新调用一次
-3	表明初始化 SDK 失败

取值	含义
-4	当前网络类型发生了变化，请稍后再试
-5	2G 网络不适合游戏，无法检测

- 字段：jump_network。
- 关键名称：网络类型。

取值	含义
0	无网络或网络类型无法识别
1	2G
2	3G
3	4G
4	Wi-Fi

- 字段：jump_signal。
- 关键名称：信号强度（仅 Wi-Fi）。

取值	含义
-1	获取强度失败，请稍后再试
0..4	Wi-Fi 信号强度

- 字段：jump_router。
- 关键名称：路由器时延（仅 Wi-Fi）。

取值	含义
-1	不支持路由延迟查询
-2	获取路由器延迟失败，请稍后再试
0..1000	当前路由器的延迟值

- 字段：jump_terminal。

- 关键名称：共享 Wi-Fi 设备数（仅 Wi-Fi）。

取值	含义
-1	仅在WIFI模式下支持共享 Wi-Fi 设备查询
0..254	链接相同 Wi-Fi 的设备数

- 字段：jump_export。
- 关键名称：宽带出口时延。

取值	含义
-1	获取社区延迟失败，请稍后再试
-2	抱歉，当前所在区域网络不支持社区宽带延迟查询
0..500	带宽出口时延值

- 字段：jump_direct。
- 关键名称：直连时延。

取值	含义
-1	获取直连延迟失败，请稍后再试
0..800	网络时延值

- 字段：netinfo_desc。
- 关键名称：直连时延。

取值	含义
String	当前网卡有丢包或错包，不适合游戏。

具体设置请参考王者荣耀的示例：（ 红色字体为备注 ）

Wi-Fi 直连环境下图示如下：

4G 直连环境下图示如下：

Unity SDK

最近更新时间：2019-09-20 17:37:55

注意：
请先单击 智营网优申请页 进行申请。

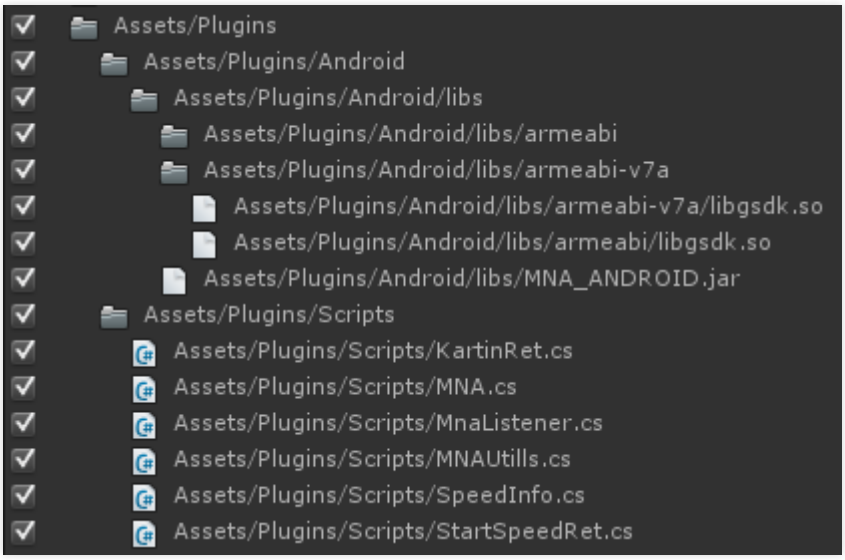
接入流程

获取 SDK

1. 登录 智营网优控制台，单击左侧菜单【SDK 下载】。
2. 请按系统平台选择对应版本的 SDK，并单击【下载】。

配置 SDK

1. 将下列文件导入到项目中：
   - MNA_ANDROID.jar
   - beacongsdk_android_v2.7.0_gsdk_cover.aar
   - libgsdk.so
2. 将Assets/Plugins/Scripts下的脚本拷贝到项目中；



3. 修改项目中的 AndroidManifest.xml 文件，添加以下权限：

```
android:permission.INTERNET
android:permission:ACCESS_NETWORK_STATE
```

```
android:permission:READ_PHONE_STATE
android:permission:ACCESS_WIFI_STATE
android.permission.ACCESS_FINE_LOCATION
```

安装包结构

目录	说明
MNA\MNA.framework	适用“Build Setting->C++ Language Dialect”配置为 GNU++98，“Build Setting->C++ Standard Library”为“libstdc++(GNU C++ standard library)”的工程。
MNA_C11\MNA.framework	适用于该两项配置分别为“GNU++11”和“libc++(LLVM C++ standard library with C++11 support)”的工程。

操作步骤

1. 在 Linked Frameworks and Libraries 中引入 MNA\MNA.framework 或 MNA_C11\MNA.framework。

> 注意：
> 腾讯云客户还需引入GBeaconAPI_Base.framework。

2. 添加系统库：
   - SystemConfiguration.framework
   - libz.dylib
   - libstdc++.dylib （或 libc++.dylib ）
   - libsqlite3.dylib
   - CoreTelephoney.framework
   - Security.framework
   - AdSupport.framework
   - CoreLocation.framework
   - MobileCoreService.framework
   - UIKit.framework
3. 在 Other linker flag 里加入 -ObjC 标志。
4. 在 info.plist 中添加如下设置：

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
```

```
<true/>
</dict>
```

> 注意：
> 在 NSAppTransportSecurity 下，只需设置 NSAllowsArbitraryLoads 为 YES 即可，请勿再添加其他配置，如 NSAllowsArbitraryLoadsForMedia。

5. 引入文件：
   - MyMNAObserver.h
   - MyMNAObserver.mm
6. 在 AppDelegate.mm 或者 Classes/UnityAppController.mm 文件中，引入头文件：

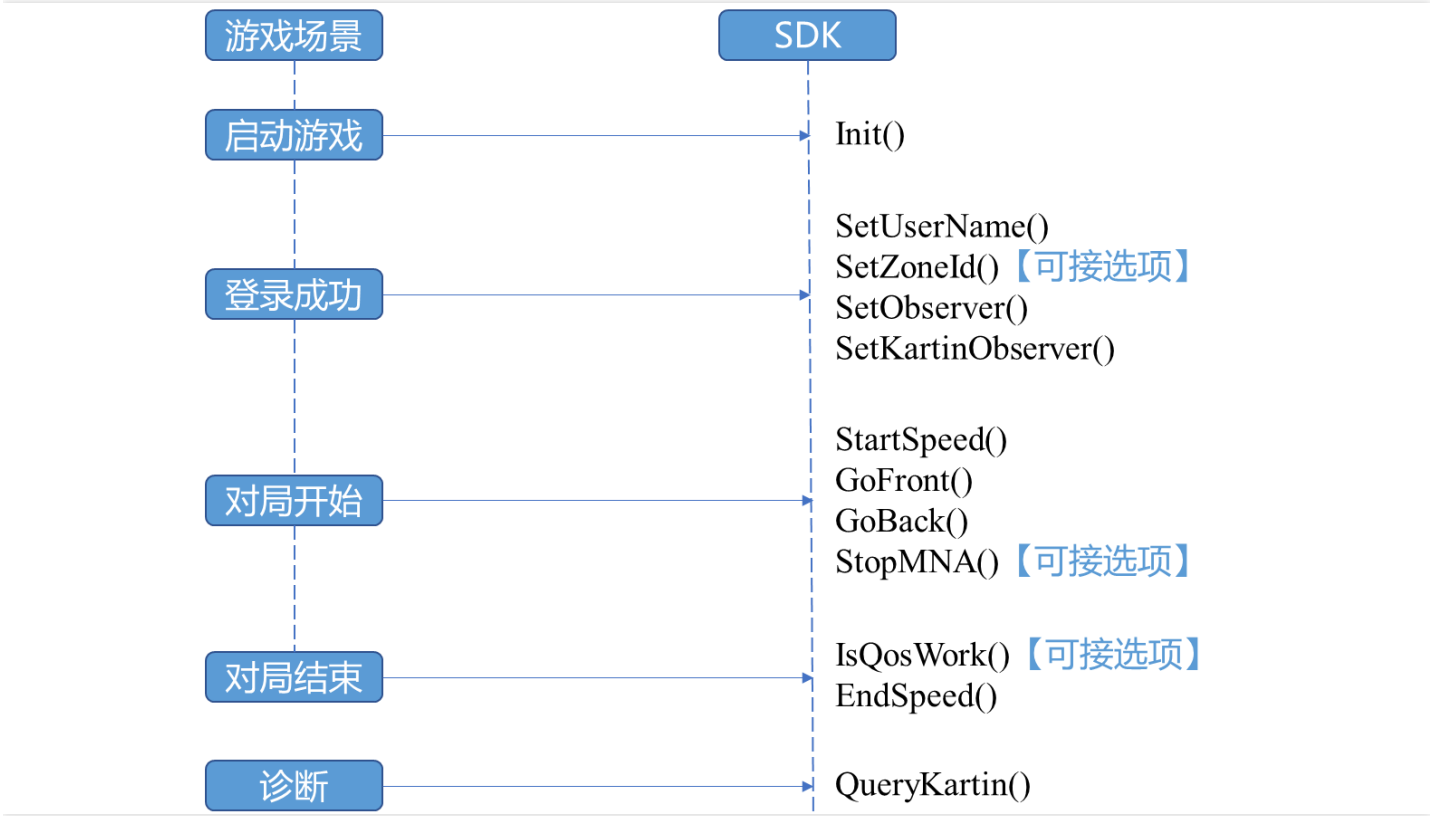
```
#import <MNA/MNAPublic.h>
#import “MyMNAObserver.h”
```

并在 application:didFinishLaunchingWithOptions 函数中设置回调函数，即粘帖如下代码：

```
MNAPublic * mna = MNAPublic::GetInstance();
MyMNAObserver * observer = MyMNAObserver::GetInstance();
mna->MNASetObserver(observer);
```

7. 完成后，按照所需在 Unity 调用相应接口即可。

接入步骤



1. 游戏启动时，调用初始化 Init 。
2. 游戏登录成功后，需要调用 SetUserName 设置 openid ；并设置加速结果回调 SetObserver ；使用 SetZoneId 来设置用户的zone id；如果需要接入诊断部分，则还需要调用 SetKartinObserver 函数。
3. 游戏对局开始时，获取到对局中所使用到的域名或 IP 后，尽可能早的调用 StartSpeed ，玩家因网络或者其他异常无法连接到游戏，重新进行连接时，为保证加速效果，需要再次调用 StartSpeed ；当游戏切换到前台时，调用 GoFront；当游戏切换到后台时，调用 GoBack ；当游戏不想进行加速，但依然想保留 SDK 的网络探测功能时可调用 StopMNA 函数。
4. 游戏退出对局时，先调用 IsQosWork 函数获取 QoS 保障标识，再调用 EndSpeed 结束当局加速，并上报游戏网络质量数据。
5. 进入游戏大厅后可通过调用 QueryKartin 函数来对网络进行诊断。
   接入步骤注意事项请参见下文 附录1。

API 介绍

API 接口由 com.tencent.mna 包下的 MNA 类提供，该类将所有方法都设计为静态函数，开发者可用类名直接进行调用。

初始化

初始化 SDK

```
public static void Init(string qqappid, bool debug, int zoneid, bool isReleaseEnv, bool useBatteryNotify, string tCloudKey)
```

参数	含义
qqappid	惟一标识该应用,即腾讯云控制台加速服务对应的游戏 ID
debug	控制 log 的输出方便联调，默认直接填 false 即可
zoneid	玩家大区 ID
isReleaseEnv	云控正式环境，默认直接填 true 即可
useBatteryNotify	电量统计信息，默认直接填 false 即可
tCloudKey	腾讯云申请的 key 值

设置用户信息

```
public static void SetUserName(int userType, string openid)
//userType：此参数为 int 型，与 MSDK 中定义账号类型一致
//openid：用户的 openid
```

类型	取值
ePlatform_None	0
ePlatform_Weixin	1
ePlatform_QQ	2
ePlatfrom_WTLogin	3
ePlatform_QQHall	4
ePlatform_Guest	5

开启加速异步回调

```
public static void SetObserver(GSDKObserver d)
```

需要在初始化后设置 `GSDKObserver` 参数，参数为委托类型，定义如下：

```
public delegate void GSDKObserver(StartSpeedRet ret);
public class StartSpeedRet {
```

```
public int flag; // 启动加速结果返回值
public string desc; // 启动加速结果描述
public int type; // hook 类型
}
```

其中 flag 值及对应描述如下：

错误码	具体含义
2	无需开启加速
1	正在执行 startspeed
0	成功
-1	初始化时获取 JVM 失败
-2	unknown/2G 或者无网络下
-3	请求中控失败
-4	未达到加速条件
-5	hook 失败
-6	GameServer DNS 请求失败
-7	没有成功加载 so 库
-8	中控域名 DNS 解析失败
-9	StartSpeed 请求阶段超时
-10	StartSpeed 测速阶段超时
-11	请求调度失败
-12	请求协商失败

加速

开启加速器引擎 StartSpeed

```
public static void StartSpeed(string vip, int vport, int htype, string hookModules, int zoneid, int sto
pWlanAcc, string pvpid, int timeout)
```

参数	含义

参数	含义
vip	游戏服务器地址（IP 或域名，强烈建议使用 IP）
vport	游戏服务器端口
htype	本参数决定 HOOK 的函数种类，取值有如下： • htype = 1：表示只处理 sendto() 和 recvfrom() ，用于核心协议是 UD 且使用这两个函数的游戏。 • htype = 3：表示只处理 connect 和 send 函数，用于核心协议时 UDP 且使用这两个函数的游戏。
hookModules	hookModules 指定要HOOK的动态链接库，多个库名用英文（半角）逗号分开。若使用 apollo 的网络通信模块，则填 libapollo.so；若使用 C# 的网络通信模块，则填 libmono.so。
stopWlanAcc	默认值为0；为1表示强制关闭公网加速功能，4G QOS加速功能、网络诊断功能不受影响。
pvpid	对局唯一 ID，用来定位对局的网络问题；
timeout	默认值为0；设置启动阶段超时时间，单位为毫秒，当 timeout<=0 时，表示不设置启动超时。

本函数被调用后将开始异步对所有加速节点进行测速，判断是否执行加速。整个过程需要5 - 6秒。完成后会回调 GSDKObserver 函数。

通知加速引擎：游戏目前在前台

```
public static void GoFront()
```

当游戏切换到前台时,必须调用此函数，通知加速引擎。

通知加速引擎：游戏目前切换到后台

```
public static void GoBack()
```

当游戏切换到后台（例如被其它应用遮挡），必须调用此函数，通知加速引擎。

强行关闭加速

```
public static void StopMNA (string vip, int vport)
//参数： vip 游戏服务器地址,跟 startspeed 保持一致
//参数： vport 游戏服务器端口
```

强行关闭加速功能，对局过程中的网络探测功能。

正常结束加速器引擎

```
public static void EndSpeed(string vip, int vport)
//参数： vip 游戏服务器地址,跟 startspeed 保持一致
//参数： vport 游戏服务器端口
```

正常结束加速器引擎（选接项）

```
public static void GSDKEndSpeed(string vip, int vport，string extrainfo)
//参数： vip游戏服务器地址,跟startspeed保持一致
//参数： vport 游戏服务器端口
//参数：extrainfo传入网络质量统计等任意业务希望上报的参数，做上报使用
```

查询 4G QoS 是否保障

```
public static int IsQosWork()
```

返回 QoS 是否有保障成功：需要把此状态信息在游戏体验数据中记录并上报。4G QoS 在移动 4G 网络下才会生效，一般在结束加速 EndSpeed 时调用，返回值具体含义如下，这个可以用于游戏上线后核对 4G QOS 加速效果。

返回值	具体含义
1	保障成功
0	保障失败（默认值）
-1	非 4G
-2	不满足保障条件
-3	云控配置关闭
-4	云控返回 errno 非 0
-5	解析云控数据失败
-6	获取本地 IP 失败
-7	运营商返回 errno 非 0
-8	解析运营商数据失败

网络诊断

设置实时网络诊断

1. 需要在初始化后设置 GSDKKartinObserver

```
public static void SetKartinObserver(GSDKKartinObserver d)
```

参数 GSDKKartinObserver 类型如 附录1。

2. 调用实时网络诊断

```
public static void QueryKartin(string tag)
//参数：tag，作为标识每一次查询的 ID。
```

网络诊断是一个耗时的过程，大概有4 - 8秒钟，因此 GSDK 使用异步返回来实现。调用此函数后，查询结果会通过回调 KartinObserver 返回。

数据上报

1. 为了更好的衡量网络加速效果，需要从游戏客户端，把游戏过程的网络质量数据来进行统计后上报到灯塔，智营网优服务端会进行分析，上报的内容参考以下表格。
2. 客户端上报需要使用灯塔 SDK，已接入 msdk 的业务可以直接使用 msdk 的灯塔接口 上报。

附录

附录1

参数 GSDKKartinObserver 类型定义：

```
public delegate void GSDKKartinObserver(KartinRet ret);
public class KartinRet {
public string tag; // 游戏传入的Tag
public int flag; // 查询成功标识，若为0则成功
public string desc; // 查询flag的具体描述
// 当时网络类型0: 无网络,1: 2G, 2: 3G, 3: 4G, 4: wifi
public int jump_network;
public int jump_signal; // 信号强度
// 0表示绿色，信号强；1表示黄色，信号弱；2表示红色，信号极弱
public int signal_status = -1;
public String signal_desc = ""; // 3、信号强度描述
public int jump_router; // ping路由时延
// ping状态,0表示绿色，时延低；2表示红色，时延高
public int router_status = -1;
public String router_desc = ""; // ping描述
public int jump_export = -1; // 宽带或基站出口时延
// export状态,0表示绿色，时延低；2表示红色，时延高
```

```
public int export_status = -1; // 宽带出口和基站出口状态
public String export_desc = ""; //宽带出口和基站出口描述
public int jump_proxy; // ping代理时延
public int jump_edge; // ping边缘时延
public int jump_terminal; // wifi终端数
// terminal状态,0表示绿色终端数少；2表示红色，时延高
public int terminal_status = -1; //wifi终端数状态
public String terminal_desc = ""; //wifi终端数描述
public int jump_terminal; // wifi终端数
public int jump_direct; // 直连测速时延
// 直连状态,0表示绿色时延低；2表示红色，时延高
public int direct_status; // 直连测速时延状态
public int direct_desc; // 直连状态的描述
// 网卡状态, 0表示绿色网卡无问题；2表示红色网卡有问题
public int netinfo_status; // 网卡状态
public int netinfo_desc; // 网卡情况具体描述
}
```

附录2

KartinRet 关键参数取值及其说明如下：

- 字段：flag。
- 关键名称：查询结果标识。

取值	含义
0	查询成功
-1	表明无网络
-2	表明请求云控失败，重新调用一次
-3	表明初始化 SDK 失败
-4	当前网络类型发生了变化，请稍后再试
-5	2G 网络不适合游戏，无法检测

- 字段：jump_network。
- 关键名称：网络类型。

取值	含义

取值	含义
0	无网络或网络类型无法识别
1	2G
2	3G
3	4G
4	Wi-Fi

- 字段：jump_signal。
- 关键名称：信号强度（仅 Wi-Fi）。

取值	含义
-1	获取强度失败，请稍后再试
0..4	Wi-Fi 信号强度

- 字段：jump_router。
- 关键名称：路由器时延（仅 Wi-Fi）。

取值	含义
-1	不支持路由延迟查询
-2	获取路由器延迟失败，请稍后再试
0..1000	当前路由器的延迟值

- 字段：jump_terminal。
- 关键名称：共享 Wi-Fi 设备数（仅 Wi-Fi）。

取值	含义
-1	仅在WIFI模式下支持共享 Wi-Fi 设备查询
0..254	链接相同 Wi-Fi 的设备数

- 字段：jump_export。

- 关键名称：宽带出口时延。

取值	含义
-1	获取社区延迟失败，请稍后再试
-2	抱歉，当前所在区域网络不支持社区宽带延迟查询
0..500	带宽出口时延值

- 字段：jump_direct。
- 关键名称：直连时延。

取值	含义
-1	获取直连延迟失败，请稍后再试
0..800	网络时延值

- 字段：netinfo_desc。
- 关键名称：直连时延。

取值	含义
String	当前网卡有丢包或错包，不适合游戏。

具体设置请参考王者荣耀的示例：（ 红色字体为备注 ）

WIFI 直连环境下图示如下：

4G 直连环境下图示如下：

签名与鉴权

最近更新时间：2019-11-25 19:47:02

腾讯云 API 会对每个访问的请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证用户身份。签名信息由用户所执有的安全凭证生成，安全凭证包括密钥 ID（SecretId）和密钥 KEY（SecretKey）。

申请安全凭证

进入腾讯云控制台产品页面，新建加速服务后，在列表页面可以得到游戏 ID（GameId）、密钥 ID（SecretId）和密钥 KEY（SecretKey），每开通一项加速服务，会产生一个“游戏 ID”，和“游戏 KEY”，游戏 KEY 默认只有创建者、管理员（全局协作者）才有查看游戏 KEY 的权限，如果您看不到相关的信息，请联系管理员查看。

生成签名字符串

有了安全凭证 GameId，SecretId 和 SecretKey 后，就可以生成签名串了。下面给出了一个生成签名串的详细过程：

假设用户的 GameId，SecretId 和 SecretKey 分别是：

GameId: 1794235
SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA
SecretKey: Gu5t9xGARNpq86cd98joQYCN3Cozk1qA

注意：
这里只是示例，请用户根据自己实际的 SecretId 和 SecretKey 进行后续操作。

以开通 4G 加速服务请求为例，当用户调用这一接口时，其请求参数如下:

参数名称	中文	参数值
Action	方法名	open
GameId	游戏 ID	1794235
SecretId	密钥 ID	AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA

参数名称	中文	参数值
PhoneNO	用户手机标识（按运营商接口返回的字符串）	13788282828
Timestamp	当前时间戳	1496203804
Nonce	随机正整数	1038417
DeviceCode	手机设备号（可不填）	xxx-yyy
VersionId	游戏版本信息（可不填）	1794235

对参数排序

首先对所有请求参数按参数名做字典序升序排列，所谓字典序升序排列，直观上就如同在字典中排列单词一样排序，按照字母表或数字表里递增顺序的排列次序，即先考虑第一个“字母”，在相同的情况下考虑第二个“字母”，依此类推。您可以借助编程语言中的相关排序函数来实现这一功能，如 PHP 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
"Action": "open",
"DeviceCode": "xxx-yyy",
"GameId": "2001902634",
"Nonce": 1038417,
"PhoneNO": 13788282828,
"SecretId": "AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA",
"Timestamp": 1465185768,
"VersionId": "1.2.3.0"
}
```

使用其它程序设计语言开发时, 可对上面示例中的参数进行排序，得到的结果一致即可。

拼接请求字符串

此步骤生成请求字符串。
将把上一步排序好的请求参数格式化成“参数名称”=“参数值”的形式，如对 Action 参数，其参数名称为"Action"，参数值为"open"，因此格式化后就为 Action=open 。

> 注意：
> “参数值”为原始值而非 URL 编码后的值。

将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=open&DeviceCode=xxx-yyy&GameId=1794235&Nonce=1038417&PhoneNO=13788282828&ProjectId=1006972&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA&Timestamp=1496203804&VersionId=1794235
```

拼接签名原文字符串

此步骤生成签名原文字符串。
签名原文字符串由以下几个参数构成:

- 请求方法: 支持 POST 和 GET 方式, 这里使用 GET 请求, 注意方法为全大写。
- 请求主机: `qos.api.cloud.tencent.com` 。
- 请求路径: 请求路径固定为 /qos。
- 请求字符串: 即生成的请求字符串。

签名原文串的拼接规则为:请求方法 + 请求主机 + 请求路径 + ? + 请求字符串

示例的拼接结果为：

```
GETqos.api.qcloud.com/qos?Action=open&DeviceCode=xxx-yyy&GameId=1794235&Nonce=1038417&PhoneNO=13788282828&ProjectId=1006972&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA&Timestamp=1496203804&VersionId=1794235
```

生成签名串

首先使用签名算法（HmacSHA256）对上一步中获得的签名原文字符串进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例，由于本例中所用的签名算法为 HmacSHA256，因此生成签名串的代码如下：

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3Cozk1qA'; //密钥KEY
$srcStr =
  'GETqos.qcloud.com/qos?Action=open&DeviceCode=xxx-yyy&GameId=1794235&Nonce=1038417&PhoneNO=13788282828&ProjectId=1006972&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA&Timestamp=1496203804&VersionId=1794235 ';
$signStr = base64_encode(hash_hmac('sha256', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为:

```
ORFGm9wSTiI++b/NAIG63NRuEhA0x1AjXvrg72yls5Y=
```

使用其它程序设计语言开发时, 可用上面示例中的原文进行签名验证, 得到的签名串与例子中的一致即可。

签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。
如上一步生成的签名串为 `ORFGm9wSTiI++b/NAIG63NRuEhA0x1AjXvrg72yls5Y=` ，则其编码后为 `ORFGm9wSTiI++b/NAIG63NRuEhA0x1AjXvrg72yls5Y=` 。因此，最终得到的签名串请求参数(Signature)为： `ORFGm9wSTiI++b/NAIG63NRuEhA0x1AjXvrg72yls5Y=` ，它将用于生成最终的请求 URL。

> 注意：
> 如果用户的请求方法是 GET，则对所有请求参数值均需要做 URL 编码；部分语言库会自动对 URL 进行编码，重复编码会导致签名校验失败。

鉴权失败

当鉴权不通过时，可能出现的错误如下表：

错误代码	错误类型	错误描述
1	鉴权失败	签名验证失败，请确保您请求参数中的 Signature 计算正确；也可能是密钥状态有误，请确保 API 密钥有效且未被禁用。
2	查询IP数据库失败	请确认 IP 是否正确
3	运营商不支持加速	用户所处在地区的运营商加速服务未开通
4	请求 URL 异常	请求 URL 异常
5	不满足开启条件	不满足开启条件
6	系统错误	系统出现异常错误
7	获取手机号 URL 失败	获取手机号 URL 失败
8	开启失败	开启失败
9	关闭失败	关闭失败