


iOS Installation Guide

The Zello SDK for iOS is a native framework that connects to Zello servers for tasks such as authentication and message sending.

Prerequisites

The Zello SDK requires a recent version of Xcode. Your application must have an iOS deployment target of 14.0 or above.

 Even though the Zello SDK will build and run on iOS 14.0 and above, it can only receive messages in the background and support push-to-talk functionality on iOS 17.2 and above.

The Zello SDK is distributed through [CocoaPods](#). If you have not yet installed CocoaPods, follow the [CocoaPods getting started guide](#).

Installing the SDK

Add the CocoaPods Dependency

In your application's Podfile, add the following build setting overrides:

```
1 post_install do |installer|
2   installer.pods_project.targets.each do |target|
3     target.build_configurations.each do |config|
4       config.build_settings["IPHONEOS_DEPLOYMENT_TARGET"] = "14.0" # or higher
5       config.build_settings['BUILD_LIBRARY_FOR_DISTRIBUTION'] = "YES"
6     end
7   end
8 end
```

Then, in the `target` for your application target, add the `ZelloSDK` pod, for example:

```
1 target 'MyApplication' do
2   use_frameworks!
3   pod "ZelloSDK", "~> 0.3.1"
4 end
```

On the command line, run the following commands:

```
1 pod install --repo-update
2 open *.xcworkspace
```





Configure the Push to Talk Framework

For devices running on iOS 17.2 and later, Zello's SDK uses Apple's Push to Talk framework to send and receive voice messages in the background. This new framework has some limitations and requires both you and Zello to perform some additional setup steps. For more information, see the Framework Setup and Push Notifications sections below.

On [Apple Developer](#), create an identifier for your application. In the **Capabilities** tab, enable the **Push Notifications** and **Push to Talk** capabilities:

[← All Identifiers](#)

Edit your App ID Configuration

<input type="checkbox"/>	 On Demand Install Capable for App Clip Extensions ⓘ	
<input type="checkbox"/>	 Personal VPN ⓘ	
<input checked="" type="checkbox"/>	 Push Notifications ⓘ	Configure Certificates (0)
<input checked="" type="checkbox"/>	 Push to Talk ⓘ	

Allow Zello to Run in the Background

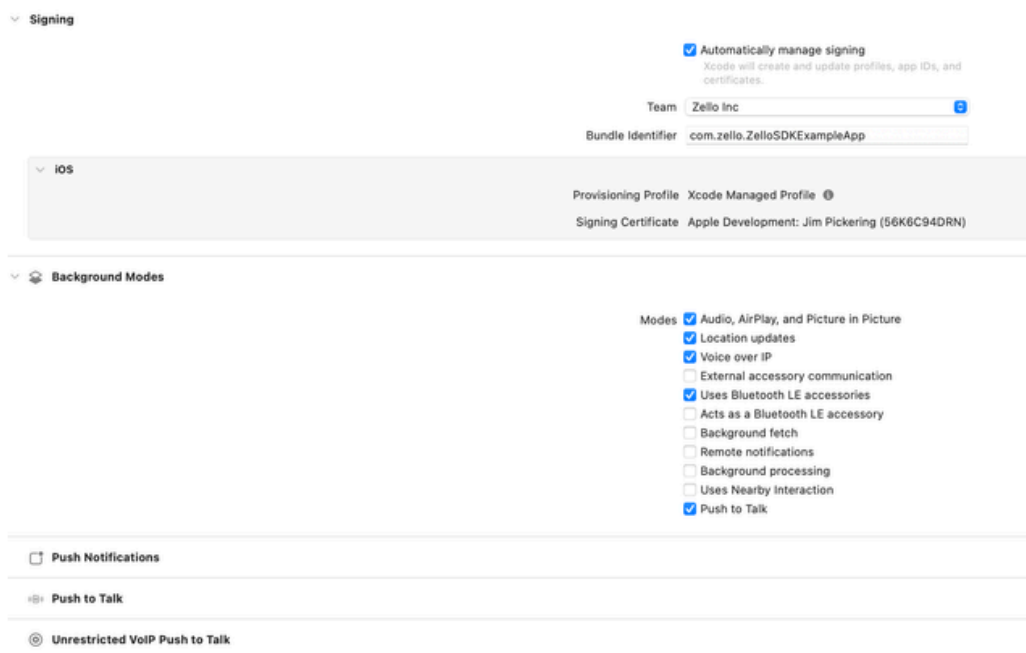
The Zello SDK uses push notifications and the Push to Talk framework so that the user can continue to receive messages from others even while it is running in the background.

In Xcode, navigate to your project. At the top of the **Signing & Capabilities** tab, enable the following background modes:

- Audio, AirPlay, and Picture in Picture
- Location updates
- Voice over IP
- Uses Bluetooth LE accessories
- Push to Talk

At the top of the tab, click the **+ Capability** button and add the following capabilities:

- Push Notifications
- Push to Talk



The screenshot shows the Xcode 'Signing & Capabilities' tab for a project named 'Zello Inc' with a bundle identifier of 'com.zello.ZelloSDKExampleApp'. The 'Signing' section is expanded, showing 'Automatically manage signing' checked, and the 'Provisioning Profile' set to 'Xcode Managed Profile'. The 'Background Modes' section is also expanded, showing a list of modes with checkboxes: 'Audio, AirPlay, and Picture in Picture' (checked), 'Location updates' (checked), 'Voice over IP' (checked), 'External accessory communication' (unchecked), 'Uses Bluetooth LE accessories' (checked), 'Acts as a Bluetooth LE accessory' (unchecked), 'Background fetch' (unchecked), 'Remote notifications' (unchecked), 'Background processing' (unchecked), 'Uses Nearby Interaction' (unchecked), and 'Push to Talk' (checked). Below the 'Background Modes' section, the 'Push Notifications' and 'Push to Talk' capabilities are listed, along with 'Unrestricted VoIP Push to Talk'.

Your application target should include an .entitlements file that includes the [APS Environment](#) and [Push to Talk](#) entitlements.

To configure your application to receive push notifications in a development build, you must call `Zello/configure(isDebugBuild:)` before `Zello/connect(credentials:)`, passing in `true`.

```

1 #if DEBUG
2     Zello.shared.configure(isDebugBuild: true)
3 #else
4     Zello.shared.configure(isDebugBuild: false)
5 #endif

```

Request User Permission to Access System Features

In the **Info** tab, add the following entries, setting them to brief messages to the user about how your application will use these system features:

- Privacy - Bluetooth Peripheral Usage Description (`NSBluetoothPeripheralUsageDescription`)
- Privacy - Bluetooth Always Usage Description (`NSBluetoothAlwaysUsageDescription`)
- Privacy - Location Always Usage Description (`NSLocationAlwaysUsageDescription`)
- Privacy - Microphone Usage Description (`NSMicrophoneUsageDescription`)

Before calling `startVoiceMessage(contact:)`, import `AVFAudio` and [request the user's permission to record audio](#):

```

1 if #available(iOS 17, *) {
2     AVAudioApplication.requestRecordPermission { granted in }
3 } else {
4     AVAudioSession.sharedInstance().requestRecordPermission { granted in }
5 }

```

In the file where you implement the `UIApplicationDelegate` protocol, import `CoreBluetooth` and `CoreLocation`. Your `UIApplicationDelegate` implementation should conform to the `CLLocationManagerDelegate` protocol and have instance properties that hold a strong reference to a `CBCentralManager` object and a `CLLocationManager` object. In your `UIApplicationDelegate.application(_:didFinishLaunchingWithOptions:)` implementation, initialize these objects and request Location Services permissions:

```

1 centralManager = CBCentralManager(delegate: nil, queue: nil)
2 locationManager = CLLocationManager()
3 locationManager?.delegate = self
4 locationManager?.requestAlwaysAuthorization()


```

For more information, see “[Configuring your app to use location services](#)”.

- [-] The Push to Talk framework will run from when your application calls `connect(credentials:)` until it calls `disconnect()`. To stop the Push to Talk framework from operating in the background, the user presses the **Leave** button provided by the framework. When the application returns to the foreground, the SDK rejoins the push-to-talk channel.

Send Notifications to the User

[Ask the user's permission to send them local notifications](#) and post local notifications in your implementation of `Zello.Delegate` where appropriate.

 Zello uses Apple's push notification server to deliver push notifications. [E-mail us your APN key](#) to enable push notifications for your network. The APN key will be stored securely on our server.

Troubleshooting

No Such Module

If your application fails to build with the compiler errors such as:

```
1 Command SwiftCompile failed with a nonzero exit code
2   Unable to open base configuration reference file '...'
3 No such module 'ZelloSDK'
```

make sure you've opened a .xcworkspace file in Xcode, not a .xcodeproj file.

Symbol Not Found

If your application crashes on launch with an error such as:

```
1 dyld[17207]: Symbol not found: _$s14PhoneNumberKit0aB6Format04e164yA2CmFWC
2   Referenced from: <478D6EC4-00BF-3B2E-BBE4-35A6D493A6B2>
```

make sure you've added the `BUILD_LIBRARY_FOR_DISTRIBUTION` build setting to your Podfile as described in [Add the CocoaPods Dependency](#).