Sample Code

Receiving Voice and Text Communications on a Local Network

Provide voice and text communication on a local network isolated from Apple Push Notification service by adopting Local Push Connectivity.

Download

iOS 26.0+ | iPadOS 26.0+ | macOS 11.0+ | Xcode 17.0+

Overview

This sample app shows how to implement and use Local Push Connectivity within a messaging app. The sample workspace contains two app components:

- `SimplePush` — An iOS app that uses Local Push Connectivity to provide text messaging and VoIP services.
- `SimplePushServer` — A lightweight Swift server that simulates the functions of a messaging server by routing messages and calls between clients.

The `SimplePush` app maintains two connections to the server:

- Notification Channel — A TCP connection the `NEAppPushProvider` maintains that provides functionality similar to Apple Push Notification service (APNs) when on a local Wi-Fi or private LTE network.
- Control Channel — A TCP connection the iOS app maintains and uses to send app control data to the server.

To run the sample, you need a Mac to operate as the server, and two iOS devices (such as an iPhone and an iPad, or an iPod touch and an iPhone) that can connect to the server to communicate with each other.

Configure the Sample Code Project

Apps that use Local Push Connectivity require the App Push Provider entitlement.

After you receive the entitlement, do the following:

1. Open `SimplePush.xcworkspace` and set the development team in the build settings of all five targets across the three projects. After you've set the team, Xcode automatically creates unique bundle identifiers that you use in the following steps.

2. Sign into your account on the Apple Developer website and register a new App Group and two App IDs: one for the SimplePush app and one for the SimplePushProvider extension. Configure each App ID with the App Groups and Network Extensions capabilities. After you register the IDs, create two provisioning profiles (one for each identifier) that include the App Push Provider entitlement.

3. Import the newly created provisioning profiles into Xcode.

4. In the SimplePush project, configure the `SimplePush` and `SimplePushProvider` targets with the imported provisioning profiles, deselect the existing App Group, and select the new App Group identifier you created in step 2.

5. Update the `pushProviderBundleIdentifier` in `PushConfigurationManager.swift` with the bundle identifier set on the `SimplePushProvider` target.

Build and Run the Sample Server

Select the `SimplePushServer` build scheme and your Mac as the run destination, then run the project to start the server.

Build, Run, and Configure the Sample App

With the server running, select the `SimplePush` iOS build scheme and run the project on your iOS device. When `SimplePush` starts, update the app's settings to connect to the server. Tap the Settings button and enter the following information:

1. Enter the Server Address, which is the IP address or hostname of the Mac where `SimplePush Server` is running.

2. Enter one or more of the following network configurations:

   - Wi-Fi — Enter the SSID of your local Wi-Fi network. The `NEAppPushProvider` runs when your device joins the Wi-Fi network with the SSID you've specified.

   - Cellular—Enter the Mobile Country Code and Mobile Network Code of your cellular provider. If the device is connected to a Band 48 CBRS cellular network, enter the Tracking Area Code for the network.

- Ethernet — The `NEAppPushProvider` runs when your device is connected to an Ethernet network and the ethernet network is the primary route on the device.

> **Important**
>
> All cellular networks other than Band 48 CBRS require the device to be supervised for `NEAppPushProvider` to run. For more information, see Supervision of Apple devices.

After configuring the settings above, the Local Push Connectivity - Active setting displays “Yes” when `NEAppPushProvider` is running. You must perform the steps above on at least two iOS devices so you can test and observe message exchanges between clients.

See Also

Essentials

`class NEAppPushManager`

An object that configures a push provider and manages its life cycle.

`class NEAppPushProvider`

An object that creates and maintains a persistent network connection to a local push server.

Maintaining a Reliable Network Connection

Implement your Local Push Connectivity app to ensure delivery of notifications.