# DLink IOS SDK Integration Steps

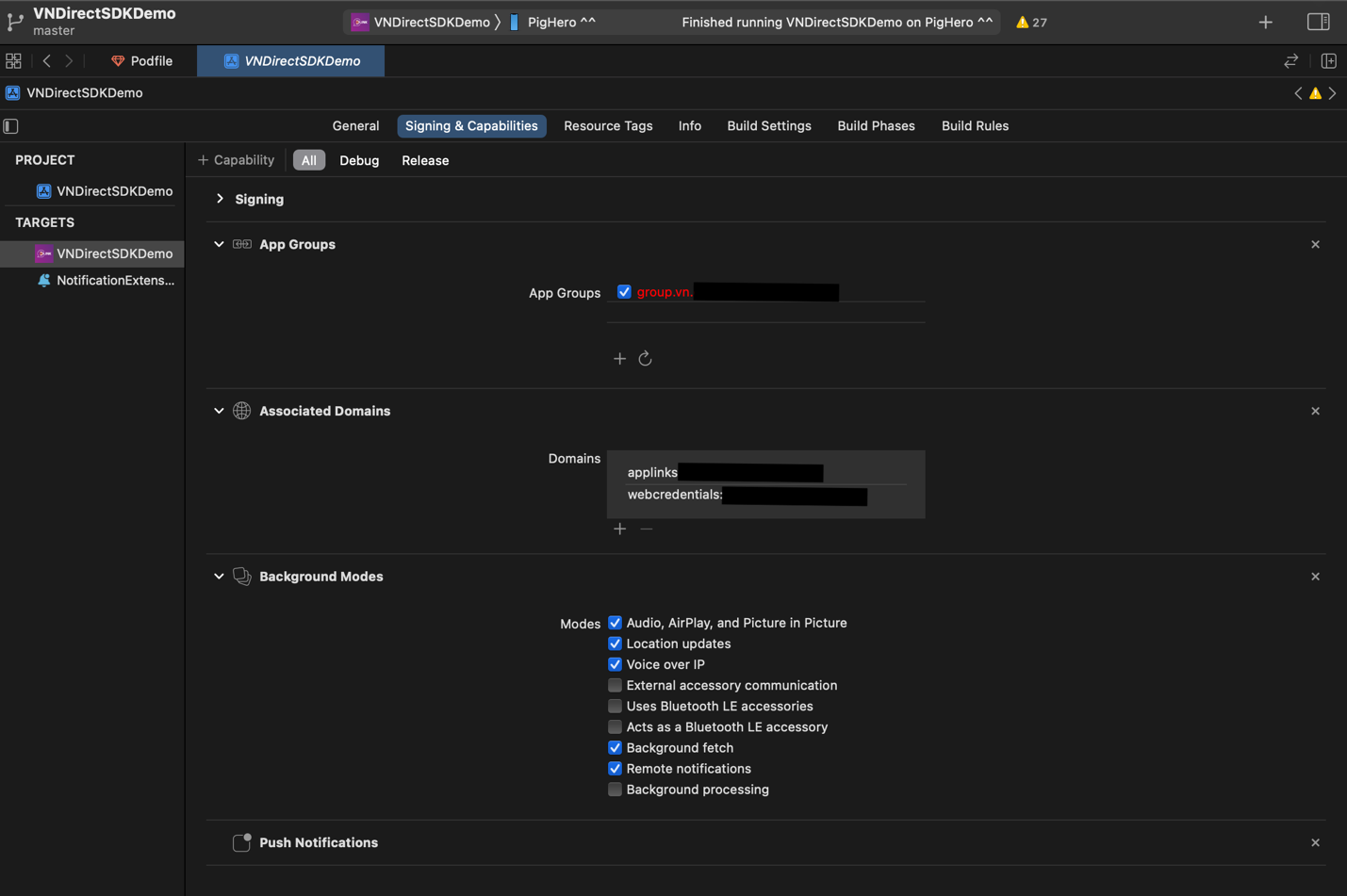
(Native)

## Versions history

|  |  |  |
| --- | --- | --- |
| Index | Date | Change logs |
| 1 | 2022/02/22 | Init document |
| 2 | 2022/02/22 | Add some comments |
| 3 | 2022/02/22 | Update github repo |
| 4 | 2022/02/22 | Config support calls feature |
| 5 | 2022/02/22 | Edit Info.plist |
| 6 | 2022/04/06 | Update SDK icon Chat & Call |
| 7 | 2022/06/21 | Show conversation with groupID |
| 8 | 2022/06/28 | Remove key NSUserTrackingUsageDescription  Hiden live location || location  Hiden GPS info |
| 9 | 2022/07/22 | Double push chat  Show Error description |
| 10 | 2022/09/16 | + Limit socket connecting when user trigger on SDK function  + Optimizing message flow |
| 11 | 2022/09/26 | + Update Swift – run real device  + Fix person call |
| 12 | 2022/10/14 | + Update Framework  + Update Podfile  + Update Document |
| 13 | 2022/10/24 | + Update Kingfisher, lottie-ios |
| 14 | 2022/11/09 | + Update Podfile  + Fix bug |

## Pre Requisites

* + Xcode 13.2.1 (13C100) (exactly version)
  + iOS 10.0 or above
  + Swift version 5 (exactly version)
  + Support bitcode: No
  + Support Features
    - Push notifications
    - Apps groups
    - Associated domains
    - Background modes: Audio, Location updates, Voice over IP, Background fetch, Remote notifications
    - I-cloud
    - Keychains
    - Notification extension service
    - Sharing extension service



## 1. Add DLink SDK to your project

* **Using Cocoapods**

Integrate Cocoapods <https://guides.cocoapods.org/using/using-cocoapods.html>

// **Edit Pod file**

platform :ios, '12.0'

use\_frameworks!

# keep\_source\_code\_for\_prebuilt\_frameworks!

inhibit\_all\_warnings!

# ======================================GROUP PODS==========================================

def netalo\_pods

pod 'NetacomSDKs', :git => 'https://github.com/Netacom-NetAlo/NetaloSDKs-iOS', tag: '0.1.2'

noti\_netalo\_pods

end

def noti\_netalo\_pods

pod 'NotificationSDK', :git => 'https://github.com/Netacom-NetAlo/NotiSDKs-iOS', tag: '0.1.1'

pod 'WebRTC', :git => 'https://github.com/Netacom-NetAlo/WebRTC-iOS', branch: 'main'

end

def resolver

pod 'Resolver', :git => 'https://github.com/Netacom-NetAlo/Resolver-iOS', branch: 'main'

end

# ======================================TARGET PODS==========================================

def app\_pods

pod 'MessageKit', :git => 'https://github.com/Netacom-NetAlo/Messagekit-iOS'

resolver

netalo\_pods

end

target 'VNDirectSDKDemo' do

app\_pods

end

# ============================Notification Extension================================

def app\_notification\_pods

resolver

noti\_netalo\_pods

end

target 'NotificationExtension' do

app\_notification\_pods

end

# Disable Bitcode because of JitsiMeetSDK

post\_install do |installer|

installer.pods\_project.targets.each do |target|

target.build\_configurations.each do |config|

config.build\_settings['ENABLE\_BITCODE'] = 'NO'

config.build\_settings['ARCHS'] = 'arm64 x86\_64'

config.build\_settings['EXCLUDED\_ARCHS[sdk=iphonesimulator\*]'] = 'arm64'

end

end

end

Replace the YOUR-BRANCH in actual implement

**// Info.plist**

Edit Info.plist File (See demo for more details)

**<key>**NSCameraUsageDescription**</key>**

**<string>**Sending photo and video call**</string>**

**<key>**NSContactsUsageDescription**</key>**

**<string>**NetAlo will continously upload your contacts to its heavily encrypted cloud servers to let you connect with your friends quickly**</string>**

**<key>**NSLocationAlwaysAndWhenInUseUsageDescription**</key>**

**<string>**If you’d like to share your Location with friends, DLink needs location access to use this feature**</string>**

**<key>**NSLocationAlwaysUsageDescription**</key>**

**<string>**If you’d like to share your Live Location with friends, DLink needs location access when the app is in background**</string>**

**<key>**NSLocationWhenInUseUsageDescription**</key>**

**<string>**If you’d like to share your Location with friends, DLink needs location access to use this feature**</string>**

**<key>**NSMicrophoneUsageDescription**</key>**

**<string>**Using voice call**</string>**

**<key>**NSPhotoLibraryUsageDescription**</key>**

**<string>**Sending photo and updating your profile**</string>**

* **Install dependencies**

pod install

## 2. Initialization of SDK

* **Prepare configurations**
  + AppBundle (ID, AppGroupID, Push P12)
    - ID: The unique ID register from apple developer portal. Ref: <https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/AppID.html>
    - AppGroupID: The ID from app group feature in apple developer portal. Ref: <https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_application-groups>
    - Push P12: The exported certificates use for push notifications. Ref: <https://developer.apple.com/documentation/usernotifications/setting_up_a_remote_notification_server/establishing_a_certificate-based_connection_to_apns>
  + DLink App (id, key, adminKey): Request DLink team to provide this information
  + Regis your app with DLink team: Provide the information in AppBundle to register your app to work with DLink ecosystem

* **Import SDK**

import NetAloFull

* **Pre-defined environments**  
  struct BuildConfig {

static var config = NetaloConfiguration(

enviroment: .production,

appId: "**YOUR-APP-ID**",

appKey: "**YOUR-APP-KEY**",

accountKey: "**YOUR-ACCOUNT-KEY**",

appGroupIdentifier: "**YOUR-APP-GROUP**",

storeUrl: URL(string: "https://apps.apple.com/vn/app/vndirect/id1594533471")!,

analytics: [],

featureConfig: FeatureConfig(

user: FeatureConfig.UserConfig(

forceUpdateProfile: false,

allowCustomUsername: false,

allowCustomProfile: false,

allowCustomAlert: false,

allowAddContact: false,

allowBlockContact: false,

allowSetUserProfileUrl: false,

allowEnableLocationFeature: false,

allowTrackingUsingSDK: false,

isHiddenEditProfile: true,

allowAddNewContact: false,

allowEditContact: false

),

chat: FeatureConfig.ChatConfig(

isVideoCallEnable: true,

isVoiceCallEnable: true,

isHiddenSecretChat: true

),

isSyncDataInApp: true,

allowReferralCode: false,

searchByLike: true,

allowReplaceCountrycode: false,

isSyncContactInApp: true

),

permissions: []

)

}

**// Implement user**

NetAloUserHolder(

id: 1234566789,

phoneNumber: "**your-phone**",

email: "",

fullName: "**your-Name**",

avatarUrl: "",

session: "**token-key**")

Replace the YOUR-APP-ID, YOUR-APP-KEY, YOUR-ACCOUNT-KEY, YOUR-APP-GROUP in the following code snippet with the actual appId, appKey, accountKey, appGroupIdentifier

* **Init the SDK**

@main

class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {

// MARK: Stored properties

private lazy var mainWindow = UIWindow(frame: UIScreen.main.bounds)

public var netAloSDK: NetAloFullManager!

private var disposeBag = DisposeBag()

// MARK: - Application Delegate

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

self.netAloSDK = NetAloFullManager(

config: BuildConfig.config

)

// Only show SDK after start success, Waiting maximun 10s

self.netAloSDK

.start()

.timeout(.seconds(10), scheduler: MainScheduler.instance)

.catchAndReturn(())

.withUnretained(self)

.observe(on: MainScheduler.instance)

.do(onNext: { (owner, \_) in

// Init rooter

let mainTabBarController = DemoMainTabbarVC()

mainTabBarController.initialize(sdk: owner.netAloSDK)

owner.netAloSDK.buildSDKModule()

owner.mainWindow.rootViewController = mainTabBarController

owner.mainWindow.makeKeyAndVisible()

//TODO -Make sure client must login successed to set user info here

owner.setUser()

})

.subscribe()

.disposed(by: disposeBag)

// Optional use notification

self.netAloSDK.requestNotificationtPermission()

UNUserNotificationCenter.current().delegate = self

return netAloSKD.application(application, didFinishLaunchingWithOptions: launchOptions)

}

public func setUser() {

let user = NetAloUserHolder(

id: 1234566789,

phoneNumber: "**your-phone**",

email: "",

fullName: "**your-name**",

avatarUrl: "",

session: "**Token-key**")

do {

try self.netAloSDK.set(user: user)

} catch let e {

print("Error \(e)")

}

bindingService()

}

// MARK: - AppDelegateViewModelOutputs

public func applicationDidBecomeActive(\_ application: UIApplication) {

netAloSDK.applicationDidBecomeActive(application)

}

public func applicationWillResignActive(\_ application: UIApplication) {

netAloSDK.applicationWillResignActive(application)

}

public func applicationWillTerminate(\_ application: UIApplication) {

netAloSDK.applicationWillTerminate(application)

}

func application(\_ application: UIApplication, supportedInterfaceOrientationsFor window: UIWindow?) -> UIInterfaceOrientationMask {

netAloSDK.application(application, supportedInterfaceOrientationsFor: window)

}

// UserActivity

func application(\_ application: UIApplication, continue userActivity: NSUserActivity, restorationHandler: @escaping ([UIUserActivityRestoring]?) -> Void) -> Bool {

netAloSDK.application(application, continue: userActivity, restorationHandler: restorationHandler)

}

// Notification methods

public func application(\_ application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {

netAloSDK.application(application, didRegisterForRemoteNotificationsWithDeviceToken: deviceToken)

}

func application(\_ application: UIApplication, open url: URL, sourceApplication: String?, annotation: Any) -> Bool {

netAloSDK.application(application, open: url, sourceApplication: sourceApplication, annotation: application)

}

func application(\_ app: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {

netAloSDK.application(app, open: url, options: options)

}

// MARK: - UNUserNotificationCenterDelegate

public func userNotificationCenter(\_ center: UNUserNotificationCenter, willPresent notification: UNNotification, withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {

netAloSDK.userNotificationCenter(center, willPresent: notification, withCompletionHandler: completionHandler)

}

public func userNotificationCenter(\_ center: UNUserNotificationCenter, didReceive response: UNNotificationResponse, withCompletionHandler completionHandler: @escaping () -> Void) {

netAloSDK.userNotificationCenter(center, didReceive: response, withCompletionHandler: completionHandler)

}

}

* **Set current user (Required)**

public func setUser() {

let user = NetAloUserHolder(

id: 1234566789,

phoneNumber: "**your-phone**",

email: "",

fullName: "**your-name**",

avatarUrl: "",

session: "**Token-key**")

do {

try self.netAloSDK.set(user: user)

} catch let e {

print("Error \(e)")

}

bindingService()

}

* **Handle NetAloSDK delegates (Required)**

**//MARK: SDK call back**

extension AppDelegate {

//SDK binding service

private func bindingService() {

self.netAloSDK.eventObservable

.asDriverOnErrorJustSkip()

.drive(onNext: { [weak self] event in

guard let self = self else { return }

dump("Event \(event)")

switch event {

case .pressedUrl(let url):

dump("PressedUrl: \(url)")

case .mediaURL(let imageUrls, let videoUrls):

dump("Images: \(imageUrls)")

dump("Video: \(videoUrls)")

case .checkUserIsFriend(let userId):

dump("Check Chat with: \(userId)")

case .didCloseSDK:

dump("didCloseSDK ")

case .pressedCall(let type):

dump("pressedCall type: \(type)")

case .sessionExpired:

dump("sessionExpired")

case .updateBadge(let badge) :

dump("updateBadge: \(badge)")

default: break

}

})

.disposed(by: disposeBag)

}

}

## 3. Use SDK DLink

* **Open list Group:**

self.netaloSDK?.showVNDemoVC(completion: { error in

let err = error as? NAError

print("showVNDemoVC with err: \(err?.description ?? "")")

})

* **Open chat 1-1 (with user want to chat):**

let testContact = NAContact(id: 123456, phone: "phone", fullName: "fullname", profileUrl: "url")

self.netaloSDK?.showChat(with: testContact, completion: { error in

let err = error as? NAError

print("showChat with err: \(err?.description ?? "")")

})

* **Open call (with user want to call):**

let testContact = NAContact(id: 123456, phone: "phone", fullName: "fullname", profileUrl: "url")

self.netaloSDK?.showCall(with: testContact, isVideoCall: false)

* **SEND MESSAGE FUNCTION:**

self.netaloSDK?.sendMessage(content: "SDK TEST SEND MESSAGE", targetUserID: "281474977725116", completion: { \_status in

print("SDK TEST SEND STATUS: \(\_status)")

})

* **Open conversation with groupID:**

self.netaloSDK?.showGroupChat(with: {GROUPID}, completion: { error in

let err = error as? NAError

print("showGroupChat with err: \(err?.description ?? "")")

})

**Caution:**

Parameters:

+ id: TargetUserID

+ phone: **TargetPhone** Or **TargetUserName**

+ fullName: TargetFullName

+ profileUrl: TargetProfileUrl

* **Logout:**

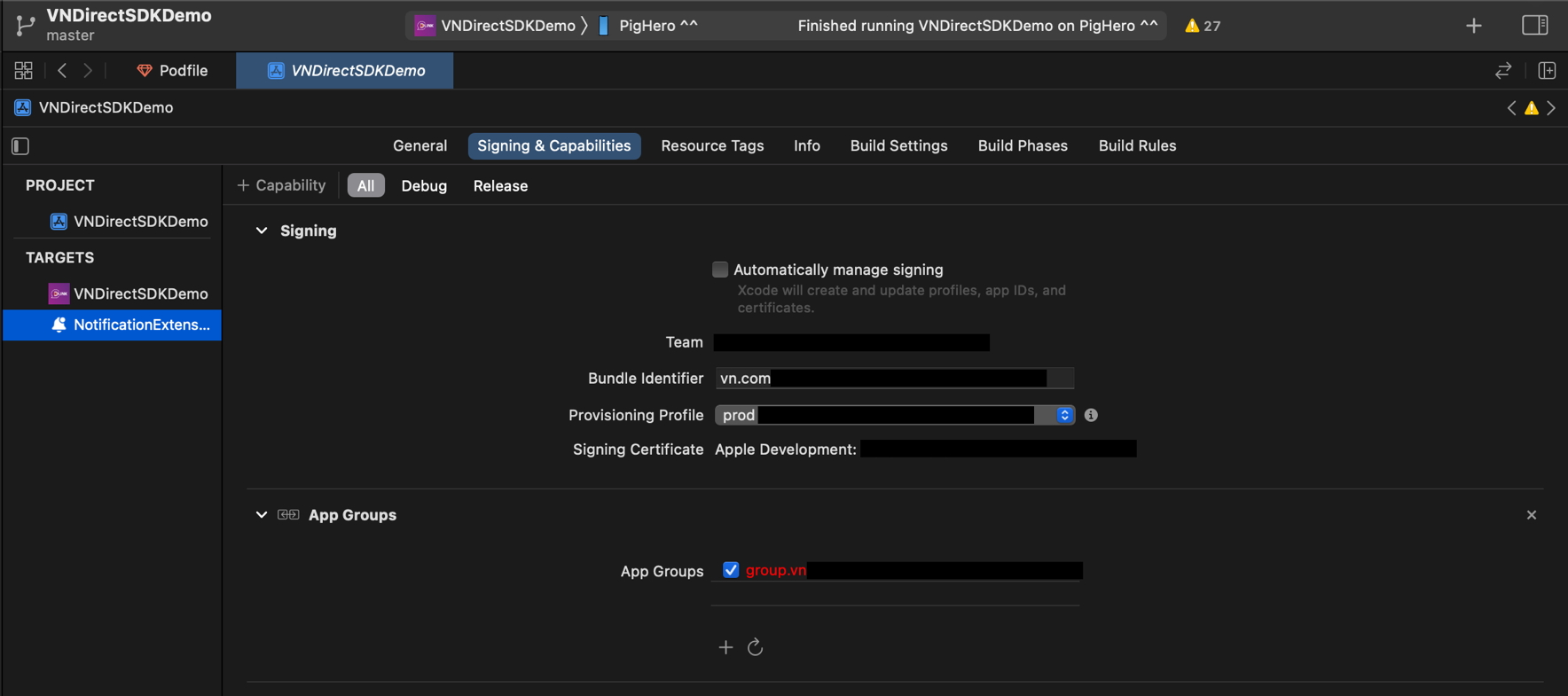
netAloSDK?.Logout()

## 4. Use Push Notification (For calls & messages)

* **Config builds**

Please follow **Section 2**

* **Config extension (For messages)**



* **Pre-defined environments**

**// BuildConfig.swift**

import NetAloLite

struct BuildConfig {

static var enviroment = EnvironmentType.production

static var appGroupId = "APP-GROUP" // note: same with main app

* **Init the SDK**

**// Injection.swift**

import Resolver

import NotificationComponent

extension MyResolver: ResolverRegistering {

public static func registerAllServices() {

register { NotificationComponentImpl(enviroment: BuildConfig.enviroment, appGroupId: BuildConfig.appGroupId) as NotificationComponentProtocol }

.scope(.cached)

}

}

// **NotificationService**.swift

} import UserNotifications

import RxSwift

import Resolver

import NotificationComponent

class NotificationService: UNNotificationServiceExtension {

var contentHandler: ((UNNotificationContent) -> Void)?

var bestAttemptContent: UNMutableNotificationContent?

private var disposeBag = DisposeBag()

@LazyInjected private var notificationRepo: NotificationComponentProtocol

override init() {

super.init()

notificationRepo.initialize()

}

override func didReceive(\_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void) {

self.contentHandler = contentHandler

bestAttemptContent = (request.content.mutableCopy() as? UNMutableNotificationContent)

if let bestAttemptContent = bestAttemptContent {

notificationRepo.replace(oldContent: bestAttemptContent)

.do(onSuccess: { (newContent) in

contentHandler(newContent)

})

.subscribe()

.disposed(by: disposeBag)

}

}

override func serviceExtensionTimeWillExpire() {

// Called just before the extension will be terminated by the system.

// Use this as an opportunity to deliver your "best attempt" at modified content, otherwise the original push payload will be used.

if let contentHandler = contentHandler, let bestAttemptContent = bestAttemptContent {

contentHandler(

notificationRepo.expired(oldContent: bestAttemptContent)

)

}

}

}

* **Change sound ring notification**

Use file: notification\_sound.wav

if not found sound ring use default device.

## 5. Use Option SDK DLink

* **Custom Theme color for app**

Currently iOS SDK does not support config Theme. So each theme will have a custom build.

* **Block/Unblock User**

netAloSDK. blockUser(blockedID: "USER\_ID", completion: @escaping (Bool) -> Void)

netAloSDK.unBlockUser(blockedID: "USER\_ID", completion: @escaping (Bool) -> Void)

* **Set Language**

netAloSDK.setLanguage(language: String)

## 6. Contact

Email: [sondh@netacom.vn](mailto:sondh@netacom.vn)

iOS developer

Mobile: (+84) 906600132

Headquarters:

5 Dien Bien Phu St., Dien Bien Ward, Ba Dinh District, Hanoi City, Vietnam

Ho Chi Minh City Office:

Ground Floor, Centre Point Tower, 106 Nguyen Van Troi St., Ward 8, Phu Nhuan District, HCMC, Vietnam.

## 7. Example

<https://github.com/Netacom-NetAlo/DLinkSDK-iOS-Demo>