

【iOS10 Swift】 プッシュ通知を組み込もう！

2016/09/27作成 (2017/09/13更新)



概要

- ニフティクラウドmobile backendの『プッシュ通知』機能を実装したサンプルプロジェクトです
- 簡単な操作ですぐに ニフティクラウドmobile backendの機能を体験いただけます★☆☆
- このサンプルはSwift3(iOS10)に対応しています
 - Swift2のサンプルは[こちら](#)

ニフティクラウドmobile backendとは

スマートフォンアプリのバックエンド機能（プッシュ通知・データストア・会員管理・ファイルストア・SNS連携・位置情報検索・スクリプト）が**開発不要**、しかも基本**無料**(注1)で使えるクラウドサービス！



注1：詳しくは[こちら](#)をご覧ください

準備

準備するもの

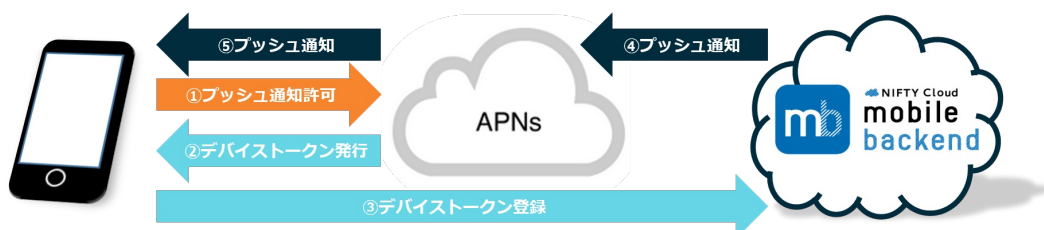
- ニフティクラウド mobile backend 会員登録
 - 下記リンクより登録（無料）をお願いします
<http://mb.cloud.nifty.com/>
- Mac と以下の環境
 - Xcode ver.8 以上推奨
- 動作確認用端末
 - iPhone ver.10 以上推奨
- Lightning ケーブル

参考：検証済み動作環境

- macOS Sierra 10.12.5
- Xcode ver. 8.3.3
- iPhone 6+ ver. 10.0.1
 - このサンプルアプリは、実機ビルドが必要です

プッシュ通知の仕組み

- ニフティクラウドmobile backendのプッシュ通知は、iOSが提供している通知サービスを利用しています
 - iOSの通知サービス **APNs（Apple Push Notification Service）**



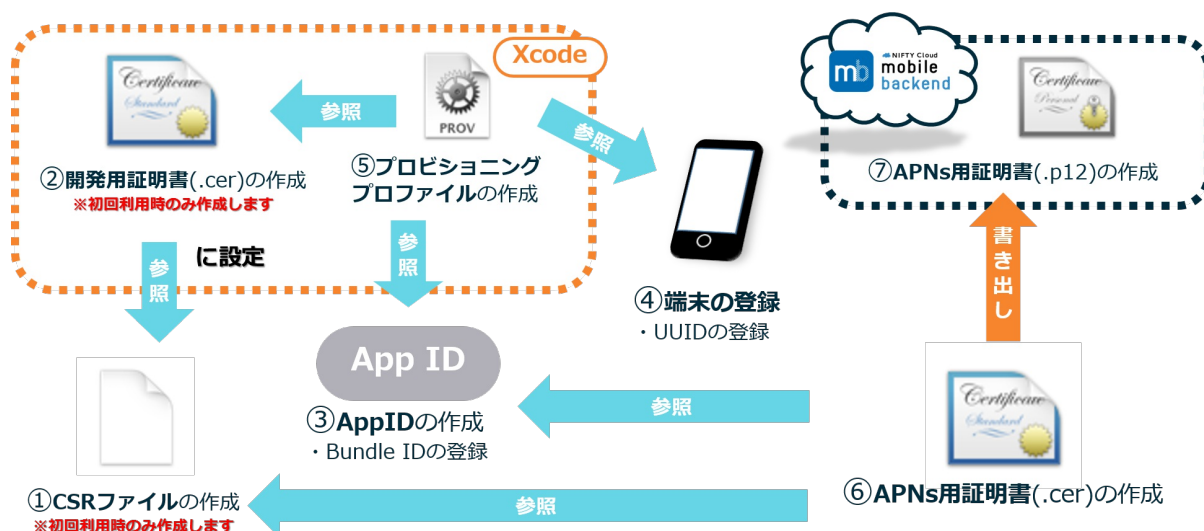
- 上図のように、アプリ（Xcode）・サーバー（ニフティクラウド mobile backend）・通知サービス（APNs）の間でやり取りを行うため、認証が必要になります
 - 認証に必要な鍵や証明書の作成は作業手順の「0.プッシュ通知機能使うための準備」で行います

作業の手順

0.プッシュ通知機能使うための準備

【iOS】 プッシュ通知の受信に必要な証明書の作り方(開発用)

- 上記のドキュメントをご覧の上、必要な証明書類の作成をお願いします
 - 証明書の作成には[Apple Developer Program](#)の登録（有料）が必要です



1. ニフティクラウド mobile backend の準備

- ニフティクラウド mobile backend にログインします

<http://mb.cloud.nifty.com/>



- 新しいアプリを作成します
- アプリ名を入力し、「新規作成」をクリックします
 - 例) **PushDemo**



- mobile backend を既に使用したことがある場合は、画面上方のメニューバーにある「+新しいアプリ」をクリックすると同じ画面が表示されます

- アプリ作成されると下図のような画面になります
- この2種類のAPIキー（アプリケーションキーとクライアントキー）はこの後 iOSアプリ との連携のために使用します



- 続けてプッシュ通知の設定を行います
- 「APNs用証明書(.p12)」を設定します

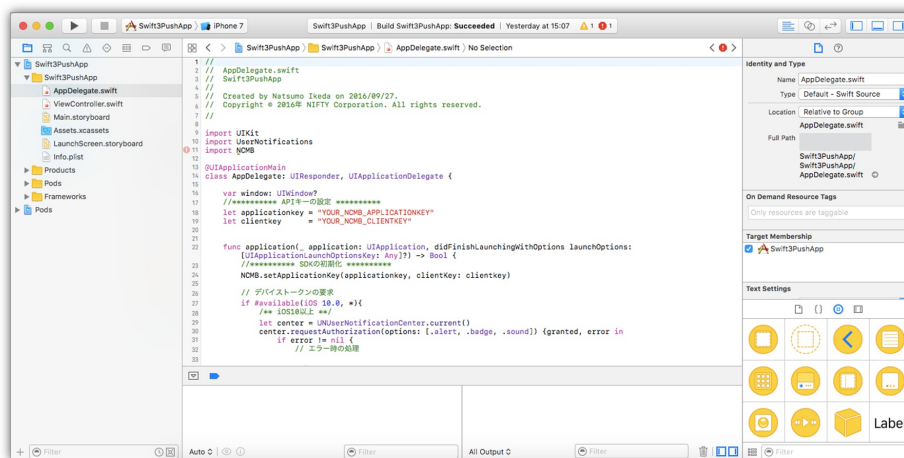


2. サンプルプロジェクトのダウンロード

- 下記リンクからプロジェクトをMacにダウンロードします
<https://github.com/natsumo/Swift3PushApp/archive/master.zip>

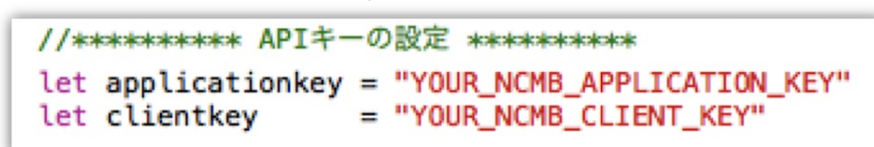
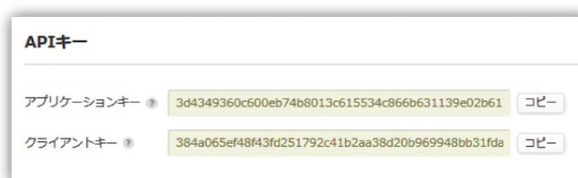
3. Xcodeでアプリを起動

- ダウンロードしたフォルダを開き、
「Swift3PushApp.xcworkspace」をダブルクリックしてXcode開きます(白い方です)
 - 「Swift3PushApp.xcodeproj」 (青い方) ではないので注意！



4. APIキーの設定

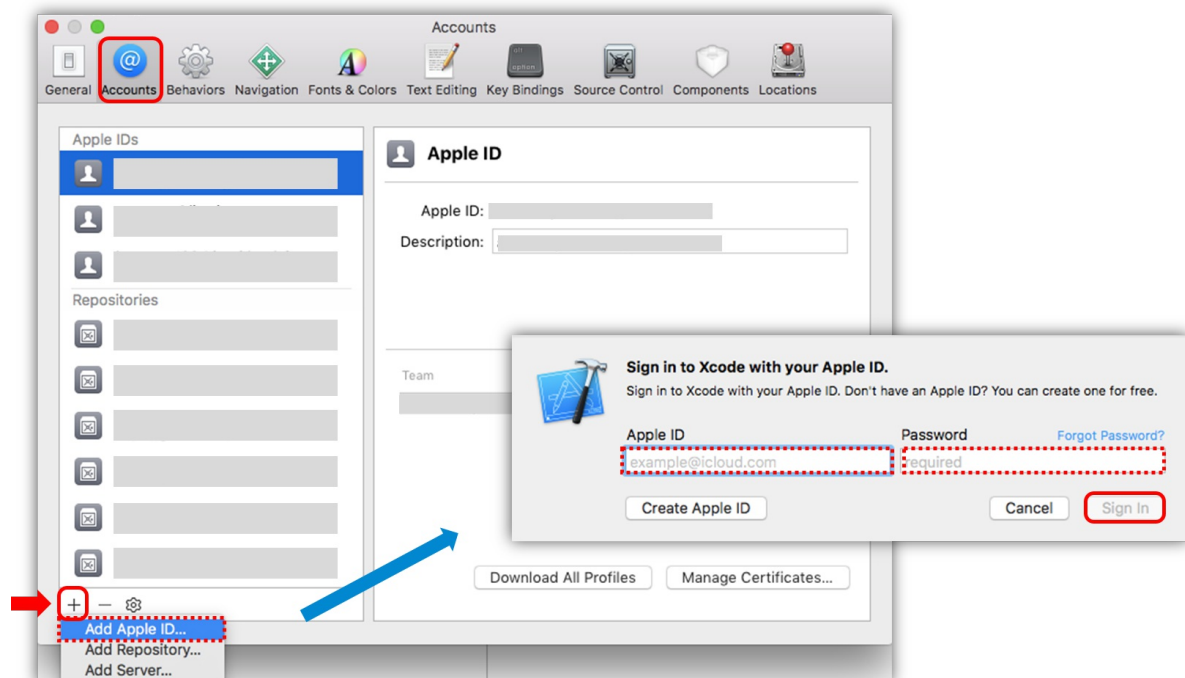
- AppDelegate.swift を編集します
- 先程ニフティクラウド mobile backend のダッシュボード上で確認したAPIキーを貼り付けます



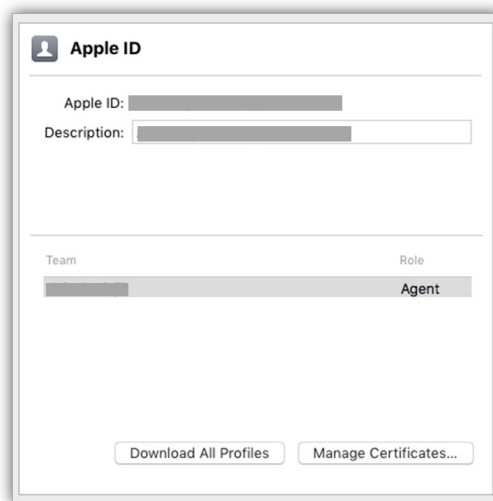
- それぞれ `YOUR_NCMB_APPLICATION_KEY` と `YOUR_NCMB_CLIENT_KEY` の部分を書き換えます
 - このとき、ダブルクォーテーション (") を消さないように注意してください！
- 書き換え終わったら `command + s` キーで保存をします

5. 実機ビルド

- 始めて実機ビルドをする場合は、Xcode に Apple developer アカウント (Apple ID) の登録をします
- メニューバーの「Xcode」 > 「Preferences...」を選択します
- Accounts 画面が開いたら、左下の「+」 > 「Add Apple ID...」をクリックします
- 「Apple ID」と「Password」を要求されるので、入力し「Sign in」をクリックします

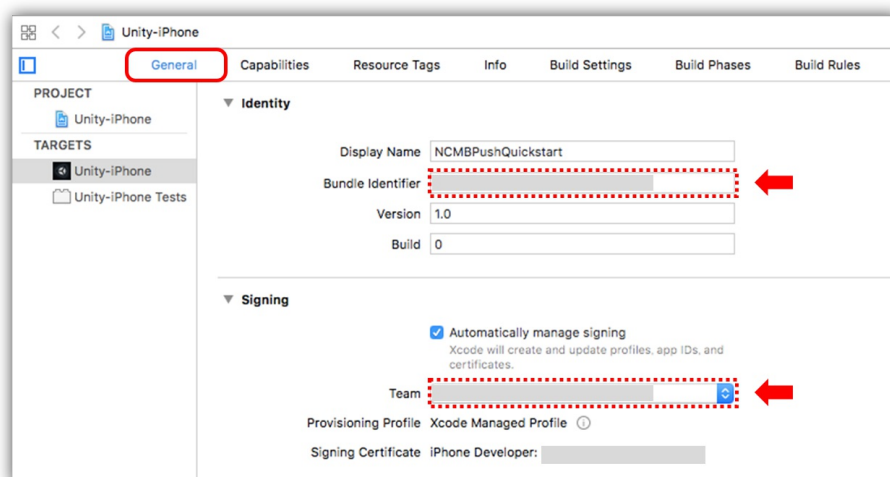


- 登録されると、下図のようになります
 - 追加した情報があっていればOKです



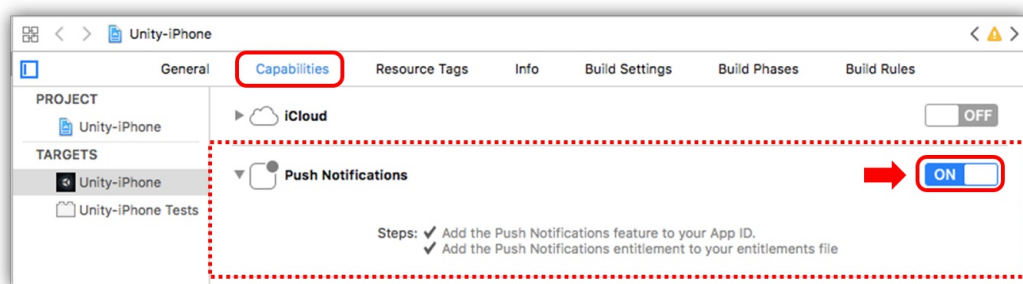
The screenshot shows the 'Apple ID' configuration window. It has a title bar with an Apple ID icon and the text 'Apple ID'. Below the title bar, there are two text input fields: 'Apple ID:' and 'Description:'. Below these fields, there is a table with two columns: 'Team' and 'Role'. The 'Team' column has a dropdown menu, and the 'Role' column has a dropdown menu with 'Agent' selected. At the bottom of the window, there are two buttons: 'Download All Profiles' and 'Manage Certificates...'.

- 確認できたら設定画面を閉じます
- ここからアプリをビルドするための設定を行います
- 「TARGETS」 > 「Unity-iPhone」 > 「General」を開きます
- まず「▼identity」 > 「Bundle Identifier」に Apple Developer Program で AppID 作成時に設定した、**Bundle ID** を入力します
 - 注意：必ず同じ Bundle ID を設定してください！
- 次に「▼Signing」を編集します
- 「Automatically manage signing」にチェックを入れた状態で、「Team」を選択します
 - 今回使用する Apple developer アカウントを選択してください



The screenshot shows the Xcode interface with the 'General' tab selected for the 'Unity-iPhone' target. The 'Identity' section is expanded, showing fields for 'Display Name' (NCMBPushQuickstart), 'Bundle Identifier' (highlighted with a red dashed box and a red arrow), 'Version' (1.0), and 'Build' (0). The 'Signing' section is also expanded, showing the 'Automatically manage signing' checkbox checked, and the 'Team' dropdown menu (highlighted with a red dashed box and a red arrow). Below the 'Team' dropdown, there are fields for 'Provisioning Profile' (Xcode Managed Profile) and 'Signing Certificate' (iPhone Developer:).

- この2点を設定することで自動的に「Provisioning Profile」が読み込まれます
 - プロビジョニングプロファイルはダウンロードしたものを一度**ダブルクリック**して認識させておく必要があります（表示されない場合はダブルクリックを実施してください）
- 最後にプッシュ通知の設定をします
- 「Capabilities」を開き、「Push Notifications」を **ON** に設定します
- 正しく設定が完了すると、以下のように「Steps」にチェックマークが表示されます



- これで準備は完了です

6.動作確認

- 登録した動作確認用 iPhone を lightningケーブルで Mac につなぎます
- Xcode 画面で左上で、接続した iPhone を選び、実行ボタン（さんかくの再生マーク）をクリックすると端末にアプリがインストールされます
- インストールしたアプリを起動します
 - **注意：**プッシュ通知の許可を求めるアラートが出たら、必ず許可してください！

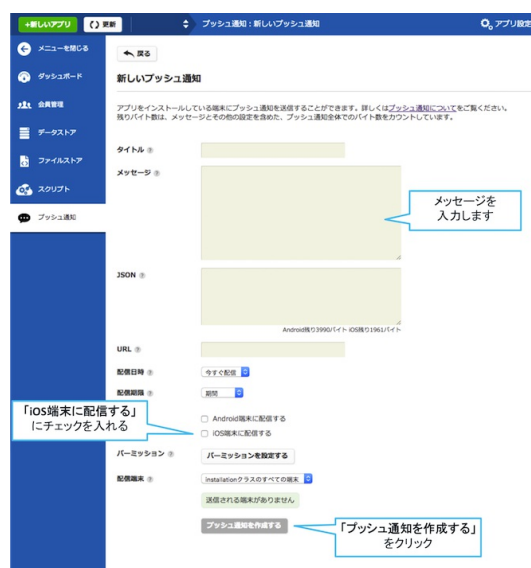
- 起動されたらこの時点で Android端末 は レジストレーションID、iOS端末 は デバイストークン が取得され、ニフティクラウド mobile backend に保存されます
- ニフティクラウド mobile backend のダッシュボードで「データストア」>「installation」クラスを確認してみましょう！



- 端末側で起動したアプリは一度閉じておきます

7. プッシュ通知を送りましょう！

- いよいよです！実際にプッシュ通知を送ってみましょう！
- ニフティクラウド mobile backend のダッシュボードで「プッシュ通知」>「+新しいプッシュ通知」をクリックします
- プッシュ通知のフォームが開かれます
- 必要な項目を入力してプッシュ通知を作成します



- 端末を確認しましょう！

- 少し待つとプッシュ通知が届きます！！！！



解説

サンプルプロジェクトに実装済みの内容のご紹介

SDKのインポートと初期設定

- ニフティクラウド mobile backend の[ドキュメント](#)（[クイックスタート](#)）をSwift版に書き換えたドキュメントをご用意していますので、ご活用ください
 - [SwiftでmBaaSを始めよう！\(<CocoaPods>でuse_frameworks!を有効にした方法\)](#)

ロジック

- AppDelegate.swift の didFinishLaunchingWithOptions メソッド内に、「APNsに対してデバイストークンを要求するコード」を記述しています
 - デバイストークンの要求はiOSのバージョンによってコードが異なるため、場合分けして記述しています

```
// デバイストークンの要求
if #available(iOS 10.0, *){
    /** iOS10以上 **/
    let center = UNUserNotificationCenter.current()
    center.requestAuthorization(options: [.alert, .badge, .sound]) {granted, error in
        if error != nil {
            // エラー時の処理
            return
        }
        if granted {
            // デバイストークンの要求
            UIApplication.shared.registerForRemoteNotifications()
        }
    }
} else {
    /** iOS8以上iOS10未満 **/
    //通知のタイプを設定したsettingを用意
    let setting = UIUserNotificationSettings(types: [.alert, .badge, .sound], categories: nil)
    //通知のタイプを設定
    application.registerUserNotificationSettings(setting)
    //DevoceTokenを要求
    application.registerForRemoteNotifications()
}
```

- デバイストークン取得後、didRegisterForRemoteNotificationsWithDeviceToken メソッドが呼ばれ、取得したデバイストークンをニフティクラウド mobile backend 上に保存しています

```
// デバイストークンが取得されたら呼び出されるメソッド
func application(_ application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
    // 端末情報を扱うNCMBInstallationのインスタンスを作成
    let installation : NCMBInstallation = NCMBInstallation.current()
    // デバイストークンの設定
    installation.setDeviceTokenFrom(deviceToken)
    // 端末情報をデータストアに登録
    installation.saveInBackground {error in
        if error != nil {
            // 端末情報の登録に失敗した時の処理
        } else {
            // 端末情報の登録に成功した時の処理
        }
    }
}
```