Kiwi sdk接入文档

Android接入

添加SDK至工程

Kiwi提供的Andoid sdk为Kiwi.aar，包含armeabi-v7a arm64-v8a x86三种arch

以Android Studio为例，推荐将Kiwi.aar复制到libs目录，参考这里

适当添加以下内容至工程的build.gradle中

```
apply plugin: 'com.android.application'

android {
    ...
}

repositories{
    flatDir {
        dirs 'libs'
    }
}

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

引入包

```
import com.kiwi.sdk.Kiwi;
```

调用初始化接口

```
/**
 * @breif 接口会访问网络，不要放在UI线程中
 * @param appkey 控制台获取的appkey
 * @return 0表示成功，非0表示失败，请咨询Kiwi开发人员
 */
public static int Init(String appkey);
```

调用代码示例：

```
// 请替换真实appkey
String appkey = "P8+LRKkfH6m59+x/WBk+8l74OmBMSRdK7rYg+FaS/74=";
int ret = Kiwi.Init(appkey);
Log.d("kiwidemo", "Kiwi.Init return " + appkey);
```

调用转化接口

```
/**
 * @breif 转化接口，将rs标识转换为本地访问。不会访问网络，不会卡顿
 * @param name 控制台配置的防护目标rs标识
 * @param ip 转换后的ip
 * @param port 转换后的端口
 * @return 0表示成功，非0表示失败，请咨询Kiwi开发人员
 */
public static int ServerToLocal(String name, StringBuffer ip, StringBuffer port);
```

调用代码示例：

```
final StringBuffer ip = new StringBuffer();
final StringBuffer port = new StringBuffer();

// 请替换真实rs标识
String name = "web";
final int ret = Kiwi.ServerToLocal(name, ip, port);
if (ret < 0) {
    Log.d("kiwidemo", "call ServerToLocal failed " + ret);
    return;
}

String url = "http://" + ip.toString() + ":" + port.toString() + "/index.html";
// 访问url ...
```

代码混淆

注意：如果使用 proguard 进行混淆代码，需要在 proguard-rules.pro 文件中新增一行 -keep class com.kiwi.sdk.**{*;}

iOS接入

添加SDK至工程

Kiwi提供的iOS sdk为Kiwi.framework，包含arm64, x86_64两种arch。如果需要模拟器SDK，请单独联系我们。

- 选中项目名称
- 选中TARGETS
- 切换到General
- 下拉找到"Frameworks, Libraries, and Embeded Content"
- 点击“+”号，点击And Other... -> And File...，选择Kiwi.framework文件

引入头文件

```
#import <Kiwi/Kiwi.h>
```

调用初始化接口

```
/**
 * @breif 初始化接口，不需要重复调用接口。会访问网络，不要放在UI线程中
 * @param appkey 控制台获取的appkey
 * @return 0表示成功，非0表示失败，请咨询Kiwi开发人员
 */
+(int) Init:(const char *)appkey;
```

调用代码示例：

```
// 请替换真实appkey
const char *appkey = "P8+LRKkfH6m59+x/WBk+8l74OmBMSRdK7rYg+FaS/74=";
int ret = [Kiwi Init:appkey];
NSLog(@"Kiwi Init return %d", ret);
```

调用转化接口

```
/**
 * @breif 转化接口，将rs标识转换为本地访问。不会访问网络，不会卡顿
 * @param name 控制台配置的防护目标rs标识
 * @param ip 转换后的ip缓冲区指针
 * @param ip_len 转换后的ip缓冲区长度
 * @param port 转换后的端口缓冲区指针
 * @param port_len 转换后的端口缓冲区长度
 * @return 0表示成功，非0表示失败，请咨询Kiwi开发人员
 */
+(int) ServerToLocal:(const char*)name :(char*)ip :(int)ip_len :(char*)port :(int)port_len;
```

调用代码示例：

```
 char ip[40] = {0};
char port[40] = {0};

// 请替换真实rs标识
const char *name = "echo";
int ret = [Kiwi ServerToLocal:name :ip :sizeof(ip) :port :sizeof(port)];
NSLog(@"L4 ServerToLocal return %d", ret);
if (ret != 0) {
    return;
}

NSString* url = [NSString stringWithFormat:@"http://%s:%s/index.html", ip, port];
// 访问url ...
```

Windows接入

添加到Windows工程

Kiwi提供的SDK为动态库，包含Kiwi.dll、Kiwi.h、Kiwi.lib三部分

- 将Kiwi.h放入工程的头文件目录中
- 打开工程->属性->链接器->输入->附加依赖项，添加Kiwi.lib，注意Debug和Release配置都需要设置
- 编译完成后，将Kiwi.dll与工程生成的exe放到同一目录运行

调用代码示例

```
 #include <stdio.h>

#include "Kiwi.h"

int main() {
    char appkey[] = "Your appkey";
    char target[] = "Your target";

    char ip[64] = {0};
    char port[16] = {0};

    int ret = KiwiInit(appkey);
    if (ret != 0) {
        printf("KiwiInit failed %d\n", ret);
        return ret;
    }

    ret = KiwiServerToLocal(target, ip, sizeof(ip), port, sizeof(port));
    if (ret != 0) {
        printf("KiwiServerToLocal failed %d\n", ret);
        return ret;
    }

    printf("KiwiServerToLocal success %s %s\n", ip, port);
    getchar();

    return 0;
}
```

附录

Kiwi接口错误码

```
enum KiwiStatus
{
    KiwiOK = 0,
    KiwiParam = -1,
    kiwiAppkey = -2,
    KiwiApiRequestFailed = -3,
    KiwiNetwork = -4,
    KiwiNoSuchRsName = -5,
    KiwiConnect = -6,
    KiwiNetworkRead = -7,
    KiwiNetworkWrite = -8,
    KiwiJson = -9,
    KiwiJsonNoRoot = -10,
    KiwiJsonNoMember = -11,
    KiwiJavaRuntime = -12,
    KiwiCherryRpcFailed = -13,
    KiwiNoRss = -14,
    KiwiNoApp = -15,
    KiwiUnknow = -999
};
```