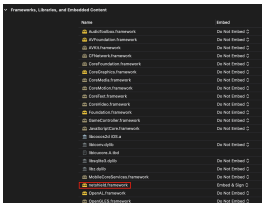
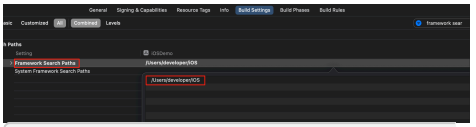
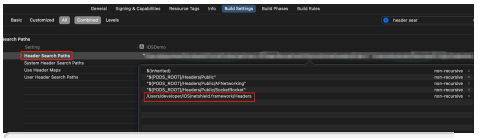
接入文档 - 正式版本v2

SDK接入指南

一、原生平台

1.1 iOS

1.1.1 工程配置

①添加framework文件	②设置framework搜索路径	③设置header搜索路径
	 即存放netshield.framework的本地目录	 即netshield.framework的全路径追加"/Headers"

1.1.2 接口说明

头文件：`LibNetshield.h`

方法	说明
`+ (void)startService:key:completionHandler:`	启动防护服务 `pid`-项目ID `key`-授权密钥 `completionHandler`-启动回调函数，详见示例代码
`+ (int)getLocalhostPort:`	根据规则名获取本地端口
`+ (void)stopService`	停止防护服务，释放资源

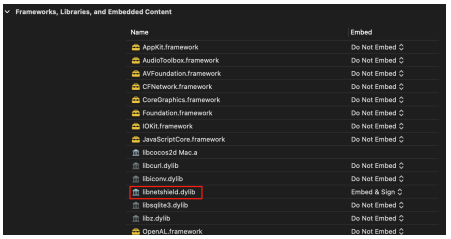
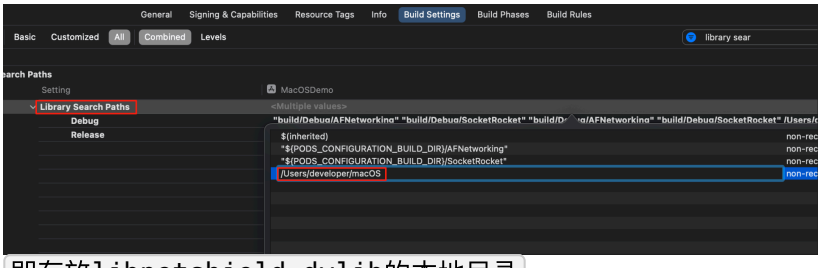
1.1.3 示例代码

```
// 引入头文件
#import <LibNetshield.h>

// 启动防护服务
[LibNetshield startService:@"123"
                       key:@"12345678901234567890"
         completionHandler:^(BOOL success, NSString *errMessage, NSString
*clientIP) {
```

```
                // success      SDK是否成功启动 YES-启动成功 NO-启动失败
                // errMessage   启动失败时用于获取失败原因，启动成功时可忽略
                // clientIP     启动成功时为客户端真实IP，启动失败为空
                if (success) {
                    NSLog(@"SDK start success,clientIP:%@", clientIP);
                } else {
                    NSLog(@"SDK start failed, errMessage:%@", errMessage);
                }
            }];

// 根据规则名获取本地端口
int localPort = [LibNetshield getLocalhostPort:@"test;10001"];
// 通过本地端口访问真实后端服务
NSString *url = [NSString stringWithFormat:@"http://127.0.0.1:%d",
localPort];

// 客户端业务流程
.........


// 应用退出前，停止防护服务，释放资源
[LibNetshield stopService];
```

1.2 macOS平台

1.2.1 工程配置

①添加dylib文件	②设置dylib链接搜索路径
	 即存放libnetshield.dylib的本地目录

1.2.2 接口说明

方法	说明
`void startNetshieldService(const char *pid, const char *key, void (*completionHandler)(int, const char *, const char *))`	启动防护服务 `pid`-项目ID `key`-授权密钥 `completionHandler`-启动回调函数，详见示例代码
`int getLocalhostPort(const char *)`	根据规则名获取本地端口
`void stopNetshieldService()`	停止防护服务，释放资源

1.2.3 示例代码

以下代码为c++和objc混编，需要将接入源码的 **.m** 后缀修改为 **.mm**

①添加extern函数声明

```
#ifdef __cplusplus
extern "C" {
#endif
extern void startNetshieldService(const char *, const char *, void (*)(int, const char *, const char *));
extern int getLocalhostPort(const char *);
extern void stopNetshieldService();
#ifdef __cplusplus
}
#endif
```

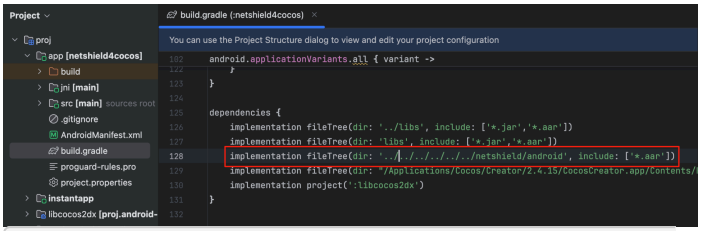
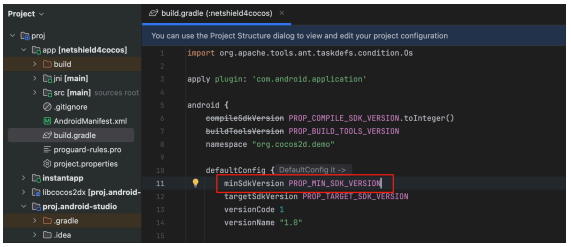
②调用接口

```
// 启动防护服务
NSString *pid = @"123";
NSString *key = @"12345678901234567890";
startNetshieldService([pid UTF8String], [key UTF8String], [](int startResult, const char *errMessage, const char *clientIP) {
    // startResult SDK是否成功启动 0-启动成功 -1-启动失败
    // errMessage  启动失败时用于获取失败原因，启动成功时可忽略
    // clientIP    启动成功时为客户端真实IP，启动失败为空
    if (0 == startResult) {
        NSLog(@"SDK started success,clientIP:%s", clientIP);
    } else {
        NSLog(@"SDK started failed, errMsg:%@", [[NSString alloc] initWithUTF8String:errMessage]);
    }
});
```

```
// 根据规则名获取本地端口
int localhostPort = getLocalhostPort("240.0.0.2;80");
// 通过本地端口访问真实后端服务
NSString *localhostUrl = [NSString stringWithFormat:@"http://127.0.0.1:%d",
localhostPort];

// 客户端业务流程
.........


// 应用退出前，停止防护服务，释放资源
stopNetshieldService();
```

1.3 Android平台

1.3.1 工程配置

①app/build.gradle添加aar路径	②调整minSdkVersion为21或以上
 dir中的目录即即存放netshield.aar的本地目录，可以为全路径	

1.3.2 接口说明

类文件：`com.netshield.Netshield`

方法	说明
`static int startService(String pid, String key)`	启动防护服务 `pid`-项目ID `key`-授权密钥
`static String getClientIP()`	获取客户端真实IP，需要在SDK启动成功后调用
`static String getErrMessage()`	获取SDK初始化失败的错误信息
`static int getLocalhostPort(String rule)`	根据规则名获取本地端口
`static void stopService()`	停止防护服务，释放资源

1.3.2 示例代码

```
// 启动防护服务
// 成功返回0，失败返回-1
int startResult = Netshield.startService("123", "12345678901234567890");
if (startResult == 0) {
    // 获取真实的客户端IP
    String clientIP = Netshield.getClientIP();
    Log.d("SDK", "SDK start success,clientIP:" + clientIP);
} else {
    String errMessage = Netshield.getErrMessage();
    Log.e("SDK", String.format("SDK start failed,errMessage:%s",
errMessage));
}

// 根据规则名获取本地端口
int localPort = Netshield.getLocalhostPort("test;10001");
// 通过本地端口访问真实后端服务
String url = String.format("http://127.0.0.1:%d", localPort);

// 客户端业务流程
.........


// 应用退出前，停止防护服务，释放资源
stopService();
```

1.4 Windows平台

1.4.1 工程配置

将netshield.dll放置到exe同级目录

1.4.2 接口说明

方法	说明
`int startNetshieldService(const char *pid, const char *key)`	启动防护服务 `pid`-项目ID `key`-授权密钥
`const char *getClientIP()`	获取客户端真实IP，需要在SDK启动成功后调用
`const char *getErrMessage()`	获取SDK初始化失败的错误信息
`int getLocalhostPort(const char *rule)`	根据规则名获取本地端口

方法	说明
`void stopNetshieldService()`	停止防护服务，释放资源

1.4.3 示例代码

① 动态加载DLL

```
HMODULE netshieldDll                                        =
LoadLibrary(L"netshield.dll");
auto (*startNetshieldService)(const char *, const char *) =
reinterpret_cast<int(__cdecl *)(const char *, const char *)>
(GetProcAddress(netshieldDll, "startNetshieldService"));
auto (*getClientIP)()                                       =
reinterpret_cast<const char *(__cdecl *)()>(GetProcAddress(netshieldDll,
"getClientIP"));
auto (*getErrMessage)()                                       =
reinterpret_cast<const char *(__cdecl *)()>(GetProcAddress(netshieldDll,
"getErrMessage"));
auto (*getLocalhostPort)(const char *)                      =
reinterpret_cast<int(__cdecl *)(const char *)>(GetProcAddress(netshieldDll,
"getLocalhostPort"));
auto (*stopNetshieldService)()                              =
reinterpret_cast<void(__cdecl *)()>(GetProcAddress(netshieldDll,
"stopNetshieldService"));
```

② 调用接口

```
// 启动防护服务
// 成功返回0，失败返回-1
int startResult = startNetshieldService("123", "12345678901234567890");
if (startResult == 0) {
    // 获取真实的客户端IP
    char message[512];
    const char* clientIP = getClientIP();
    snprintf(message, sizeof(message), "SDK start success,clientIP:%s",
clientIP);
    OutputDebugStringA(message);
} else {
    char errMessage[512];
    snprintf(errMessage, sizeof(errMessage), "SDK start
failed,errMessage:%s", getErrMessage());
    OutputDebugStringA(errMessage);
}
```

```
// 根据规则名获取本地端口
int localPort = getLocalhostPort("test;10001");
// 通过本地端口访问真实后端服务
char url[1024];
snprintf(url, sizeof(url), "http://127.0.0.1:%d", localPort);

// 客户端业务流程
.........


// 应用退出前，停止防护服务，释放资源
// 如果遇到进程无法退出，需要在main函数结束前调用该函数
stopNetshieldService();
```

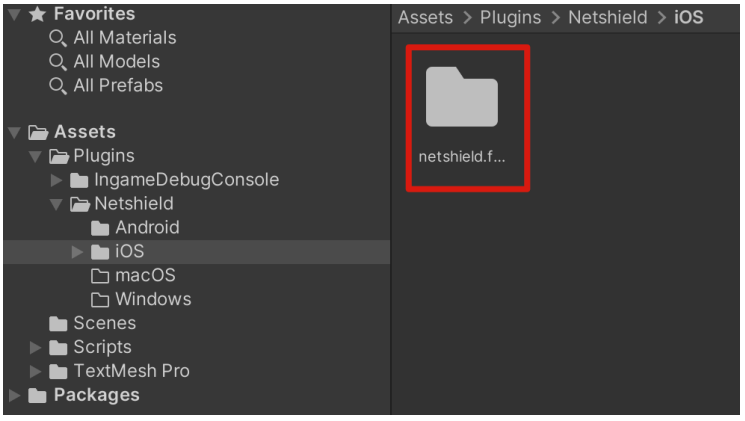
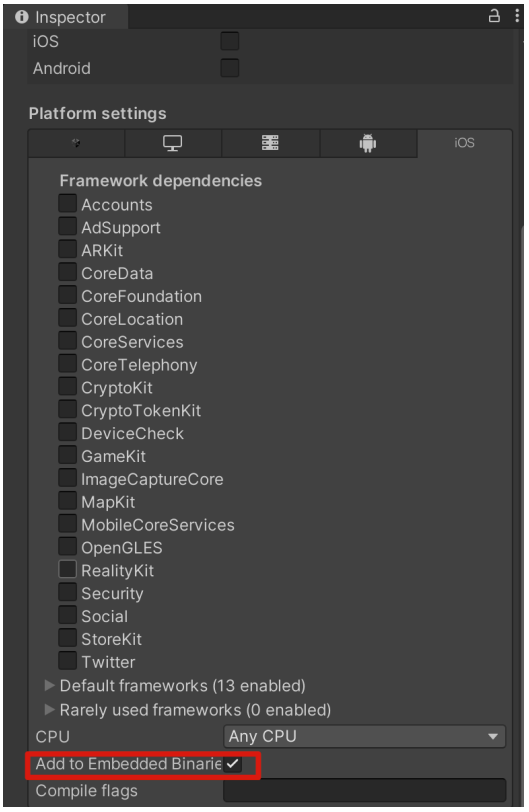
二、Unity平台

1、按下图目录结构拷贝SDK文件

```
├── Assets
│   ├── Plugins
│   │   ├── Netshield
│   │   │   ├── Android
│   │   │   │   ├── netshield-release.aar
│   │   │   ├── Windows
│   │   │   │   ├── netshield.dll
│   │   │   ├── iOS
│   │   │   │   ├── netshield.framework
│   │   │   ├── macOS
│   │   │   │   ├── libnetshield.dylib
│   ├── Scripts
│   │   ├── Netshield
│   │   │   ├── INetshieldInterface.cs
│   │   │   ├── Netshield.cs
│   │   │   ├── impl
│   │   │   │   ├── Netshield4Android.cs
│   │   │   │   ├── Netshield4OSX.cs
│   │   │   │   ├── Netshield4Others.cs
│   │   │   │   ├── Netshield4Windows.cs
```

Netshield.zip

2、设置framework为Embedded Binaries

①选中netshield.framework	②在Inspector中选中Add to Embedded Binaries
	

3、接口说明

方法	说明
`Netshield4Unity.GetInstance()`	获取Netshield4Unity单例对象
`void StartService(string pid, string key, CompletionHandler completionHandler)`	启动防护服务 `pid`-项目ID `key`-授权密钥 `completionHandler`-启动回调函数，详见示例代码
`void StopService()`	停止防护服务，释放资源
`int GetLocalhostPort(string rule)`	根据规则名获取本地端口

4、示例代码

```
// 启动防护服务
Netshield4Unity.GetInstance().StartService("123", "12345678901234567890",
(startResult, errMessage, clientIP) => {
    // startResult  SDK是否成功启动 0-启动成功 -1-启动失败
    // errMessage   启动失败时用于获取失败原因，启动成功时可忽略
    // clientIP     启动成功时为客户端真实IP，启动失败为空
    if (0 == startResult)
```

```
    {
        Debug.Log($"SDK start success,clientIP:{clientIP}");
    }
    else
    {
        Debug.LogError($"SDK start failed, errMessage:{errMessage}");
    }
});

// 根据规则名获取本地端口
int localPort =
Netshield.Netshield4Unity.GetInstance().GetLocalhostPort("test;10001");
// 通过本地端口访问真实后端服务
var url = "http://127.0.0.1:" + localPort;

// 客户端业务流程
.........


// 应用退出前，停止防护服务，释放资源
Netshield4Unity.GetInstance().StopService();
```

三、Cocos Creator

3.1 拷贝js接口文件到脚本目录

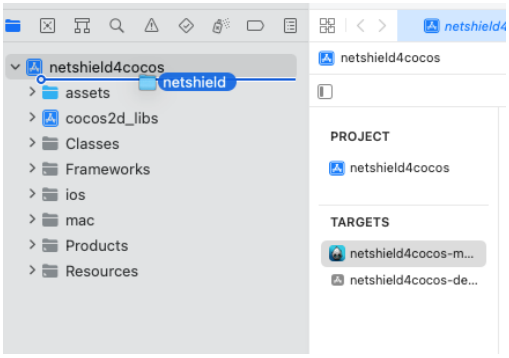
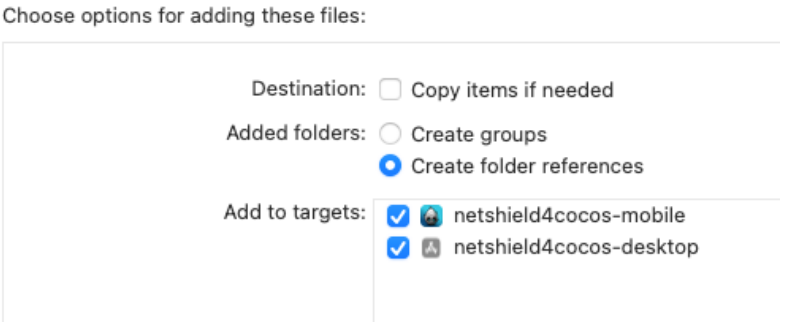
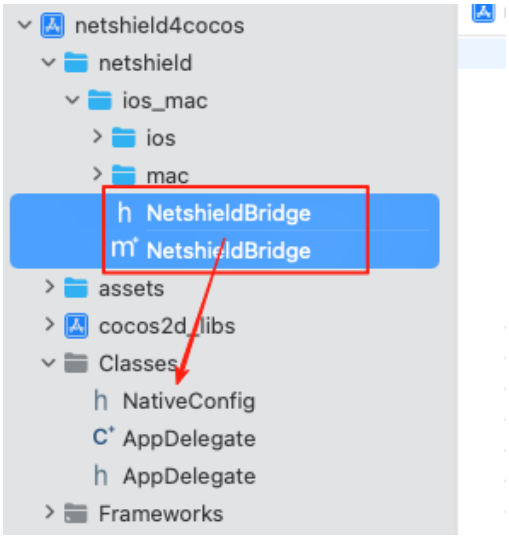
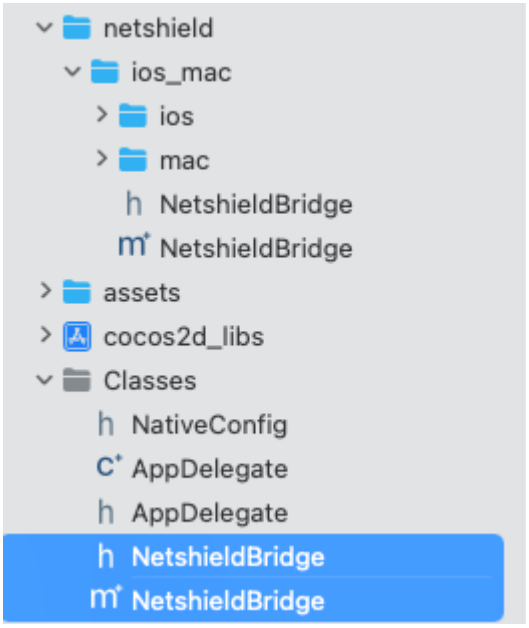
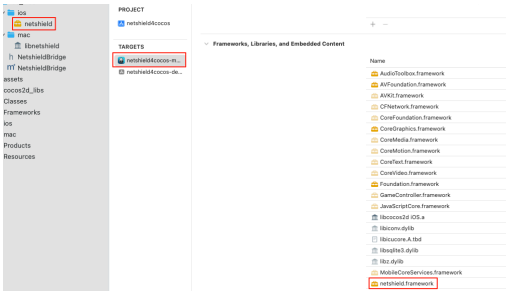
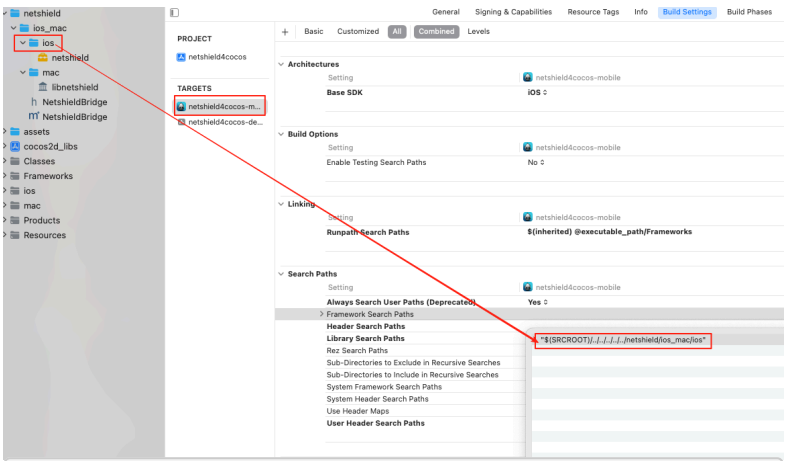
Netshield.js

3.2 拷贝库文件到各平台

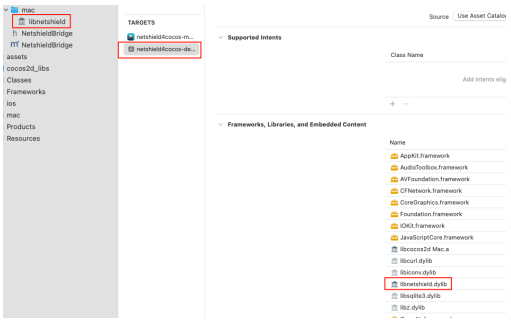
库结构

```
netshield
├── android
│   └── netshield-release.aar
├── ios_mac
│   ├── NetshieldBridge.h
│   ├── NetshieldBridge.mm
│   ├── ios
│   │   └── netshield.framework
│   └── mac
│       └── libnetshield.dylib
└── windows
    ├── NetshieldBridge.hpp
    └── netshield.dll
```

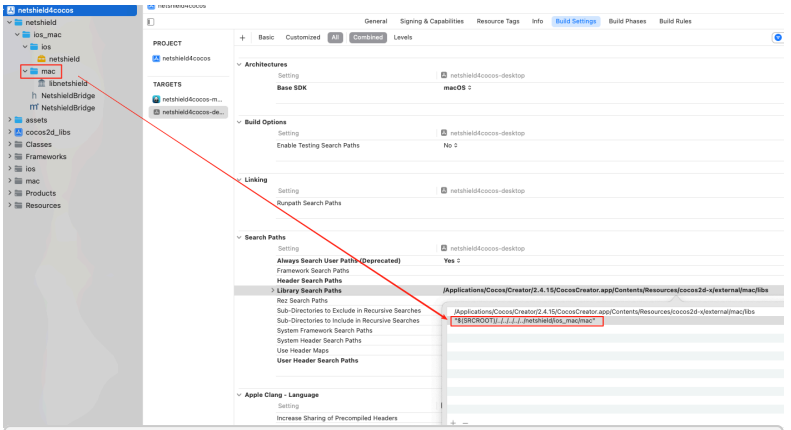
3.2.1 iOS和mac

①将SDK添加到工程中	②选择同时添加到iOS和macOS中
	
③将桥接文件添加到Classes目录	④检查结果
	
⑤iOS-添加framework文件	⑥iOS-设置framework搜索路径
	
⑦mac-添加dylib文件	⑧mac-设置dylib链接搜索路径

①将SDK添加到工程中

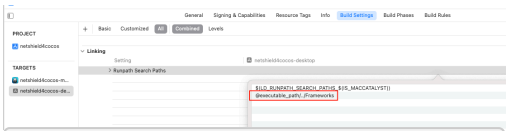


②选择同时添加到iOS和macOS中



```
${SRCROOT}/../../../../../netshield/ios_mac/mac
```

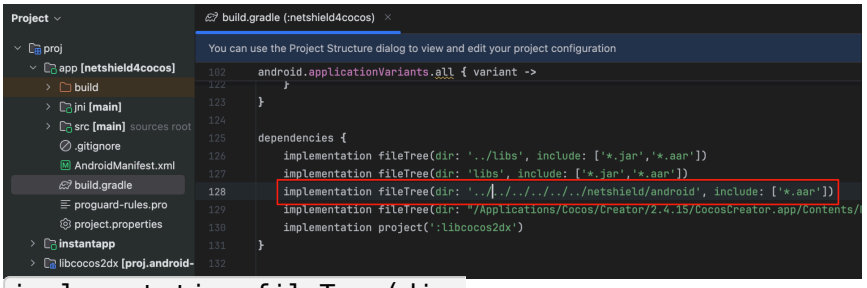
⑨mac-设置dylib运行搜索路径



```
@executable_path/../Frameworks
```

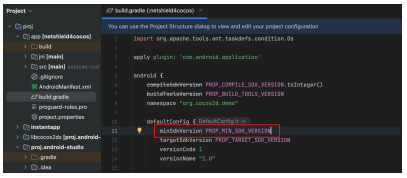
3.2.2 Android

①app/build.gradle添加aar路径



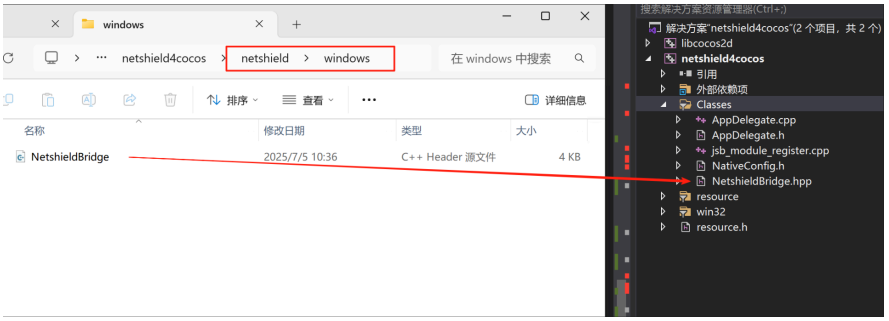
```
implementation fileTree(dir: '../../../../../../netshield/android', include: ['*.aar'])
```

②调整minSdkVersion为21或以上

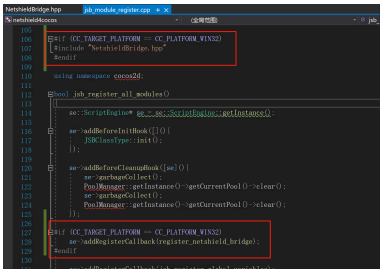


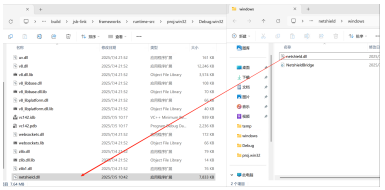
3.2.3 Windows

①将代码文件添加到工程中



②在jsb_module_register.cpp中注册



①将代码文件添加到工程中	②在jsb_module_register.cpp中注册
步骤②代码	③将dll文件添加到工程中
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WIN32) #include "NetshieldBridge.hpp" #endif // 以下代码需要添加到jsb_register_all_modules函数中 #if (CC_TARGET_PLATFORM == CC_PLATFORM_WIN32) se- >addRegisterCallback(register_netshield_bridge); #endif	

3.3 调用接口

```
async onStartSDK() {
    // 启动防护服务
    // Netshield.start返回一个对象
    // startResult.startResult  SDK是否成功启动 0-启动成功 -1-启动失败
    // startResult.errMessage   启动失败时用于获取失败原因，启动成功时可忽略
    // startResult.clientIP     启动成功时为客户端真实IP，启动失败为空
    const startResult = await Netshield.start("123",
"12312312312312312231");
    if (startResult.startResult === 0) {
        console.log("SDK start success,clientIP:" + startResult.clientIP);
    } else {
        this.appendLog("SDK start failed, errMessage:" +
startResult.errMessage);
    }
},

onRequestPage() {
    // 根据规则名获取本地端口
    const localhostPort = Netshield.getLocalhostPort("240.0.0.2;80");
    // 通过本地端口访问真实后端服务
    let url = "http://127.0.0.1:" + this.localhostPort;
    console.log("访问页面: " + url);
}

onStopSDK() {
    // 停止SDK
    Netshield.stop();
},
```

四、Flutter

SDK以flutter plugin的形式提供

4.1 添加依赖

通过本地路径的方式引入plugin

```
dependencies:
  netshield4flutter:
    path: ../path/to/your/netshield4flutter
```

4.2 接入代码（详细用法请参考plugin example）

```
import 'package:netshield4flutter/netshield4flutter.dart';

final _netshield4flutterPlugin = Netshield4flutter();
try {
  // 启动防护服务
  // Netshield.start返回一个对象
  // startResult.startResult  SDK是否成功启动 0-启动成功 -1-启动失败
  // startResult.errMessage   启动失败时用于获取失败原因，启动成功时可忽略
  // startResult.clientIP     启动成功时为客户端真实IP，启动失败为空
  final startResult = await _netshield4flutterPlugin.startService('123',
'123123123123123123123');
  if (startResult.isSuccess) {
    setState(() => sdkRunning = true);
    _addLog('SDK start success,clientIP:${startResult.clientIP}');
  } else {
    if (!mounted) return;
    _addLog("SDK start failed, errMessage:${startResult.errMessage}");
  }
} on PlatformException {
  _addLog("SDK startup exception");
}

int localhostPort = -1;
try {
    // 根据规则名获取本地端口
    localhostPort = await
_netshield4flutterPlugin.getLocalhostPort("240.0.0.2;80");
    // 通过本地端口访问真实后端服务
    String url = "http://127.0.0.1:$localhostPort";
} on PlatformException {
```

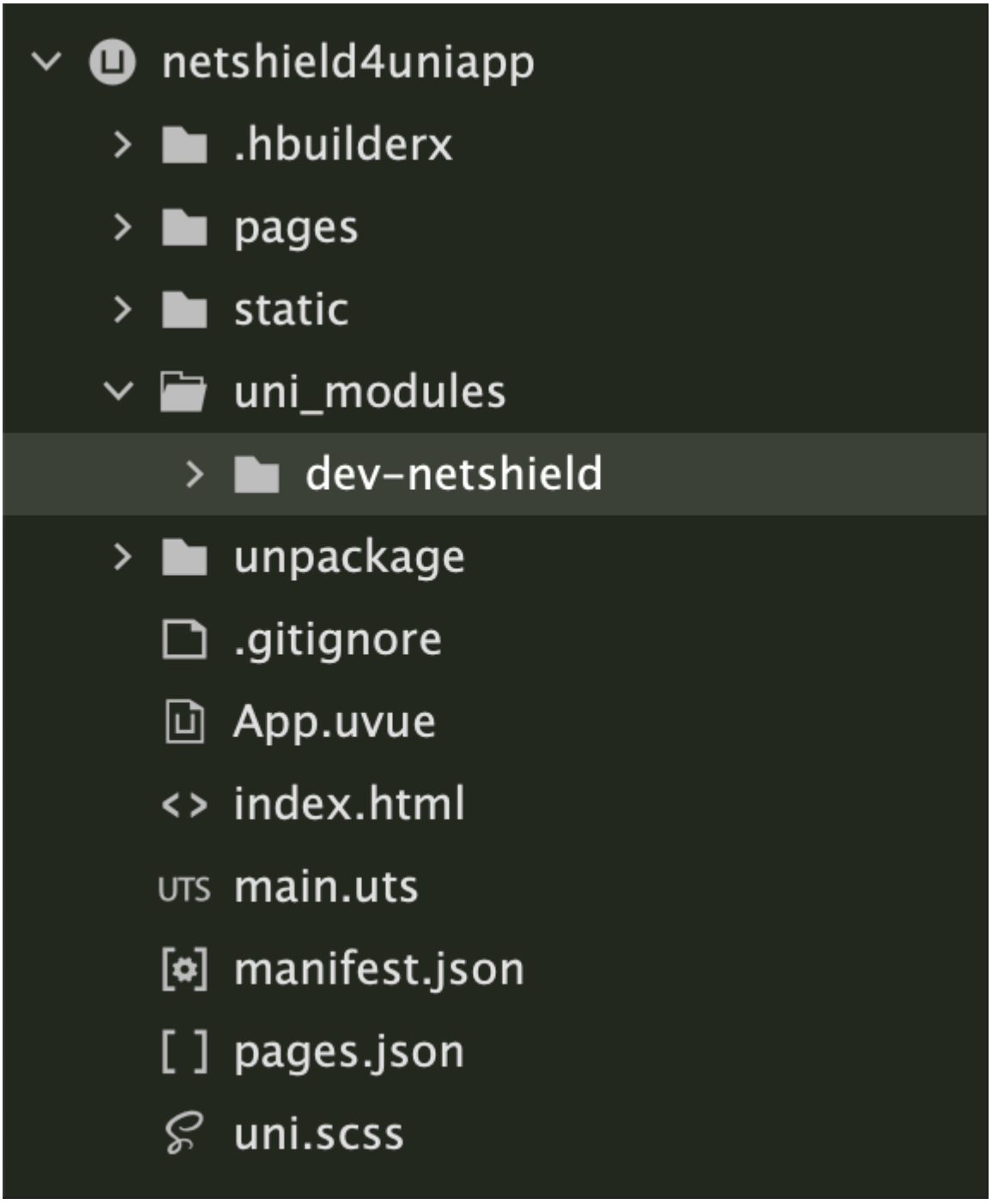
```
    print("无法获取本地端口");
}



//  （可选代码）应用退出前，停止防护服务，释放资源
try {
    // 关闭防护服务
    await _netshield4flutterPlugin.stopService();
} on PlatformException {
    print("SDK 停止失败");
}
```

五、UniApp

5.1 uts插件

5.1.1 放置插件

将uts插件放到uni_modules目录下



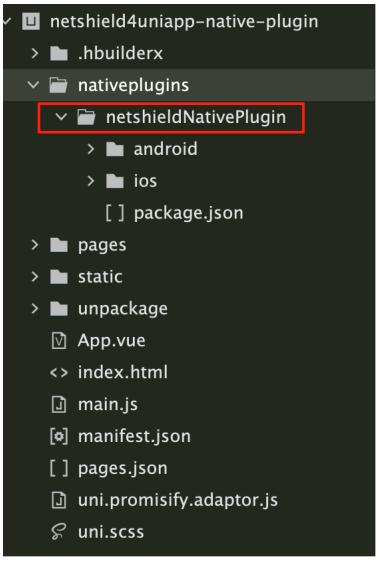
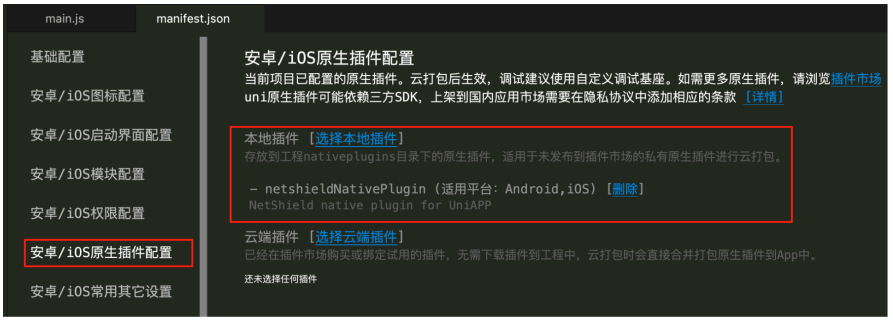
5.1.2接入代码

```
import { startNetshieldService, stopNetshieldService, getLocalhostPort }
from "@/uni_modules/dev-netshield";

export default {
    methods: {
        startSDK() {
            // 启动SDK
            // startResult[0] SDK是否成功启动 0-启动成功 -1-启动失败
            // startResult[1] 启动失败时用于获取失败原因，启动成功时可忽略
```

```
            // startResult[2] 启动成功时为客户端真实IP，启动失败为空
            startNetshieldService("123", "12312312312312312231",
(startResult : Array<string>) => {
                if ("0" == startResult[0]) {
                    this.addLog("SDK start success,clientIP:" +
startResult[2]);
                } else {
                    this.addLog("SDK start failed, errMessage:" +
startResult[1]);
                }
            });
        },
        browse() {
            // 根据规则名获取本地端口
            const localhostPort = getLocalhostPort("240.0.0.2;80");
            // 通过本地端口访问真实后端服务
            let url = "http://127.0.0.1:" + localhostPort;
            console.log("访问页面: " + url);
        },
        stopSDK() {
            // 关闭SDK
            stopNetshieldService();
        }
    }
}
```

5.2 原生插件

5.2.1 放置及启用插件

①将native插件放到nativeplugins目录下	②在manifest.json中选择本地插件
	

5.2.2 接入代码

```
const netshieldNativePlugin =
uni.requireNativePlugin('netshieldNativePlugin');

export default {
    methods: {
        startSDK() {
            // 启动SDK
            // startResult[0] SDK是否成功启动 0-启动成功 -1-启动失败
            // startResult[1] 启动失败时用于获取失败原因，启动成功时可忽略
            // startResult[2] 启动成功时为客户端真实IP，启动失败为空
            startNetshieldService("123", "123123123123123123231",
(startResult : Array<string>) => {
                if ("0" == startResult[0]) {
                    this.addLog("SDK start success,clientIP:" +
startResult[2]);
                } else {
                    this.addLog("SDK start failed, errMessage:" +
startResult[1]);
                }
            });
        },
        browse() {
            // 根据规则名获取本地端口
            const localhostPort =
netshieldNativePlugin.getLocalhostPort("240.0.0.2;80");
            // 通过本地端口访问真实后端服务
            let url = "http://127.0.0.1:" + localhostPort;
            console.log("访问页面: " + url);
        },
        stopSDK() {
            // 关闭SDK
            netshieldNativePlugin.stopNetshieldService();
        }
    }
}
```

六、注意事项

1. **初始化时机**：在应用启动后**尽早调用初始化方法**
2. **错误处理**：
    - iOS/macOS通过`completionHandler`回调判断结果

- 其他平台通过返回值判断（0=成功）

3. **资源释放**：退出时需调用`stopNetshieldService`方法释放资源
4. **动态库依赖**：Windows需正确部署`netshield.dll`，iOS和macOS平台的framework必须配置为Embed模式，否则会导致运行时崩溃

七、技术支持

商务合作与技术支持请联系telegram: **youxidun**
