UniConnect SDK 集成指导文档

iOS

深圳市联软科技股份有限公司

2022 年 10 月 31 日

目录

文档说明

文档名称	UniConnect SDK 集成指导文档			
内容描述	主要描述 UniConnect SDK 的集成步骤和说明			
修订历史				
日期	版本	修订者	修订说明	审核人
2022 年 10 月 31 日	V1.0	dengjian	新建文档	

一、 SDK 介绍

1. 兼容性

- 编译环境：Xcode11 及以上
- 运行环境：iOS12 及以上
- 支持架构：arm64（暂不支持模拟器）

2. 支持功能

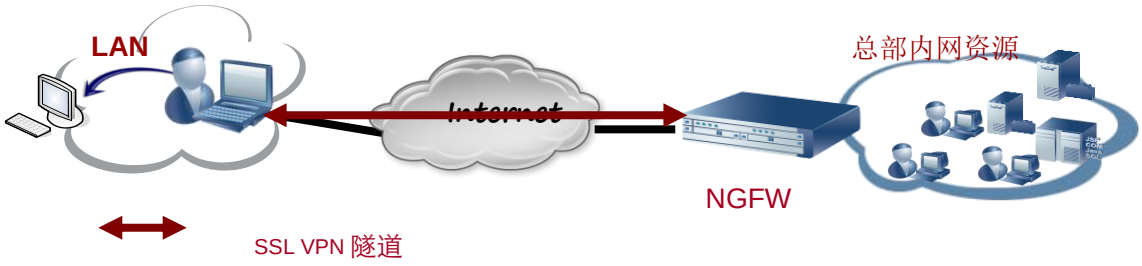
- 支持 SSL VPN 连接
- 支持网络扩展功能
- 支持用户名密码/证书匿名/证书挑战等认证方式

3. 网络扩展场景描述

移动端 SDK 目前仅支持网络扩展，此功能需要在客户端生成一个虚拟网卡，通过虚拟网卡与企业 内网进行通信。网络扩展功能将终端用户连接到企业网上，对终端用户而言，就像接入到企业内网的局域网一样，可以访问内网的任何资源。

网络扩展功能支持所有基于 IP 层之上的应用。因此被称为基于 IP 层的隧道技术，称作 L3VPN 功能。相对于其他业务功能而言，用户启用网络扩展功能后，如同接入到企业内网一样，网络扩展功能主要是为了提供用户远程接入能力，提供 IP 层的接入功能，实现对内网的全网访问。使用网络扩展功能之后，用户的 PC 就可以如同企业内网一样，可以快速、安全的访问企业内网的所有资源。

网络扩展的典型应用场景如下：



网络扩展是基于 SSL 协议进行的功能扩展。网络扩展是基于 IP 的内网业务的全面访问。

路由模式决定了客户端发送报文的路由。网络扩展功能支持二种路由模式：

全路由模式（Full Tunnel），手动模式（Manual Tunnel）。详细介绍如下：

• 全路由模式下，无论是访问什么资源，数据一概被虚拟网卡截获，转发给虚拟网关处理。

• 手动模式下，在服务端，管理员必须手动配置内网网段静态路由，然后在客户端识别前往该网段的数据，交由虚拟网卡转发。

4. SDK 说明

用户可通过集成 SDK 实现 SSL VPN 各项业务。此 SDK 包含两部分：

- UniConnectSDK.framework 用于主程序调用的 API
- UniConnectSDKTunnel.framework 用于插件集成

详细使用见以下操作说明。

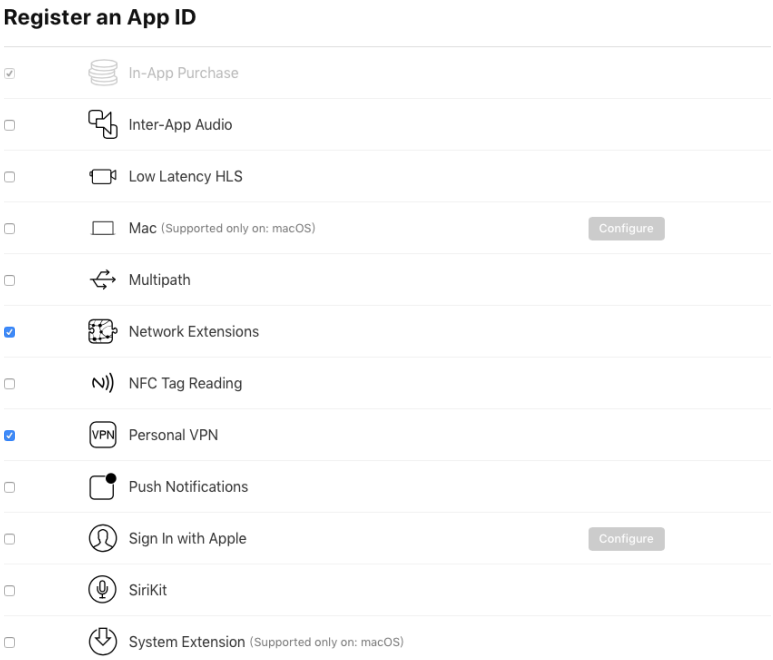
二、 开发前准备

1. 创建工程的 Provisioning Profile 文件

在苹果开发者账号网站创建扩展的 Provisioning Profile，在创建 appid 时需开启 Network

Extension、Personal VPN，如下图示。注意，app 及扩展都需要开启 Network Extension、Personal VPN，且扩展的 appid 以 APP 的 appid 为前缀。比如 APP 的 appid 为 com.testclient.client，那么扩展的 appid 可以设置为 com.testclient.client.vpnextension。

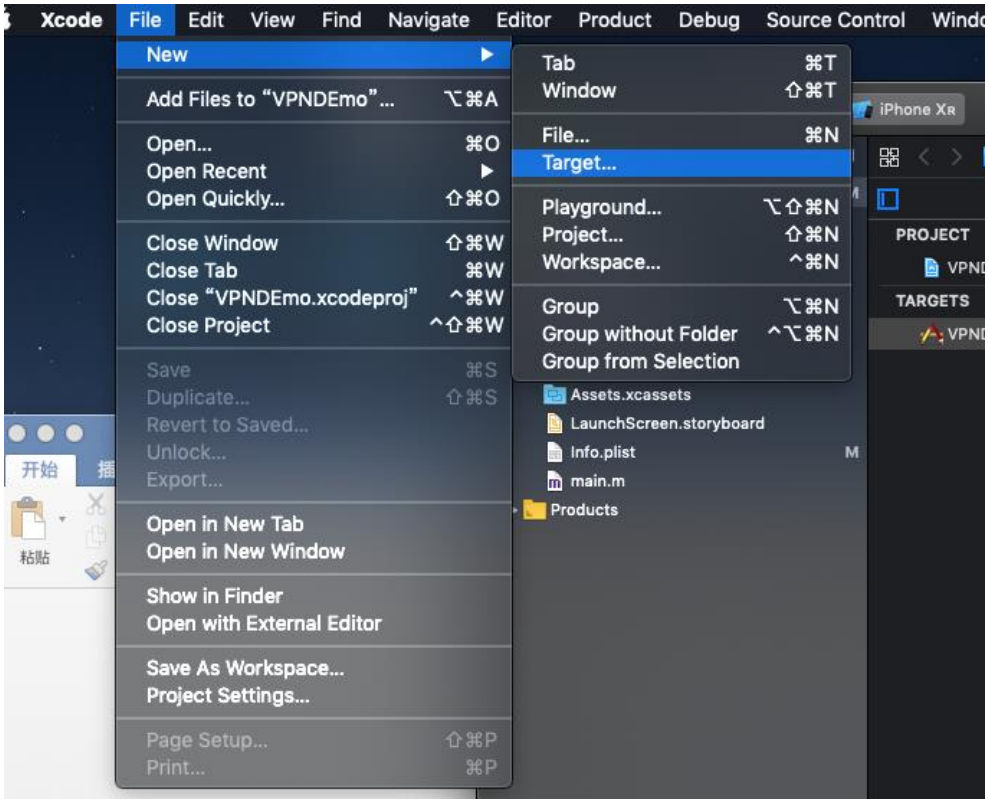
另外，app 和扩展需要配置成同一个 app group 中。

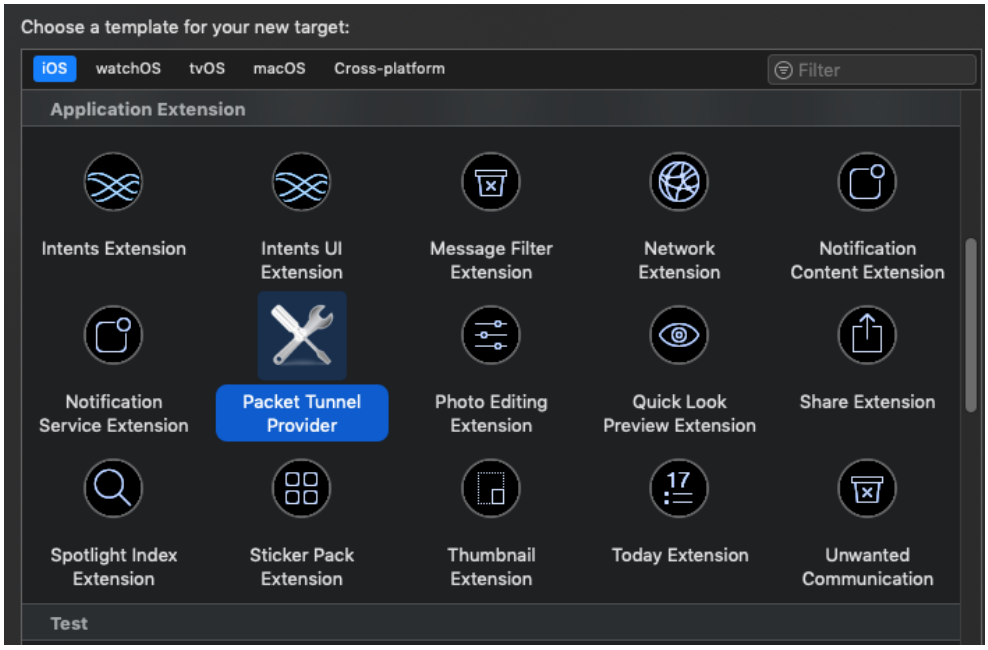


2. 配置工程开发环境

(1) 创建 target

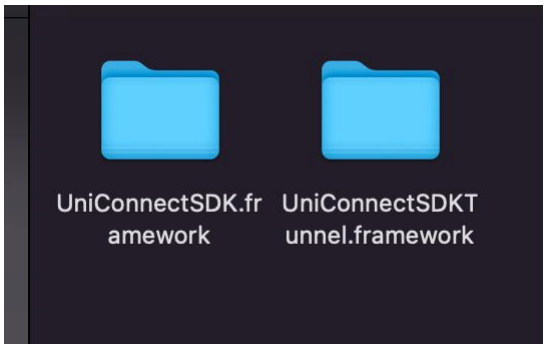
创建工程后新建一个 Target，选择 File->New->Target，选择 Packet Tunnel Provider

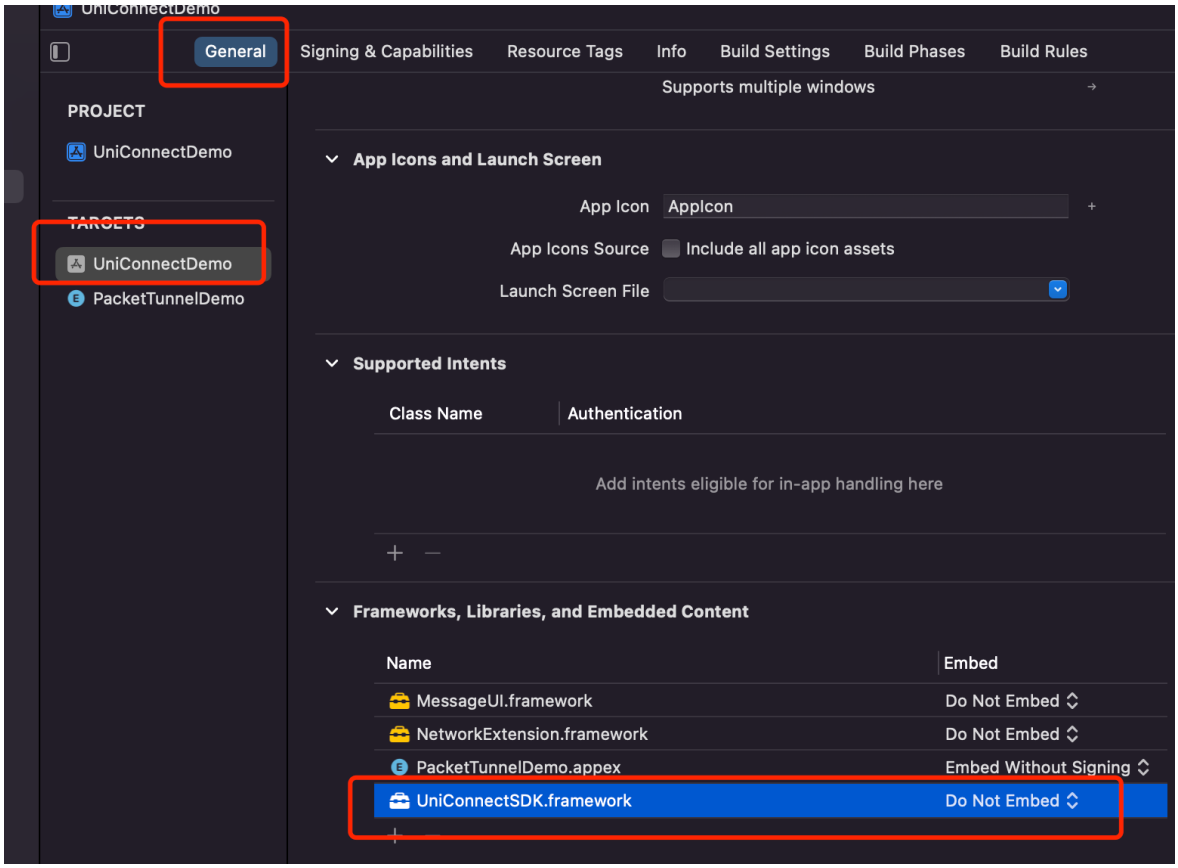


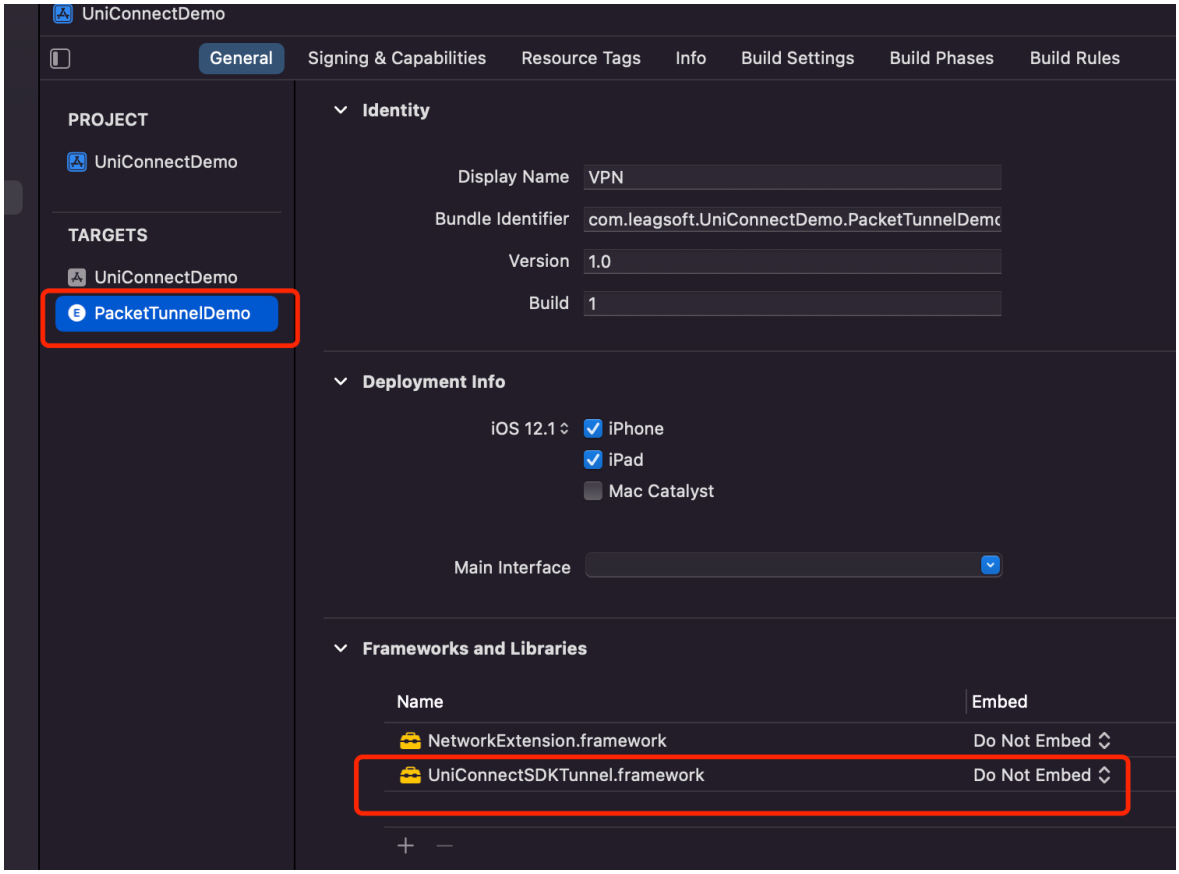


(2) 添加 sdk 到 target

将提供的静态库 UniConnectSDK.framework、UniConnectSDKTunnel.framework 分别引入到工程和新建的 Packet Tunnel Provider 中

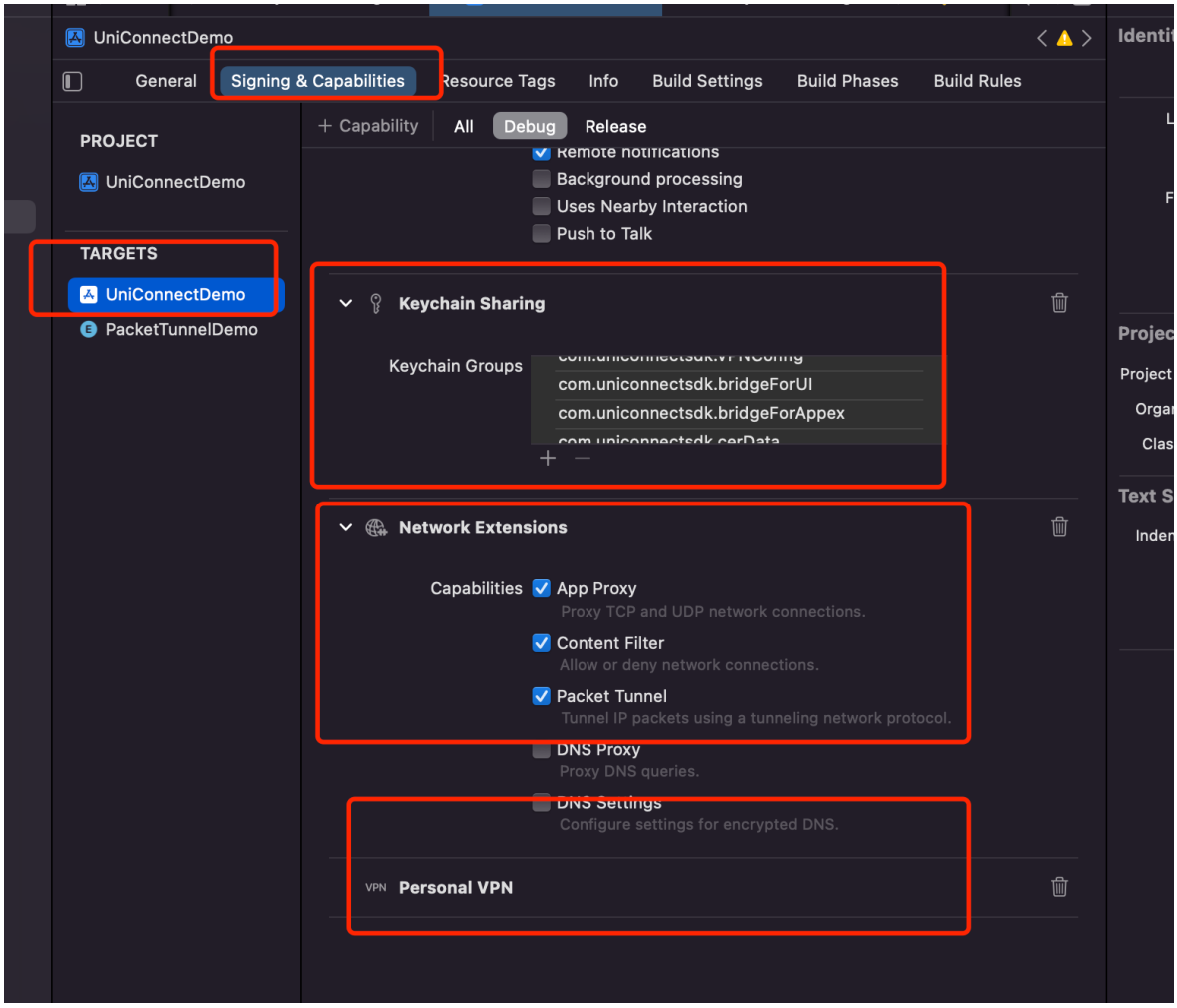

UniConnectSDK.framework
UniConnectSDKTunnel.framework


General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
UniConnectDemo
TARGETS
UniConnectDemo
PacketTunnelDemo
Supports multiple windows
App Icons and Launch Screen
App Icon
AppIcon
App Icons Source
Include all app icon assets
Launch Screen File
Supported Intents
Class Name
Authentication
Add intents eligible for in-app handling here
Frameworks, Libraries, and Embedded Content
Name
Embed
MessageUI.framework
Do Not Embed
NetworkExtension.framework
Do Not Embed
PacketTunnelDemo.appex
Embed Without Signing
UniConnectSDK.framework
Do Not Embed



(3) 权限配置

在两个 Target 中配置申请好的 appid，在 xcode 的 Capabilities 中开启相应权限。



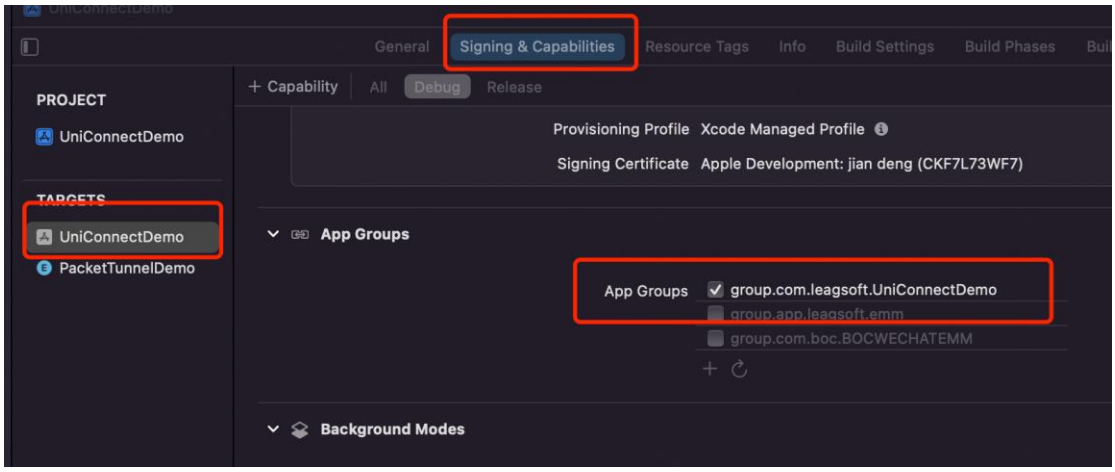
此处注意开启 Keychain Sharing 时需要添加以下 Keychain Groups：

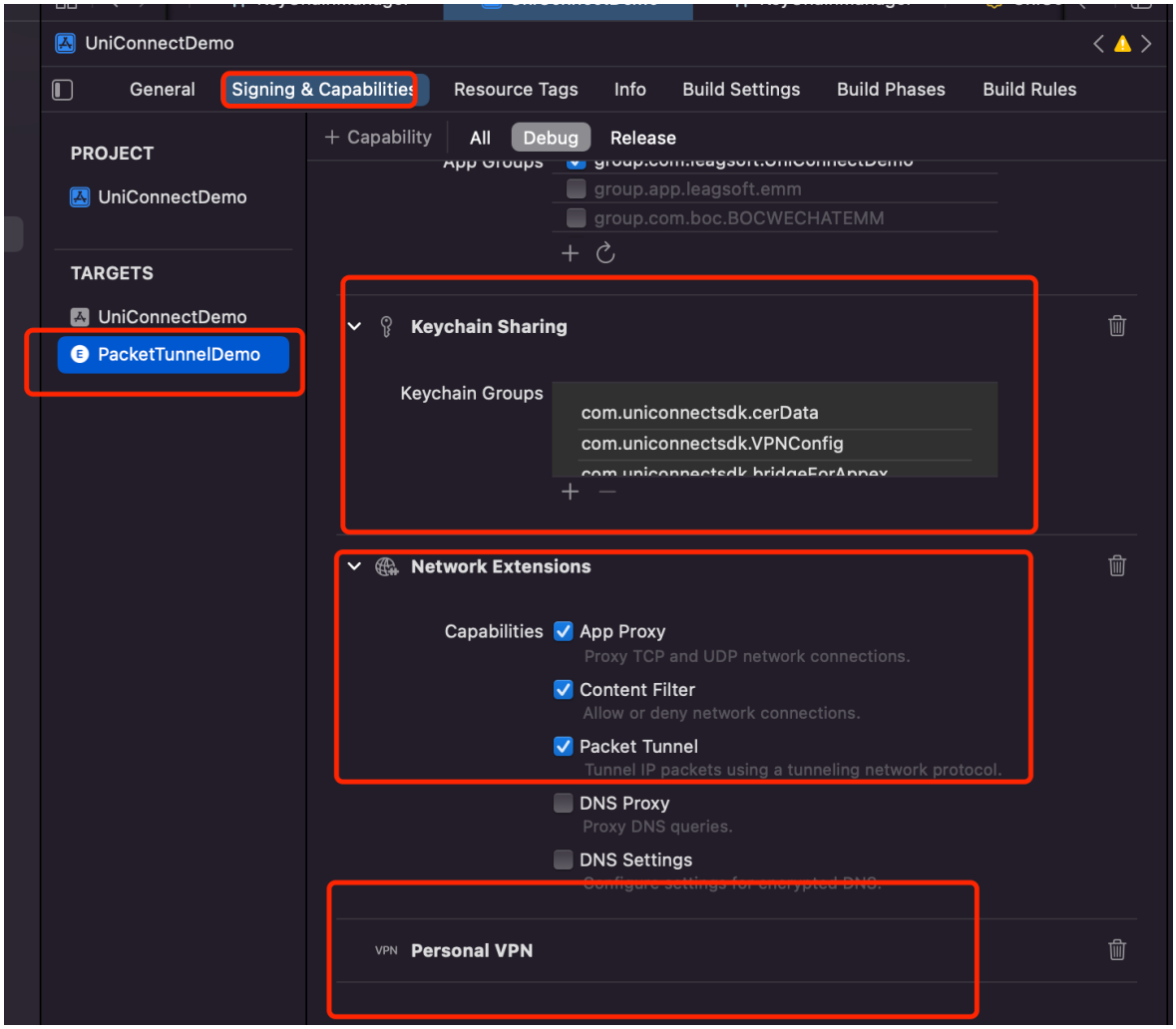
- com.uniconnectsdk.VPNConfig
- com.uniconnectsdk.bridgeForUI
- com.uniconnectsdk.bridgeForAppex
- com.uniconnectsdk.cerData

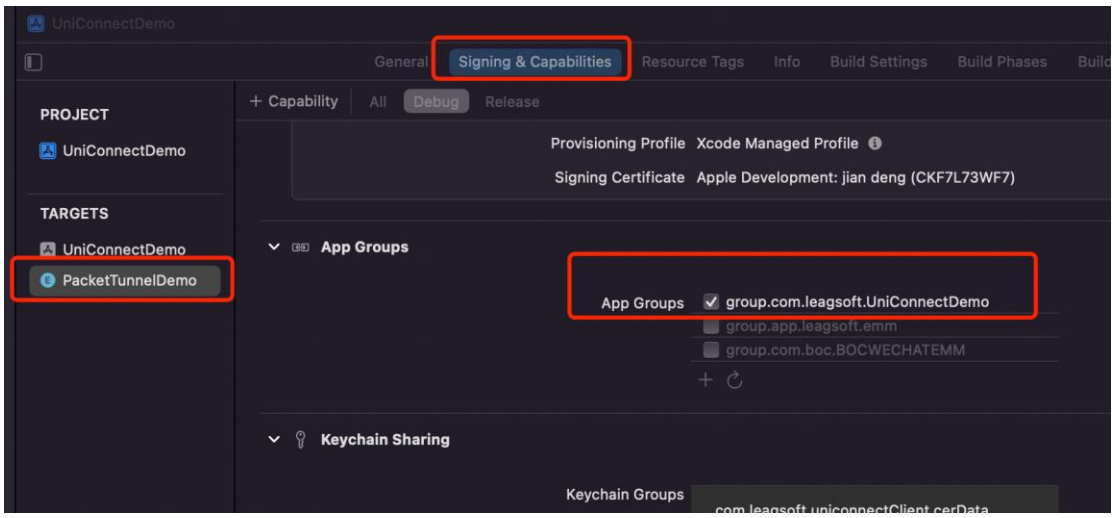
(4) 配置 app group id

capabilities 中配置勾上相应的 group id，并且在主应用的 info.plist 也配置上。

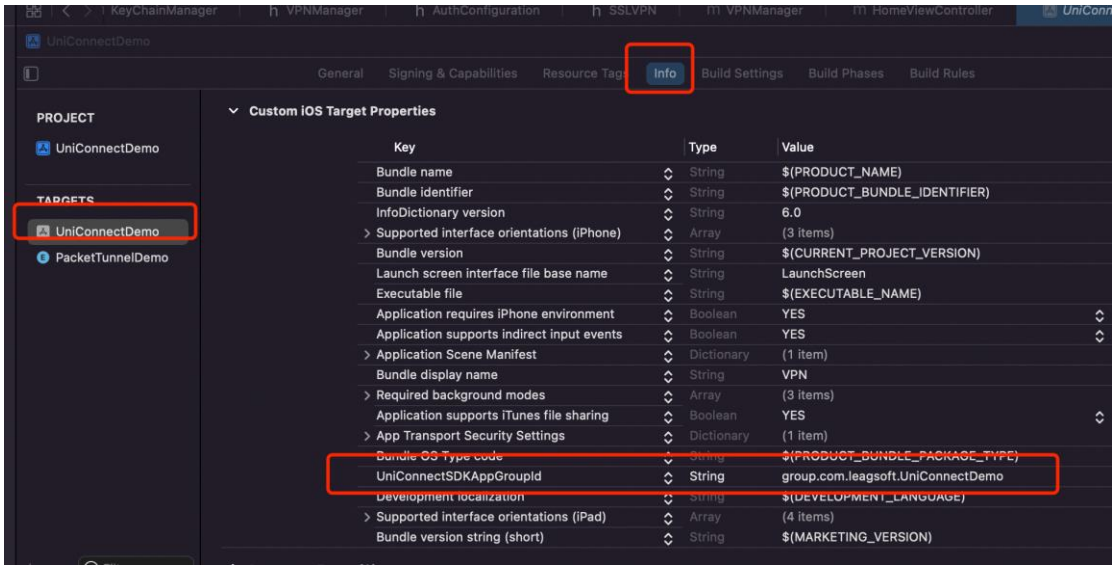
主 target 和扩展 target 都要配置，并且是同一个。


General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
PROJECT
UniConnectDemo
TARGETS
UniConnectDemo
PacketTunnelDemo
+ Capability
All
Debug
Release
Provisioning Profile Xcode Managed Profile
Signing Certificate Apple Development: jian deng (CKF7L73WF7)
App Groups
App Groups
group.com.leagsoft.UniConnectDemo
group.app.leagsoft.emm
group.com.boc.BOCWECHATEMM
Background Modes


UniConnectDemo
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
UniConnectDemo
TARGETS
UniConnectDemo
PacketTunnelDemo
+ Capability
All
Debug
Release
group.app.leagsoft.emm
group.com.boc.BOCWECHATEMM
Keychain Sharing
Keychain Groups
com.uniconnectsdk.cerData
com.uniconnectsdk.VPNConfig
Network Extensions
Capabilities
App Proxy
Proxy TCP and UDP network connections.
Content Filter
Allow or deny network connections.
Packet Tunnel
Tunnel IP packets using a tunneling network protocol.
DNS Proxy
Proxy DNS queries.
DNS Settings
VPN
Personal VPN

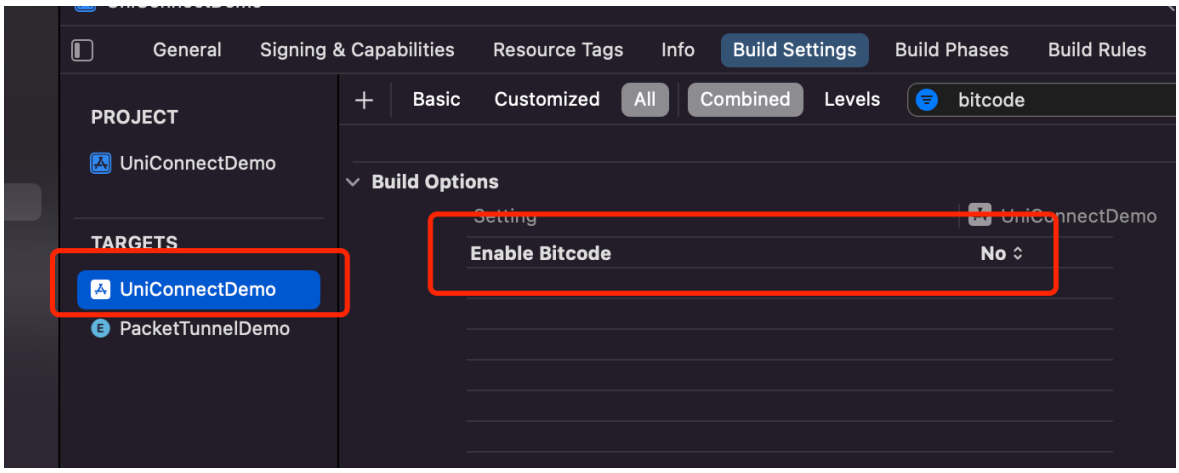


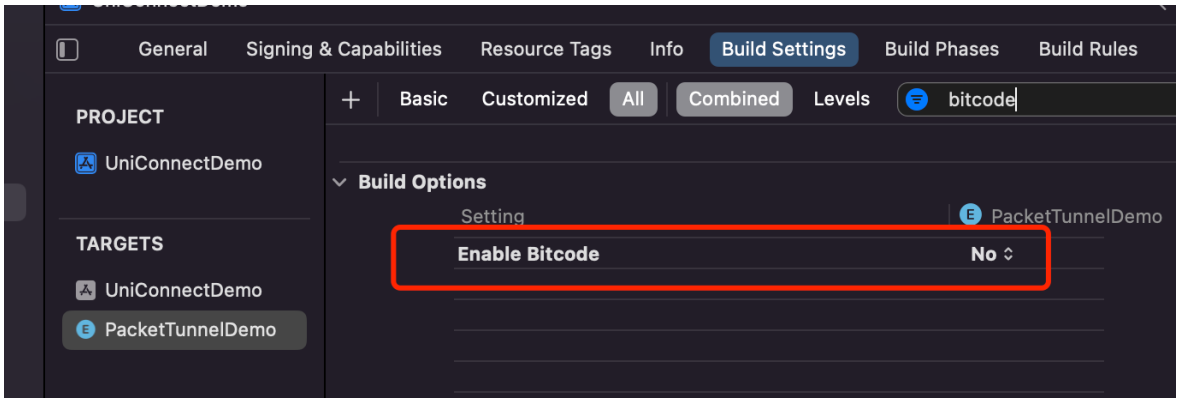
在主 target 的 info.plist 配置 app group id。key 为 UniConnectSDKAppGroupId。



(5) 关闭 Bitcode

在 Build Settings 里面设置关闭 Bitcode。





三、 SDK 使用说明

1. UniconnectSDKTunnel.framework 的使用

(1) 引入头问题

PacketTunnelProvider.h 文件中引入 “#import <UniConnectSDKTunnel/UniConnectSDKTunnel.h>”，并且使 PacketTunnelProvider.h 继承于 TunnelManager：

```
//
//  Created by leagsoft on 2022/8/31.
//

#import <NetworkExtension/NetworkExtension.h>
#import <UniConnectSDKTunnel/UniConnectSDKTunnel.h>

@interface PacketTunnelProvider : TunnelManager

- (void)startTunnelWithOptions:(nullable NSDictionary<NSString
    completionHandler:(void (^)(NSError * __nullable error))co

- (void)stopTunnelWithReason:(NEProviderStopReason)reason comp
    (^)(void))completionHandler;
```

(2) 调用方法

在 PacketTunnelProvider.m 中的三个代理中分别自调用各自代理方法

```
- (void)startTunnelWithOptions:(NSDictionary *)options completionHandler:(void (^)(NSError *))completionHandler {
    // Add code here to start the process of connecting the tunnel.
    NSLog(@"PacketTunnelProvider  startTunnelWithOptions:%@",options);
    [super startWithOptions:options completionHandler:completionHandler];
}

- (void)stopTunnelWithReason:(NEProviderStopReason)reason completionHandler:(void (^)(void))completionHandler {
    NSLog(@"PacketTunnelProvider  stopTunnelWithReason");
    // Add code here to start the process of stopping the tunnel.
    [super stopTunnelWithReason:reason completionHandler:completionHandler];
    completionHandler();
}

- (void)handleAppMessage:(NSData *)messageData completionHandler:(void (^)(NSData *))completionHandler {
    NSLog(@"PacketTunnelProvider handleAppMessage  messageData:%ld",messageData.length);
    // Add code here to handle the message.
    [super handleAppMessage:messageData completionHandler:completionHandler];
}
```

通过以上两个步骤，Packet Tunnel Provider 已经配置完成，不需要再做其他操作。

2. UniConnectSDK.framework 的使用

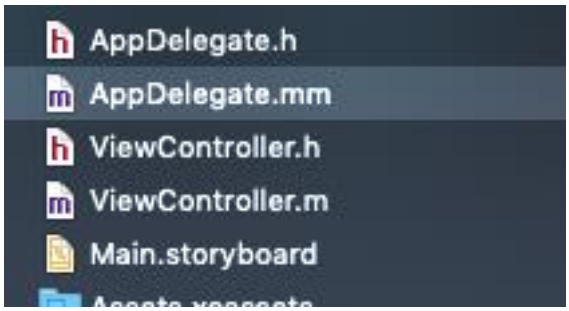
(1) 引入头文件

在 AppDelegate.m 中引入 “import <UniConnectSDK/SSLVPN.h>”，在 “didFinishLaunching” 中调用

“SSLVPN InitWithPacketTunnelBundleIdentify:” 方法进行初始化 SDK。

调用 “SSLVPN initLog:loglevel” 方法初始化日志。

注意：由于静态库含有 c++的代码，此时需要将 “AppDelegate.m” 改为 “AppDelegate.mm” 。



(2) VPN 连接状态

1) 主动获取。

可以通过获取 vpnStatus 的值，来主动获取 VPN 的连接状态:

```
VPNStatus status = [SSLVPN shareInstance].vpnStatus;
```

2) 被动获取。

可以通过设置代理，来被动接收 VPN 的连接状态:

```
[SSLVPN shareInstance].delegate = self;
```

在代理（self）中，实现 VPNStatusDelegate 的函数 -(void)didVPNStatusChanged:(VPNStatus)vpnStatus data:(_Nullable id)data; 可以被动接收状态变化信息。

```
27  @implementation HomeViewController
343 // MARK: - SSLVPN Delegate
344
345 - (void)didVPNStatusChanged:(VPNStatus)vpnStatus data:(_Nullable id)data {
346
347     dispatch_async(dispatch_get_main_queue(), ^{
348         NSString * statusStr = [self vpnStatusDesc:vpnStatus];
349         [DemoLog log:@"[onDelegate] didVPNStatusChanged:(%ld) data:%@",vpnStatus, statusStr, data];
350         self.statusLabel.text = statusStr;
351         switch(vpnStatus) {
352             case  VPNStatusInvalid:
353                 self.statusLabel.backgroundColor = UIColor.grayColor;
354                 break;
355             case  VPNStatusConnected:
356                 self.statusLabel.backgroundColor = UIColor.greenColor;
357                 break;
```

(3) VPN 状态说明

状态	说明
VPNStatusInvalid	未开始
VPNStatusDisconnected	未连接

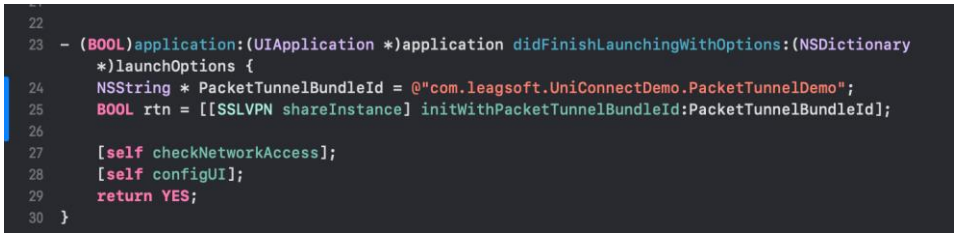
VPNStatusConnecting	连接中
VPNStatusConnected	已连接
VPNStatusReasserting	重连中
VPNStatusDisconnecting	断开连接中
VPNStatusConnectingWaitAuth	等待授权
VPNStatusConnectingWaitAuthSMS	等待二次认证

3.UniConnectSDK.framework 中的接口使用

(1) 初始化 SDK 接口

名称	- (BOOL)initWithPacketTunnelBundleId:(NSString *)bundleId;
功能	初始化 SDK

参数	参数名	类型	说明
	bundleId	NSString	networkExtension 的 bundleId
返回值	YES: 初始化成功 NO: 初始化失败		
备注	**注意：必须在初始化接口调用成功之后，才能正常使用其他接口。**		

示例	

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
    *)launchOptions {
    NSString * PacketTunnelBundleId = @"com.leagsoft.UniConnectDemo.PacketTunnelDemo";
    BOOL rtn = [[SSLVPN shareInstance] initWithPacketTunnelBundleId:PacketTunnelBundleId];

    [self checkNetworkAccess];
    [self configUI];
    return YES;
}
```

(2) 初始化日志接口

名称	- (BOOL)initLog:(NSString *)path loglevel:(int)level;
功能	初始化日志

参数	参数名	类型	说明
	path	NSString	日志存放路径
	level	int	日志等级,取值：[3,4] 3:打印 info、warning、error 日志，当日志等级为 4:打印 info、warning、error 及 debug 日志
返回值	YES: 初始化成功 NO: 初始化失败		

(3) 创建 SSLVPN 连接

名称	- (void)startConnectWithConfig:(ConnectionConfiguration *)config complete:(void (^)(ConnectErrorCode, id))complete;
功能	创建 SSLVPN 连接

参数	参数名	类型	说明
	config	ConnectionConfiguration	连接配置，参考 ConnectionConfiguration 结构说明。
	complete	block	连接返回结果
结果回调说明	ConnectErrorCodeSucceed		连接成功
	ConnectErrorCodeFailed		连接失败
	ConnectErrorCodeNotInit		SDK 未初始化
	ConnectErrorCodeParamError		参数错误
	ConnectErrorCodeStatusRefuse		当前状态不允许连接接。如连接未返回结果，又调用连接。
	ConnectErrorCodeRepeat		配置名称重复
	ConnectErrorCodeInvalid		配置不全或格式有误
	ConnectErrorCodeUserRefuse		首次保存配置到系统 vpn 配置时，用户拒绝。需要用户同意并输入终端锁屏密码
备注	**注意：VPN 创建连接结果需要刷新 UI 时，请在主线程进行操作**		

示例	

```
ConnectionConfiguration * config = [[ConnectionConfiguration alloc] init];
config.connectionName = @"VPN";
config.connectionType = @"SSL";
config.serverAddress = @"cstestarea.leagsoft.net";
config.serverPort = @"5557";
config.transferMode = RELIABLE_TRANSFER_MODE;
config.autoReconnect = NO;
config.isSecurity = NO;
config.isRoute = YES;
config.isTCPBuffer = FALSE;
config.isAutoGateway = FALSE;
config.smSignPath = @"xxxxx/国密证书路径/sm2.sig.pem";
config.smEncPath = @"xxxxx/国密证书路径/sm2.enc.pem";
config.cerPath = @"xxxxx/证书路径/xx.crt";
config.gatewayAdrArray = [NSArray array];

[[SSLVPN shareInstance] startConnectWithConfig:config complete:^(ConnectErrorCode code, id data) {
    if(code == ConnectErrorCodeSucceed) {
        self.serverAuthType = data;
        [DemoLog log:@"[startConnectWithConfig:complete:] end connectError:(%ld)%@ authType:%@",code,[self
            connectCodeDesc:code],data];
    }else {
        [DemoLog log:@"[startConnectWithConfig:complete:] end connectError:(%ld)%@ msg:%@",code,[self
            connectCodeDesc:code],data];
    }
}];
```

(4) 获取管理类单例对象

名称	+ (SSLVPN *)shareInstance;
功能	获取管理类单例对象
备注	管理类单例对象，调用接口，设置代理均要使用

(5) 授权

名称	- (void)startAuthWithConfig:(AuthConfiguration*)config complete:(void(^)(AuthResultCode code))complete;		
功能	授权		
参数	参数名	类型	说明
	config	AuthConfiguration	授权配置，参考AuthConfiguration结构说明。
	complete	block	授权返回结果
结果回调说明	AuthResultCodeSucceed		登录成功
	AuthResultVerityServerFailed		开启可信证书校验

		时，校验失败
	AuthResultCodeFirstLoginError	首次密码登陆,需要修改
	AuthResultCodeSMSCheckWarning	需要进行双因子验证
	AuthResultCodeLoginFailed	登录失败
	AuthResultCodeUserExpireWeakError	您的密码已过期，请改用网页登陆，并修改密码
	AuthResultCodeStatusRefuse	当前非等待授权状态
	AuthResultCodeCNEMFailed	网络扩展失败
	AuthResultCodeNotSupportUDP	Udp 隧道建立失败
示例		

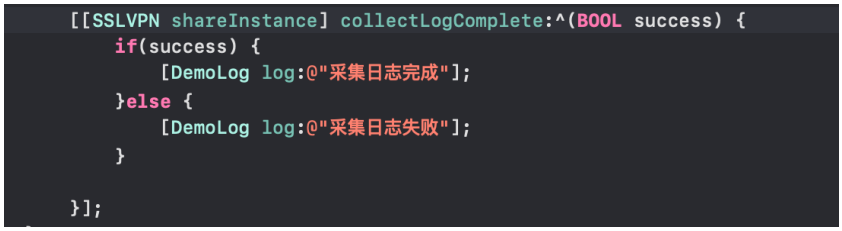
```
190
191     AuthConfiguration * authConfig = [[AuthConfiguration alloc] init];
192     authConfig.authType = AuthTypeAccountPwd; // 这个类型，是连接之后告知的类型。此处传值需要和后端告知的一致
193     authConfig.userName = @"zhangsan";
194     authConfig.password = @"123";
195     authConfig.isCACheck = NO;
196
197     [[SSLVPN shareInstance] startAuthWithConfig:authConfig complete:^(AuthResultCode resultCode) {
198         [DemoLog log:@"[startAuthWithConfig:complete] end resultCode:(%lu)%@",resultCode,[self
                authCodeDesc:resultCode]];
199     }];
```

(6) 关闭 VPN 连接

名称	- (void)disconnect;
功能	关闭 VPN 连接

(7) 收集日志

名称	- (void)collectLogComplete;
功能	收集日志
添加日志	需要在代码中添加日志需要引入“import <SecoSDK/WriteLog.h>”,直接使用“LOGWSERR(format,...)、LOGWSWARN(format,...)、LOGWSNOTICE(format,...)、LOGWSDEBUG(format,...)”进行添加

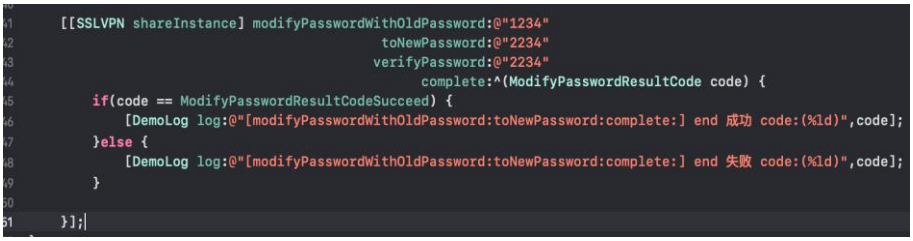
备注	日志将保存到主应用的沙盒中。
示例	

```
[[SSLVPN shareInstance] collectLogComplete:^(BOOL success) {
    if(success) {
        [DemoLog log:@"采集日志完成"];
    }else {
        [DemoLog log:@"采集日志失败"];
    }

}];
```

(8) 修改登录密码

名称	- (void)modifyPasswordWithOldPassword:(NSString *)oldPassword toNewPassword:(NSString *)newPassword verifyPassword:(NSString *)verifyPassword complete:(nullable void (^)(ModifyPasswordResultCode code))complete;		
功能	修改登录密码		
参数	参数名	类型	说明
	oldPassword	NSString	旧密码
	newPassword	NSString	新密码
	verifyPassword	NSString	确认密码
	complete	block	修改结果回调
结果回调说明	ModifyPasswordResultCodeSucceed		修改成功
	ModifyPasswordResultCodeOldPwdError		旧密码错误
	ModifyPasswordResultCodeIsLowError		修改密码时新密码不符合低密码策略
	ModifyPasswordResultCodeIsMiddleError		修改密码时新密码不符合中密码策略
	ModifyPasswordResultCodeIsHighError		修改密码时新密码不符合高密码策略

	ModifyPasswordResultCodeFailed	其他错误
	ModifyPasswordResultCodePasswordFmtError	密码格式错误
	ModifyPasswordResultCodeNewVerifyPasswordDiffError	新密码和确认密码不一致
	ModifyPasswordResultCodeNewPasswordSameError	新密码和旧密码一致
	ModifyPasswordResultCodeNotEnabled	不允许修改密码
备注	只能是在连接成功状态下才可以进行密码修改	
示例		

```
[[SSLVPN shareInstance] modifyPasswordWithOldPassword:@"1234"
                                           toNewPassword:@"2234"
                                          verifyPassword:@"2234"
                                                complete:^(ModifyPasswordResultCode code) {
    if(code == ModifyPasswordResultCodeSucceed) {
        [DemoLog log:@"[modifyPasswordWithOldPassword:toNewPassword:complete:] end 成功 code:(%ld)",code];
    }else {
        [DemoLog log:@"[modifyPasswordWithOldPassword:toNewPassword:complete:] end 失败 code:(%ld)",code];
    }

}];
```

(9) 双因子认证

名称	- (void)verificationCode:(NSString *)verificationCode complete:(void(^)(VerificationCodeResultCode code))complete;		
功能	双因子认证		
参数	参数名	类型	说明
	verificationCode	NSString	验证码
	complete	block	结果回调
结果回调说明	VerificationCodeResultCodeSucceed		双因子登陆成功
	VerificationCodeResultCodeFailed		双因子验证失败
	VerificationCodeResultCodeStatusRefuse		当前非等待双因子认证状态
	ModifyPasswordResultCodeNotEnabled		不允许修改密码

示例		

```
[[SSLVPN shareInstance] verificationCode:@"8848" complete:^(VerificationCodeResultCode code) {
    if(code == VerificationCodeResultCodeSucceed) {
        [DemoLog log:@"[verificationCode:complete:] end 成功 code:(%ld)",code];
    }else if(code == VerificationCodeResultCodeFailed) {
        [DemoLog log:@"[verificationCode:complete:] end 失败 code:(%ld)",code];
    }else if(code == VerificationCodeResultCodeStatusRefuse) {
        [DemoLog log:@"[verificationCode:complete:] end 当前状态此接口无效 code:(%ld)",code];
    }else {
        [DemoLog log:@"[verificationCode:complete:] end 错误code:(%ld)",code];
    }

}];
```

(10) 获取日志的文件夹绝对路径

名称	- (NSString *)getLogPath;
功能	获取日志的文件夹绝对路径
备注	

(11) 根据配置名称删除配置

名称	- (void)deleteConfigWithName:(NSString *)configName complete:(void (^)(DeleteConnectionConfigurationResultCode))complete;		
功能	根据配置名称删除配置		
参数	参数名	类型	说明
	configName	NSString	配置名
	complete	block	删除结果
结果回调说明	DeleteConnectionConfigurationResultCodeSucceed		删除成功
	DeleteConnectionConfigurationResultCodeFailed		删除失败

示例	[[SSLVPN shareInstance] deleteConfigWithName:@"VPN" complete:^(DeleteConnectionConfigurationResultCode result) { [DemoLog log:@"[deleteConfigWithName:complete:] result:(%ld)",result]; }];

4.通用结构说明

(1) ConnectionConfiguration 连接配置

属性	类型	说明
connectionName	NSString	连接名字
connectionType	NSString	连接类型 “SSL”
serverAddress	NSString	网关地址
serverPort	NSString	网关端口
transferMode	TRANSFER_MODE	隧道模式 RELIABLE_TRANSFER_MODE 可靠模式 FAST_TRANSFER_MODE 快速模式 ADAPTIVE_TANSFER_MODE 自适应模式
autoReconnect	BOOL	自动连接
isSecurity	BOOL	是否开启国密：0 关闭 1 开启 默认关闭
isRoute	BOOL	是否开启路由覆盖：0 关闭 1 开启 默认开启
isTCPBuffer	BOOL	是否开启 TCP 缓冲区:0 关闭 1 开启 默认关闭
isAutoGateway	BOOL	是否开启站点优选：0 关闭 1 开启 默认关闭
smSignPath	NSString	/国密签名证书路径 开启国密功能必传

smEncPath	NSString	国密加密证书路径 开启国密功能必传
cerPath	NSString	普通证书路径 开启证书功能必传（CA 证书，其他非国密认证证书）
gatewayAdrArray	NSArray	站点优选数组（开启站点优选必传）

(2) AuthConfiguration 授权配置

属性	**类型**	**说明**
authType	AuthType	授权类型，枚举。 AuthTypeAccountPwd：用户名密码 AuthTypeAnonymous：匿名登录 AuthTypeCert：证书挑战 AuthTypeCertAccountPwd：证书+用户名密码
userName	NSString	用户名
password	NSString	密码
certPath	NSString	证书文件路径
certPassword	NSString	证书密码
isCACheck	BOOL	是否验证服务器是否可信 0 不验证 1 验证 默认值：0
caCertFilePath	NSString	CA 证书文件路径