AdLocus iOS SDK- Swift整合說明

版本

- 文件版本：1.0.0
- SDK版本： 5.2.5 以上

1. 環境

本文件說明如何將 AdLocus iOS SDK 整合到您的 iOS Swift 專案中，整合前請先行確認

- Swift 開發環境
- iOS Deployment Target iOS 10.0
- 使用套件
    - Firebase/Analytics
    - Firebase/Messaging

2. SDK 整合

2.1 載入 SDK

AppDelegate 初始設定, 請於自行輸入 `pushService?.key = "<AdLocus app key>"` 與 `pushService?.fcmapIkey = "<FCM API KEY>"`

```
import UIKit
import UserNotifications
import Firebase
import AdLocus

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate,
UNUserNotificationCenterDelegate, MessagingDelegate {

    var window: UIWindow?
    var pushService: ALPushService?

    var kGCMMessageIDKey: String { return "gcm.message_id" }

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

        // Firebase Cloud Message
        FirebaseApp.configure()
        Messaging.messaging().delegate = self

        // UserNotifications
        UNUserNotificationCenter.current().delegate = self
        let authOptions: UNAuthorizationOptions = [.alert, .badge, .sound]
```

```
        UNUserNotificationCenter.current().requestAuthorization(options:
authOptions, completionHandler: { _, _ in })
        application.registerForRemoteNotifications()

        // Background Fetch
        if (UIDevice.current.systemVersion as NSString).floatValue >= 7.0
{
            UIApplication.shared.setMinimumBackgroundFetchInterval(1 * 60)
        }

        // AdLocus Push Service
        if pushService == nil {
            pushService = ALPushService.sharedInstance() as? ALPushService
            pushService?.key = "<AdLocus app key>"
            pushService?.fcmapIkey = "<FCM API KEY>"
            let targeting: ALTargeting = ALTargeting()
            pushService?.targeting = targeting
            pushService?.enable = true
        }

        return true
    }

}
```

2.2 推播接收處理

接收到Firebase 的推播訊息時，有兩個部份需要處理：

1. 收到推播時，需要呼叫 SDK 去抓取廣告內容呈現出來, 這個部份是由 `func application(_ application: UIApplication, didReceiveRemoteNotification...` 函數進行處理
2. 廣告呈現後，使用者按下廣告，程式需要呼叫 SDK 完成點擊紀錄, 這個部份是由 `func userNotificationCenter(_ center: UNUserNotificationCenter, didReceive response...`函數進行處理

```
    func application(_ application: UIApplication,
didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
        Messaging.messaging().apnsToken = deviceToken
    }

    func messaging(_ messaging: Messaging, didReceiveRegistrationToken
fcmToken: String?) {
        print("Firebase registration token: \(fcmToken ?? "")")
        pushService?.fcMkey = fcmToken
    }

    // AdLocus Push Service Handler
    func application(_ application: UIApplication,
performFetchWithCompletionHandler completionHandler: @escaping
(UIBackgroundFetchResult) -> Void) {
```

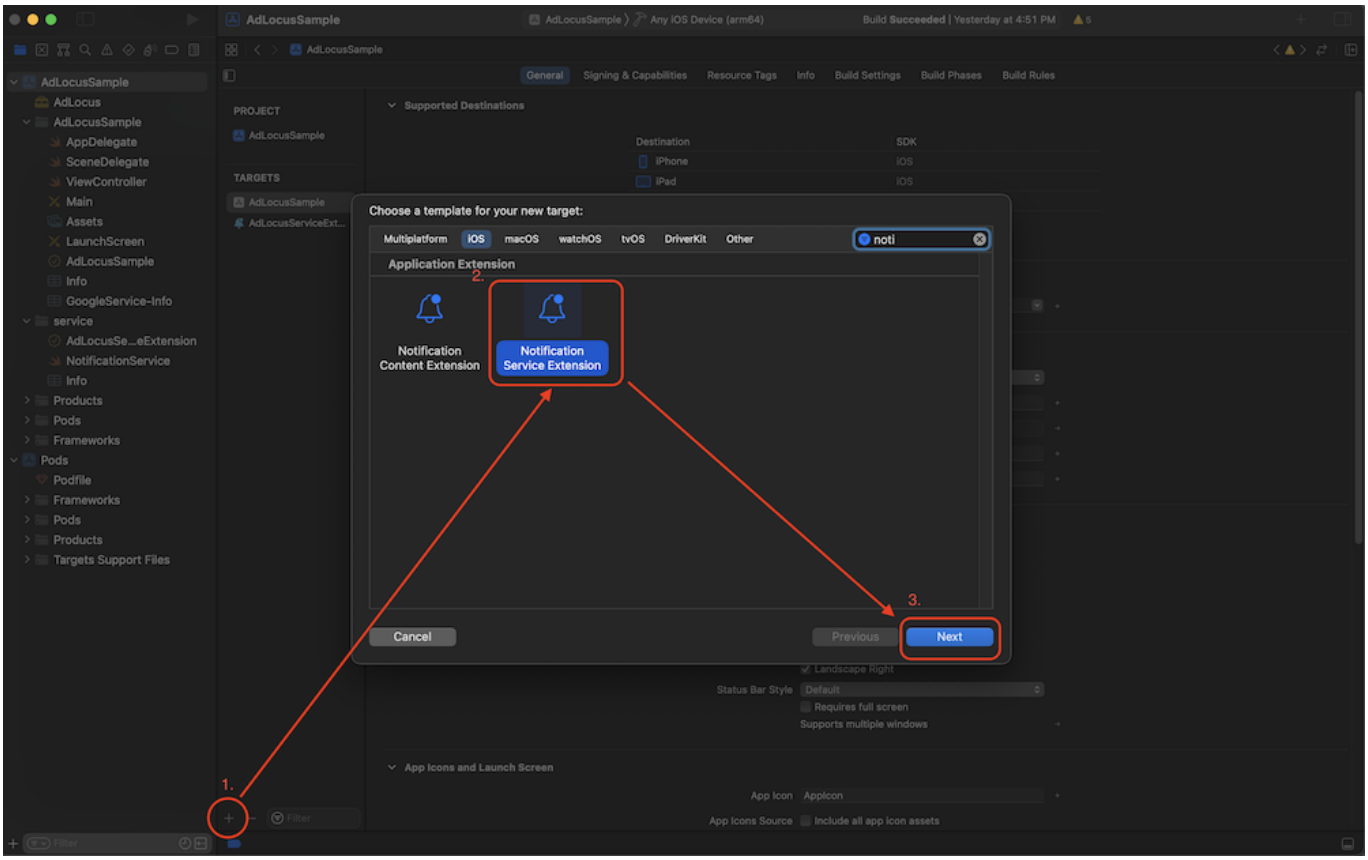
```
        ALPushService.performFetch(completionHandler: completionHandler)
        completionHandler(.newData)
    }

    // LBS Function, 收到推播時，需要呼叫SDK去抓取廣告內容呈現出來
    func application(_ application: UIApplication,
didReceiveRemoteNotification userInfo: [AnyHashable : Any],
fetchCompletionHandler completionHandler: @escaping
(UIBackgroundFetchResult) -> Void) {
        if let messageId = userInfo[kGCMMessageIDKey] {
            print("Message ID: \(messageId)")
        }
        ALPushService.performFetch(completionHandler: completionHandler,
withUserinfo: userInfo)
    }

    func userNotificationCenter(_ center: UNUserNotificationCenter,
willPresent notification: UNNotification, withCompletionHandler
completionHandler: @escaping (UNNotificationPresentationOptions) -> Void)
{
        if let userInfo = notification.request.content.userInfo as?
[String : Any] {
            Messaging.messaging().appDidReceiveMessage(userInfo)
        }
        completionHandler([.badge, .sound, .alert])
    }

    // Click Notification Event, 使用者按下廣告後，呼叫SDK完成點擊紀錄
    func userNotificationCenter(_ center: UNUserNotificationCenter,
didReceive response: UNNotificationResponse, withCompletionHandler
completionHandler: @escaping () -> Void) {
        if let userInfo = response.notification.request.content.userInfo
as? [String : Any], response.notification.request.trigger is
UNPushNotificationTrigger {
            if let url = URL(string: userInfo["ad_link"] as? String ?? "")
{
                DispatchQueue.main.async {
                    UIApplication.shared.open(url)
                }
            }
            completionHandler()
        } else {

ALPushService.didReceiveLocalNotification(response.notification)
            completionHandler()
        }
    }
```

2.3 NotificationService額外處理

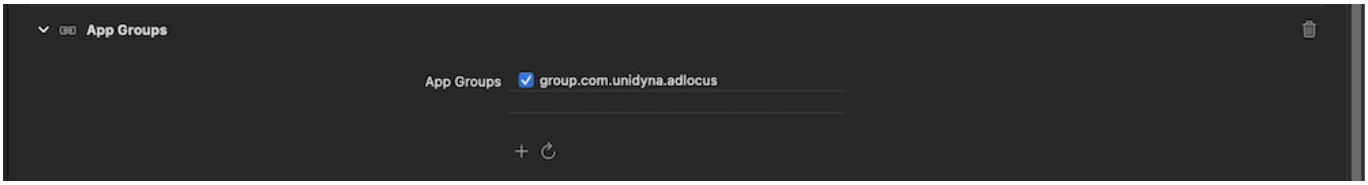
收到推播後，如果推播內容有 `fcm mutable_content: true` 欄位，會需要使推播訊息經由 NotificationService 串接 ADLocus API 轉換為相應推播訊息。該功能需要進行以下修改來達到這個目的

1. 專案新增 NotificationServiceExtension

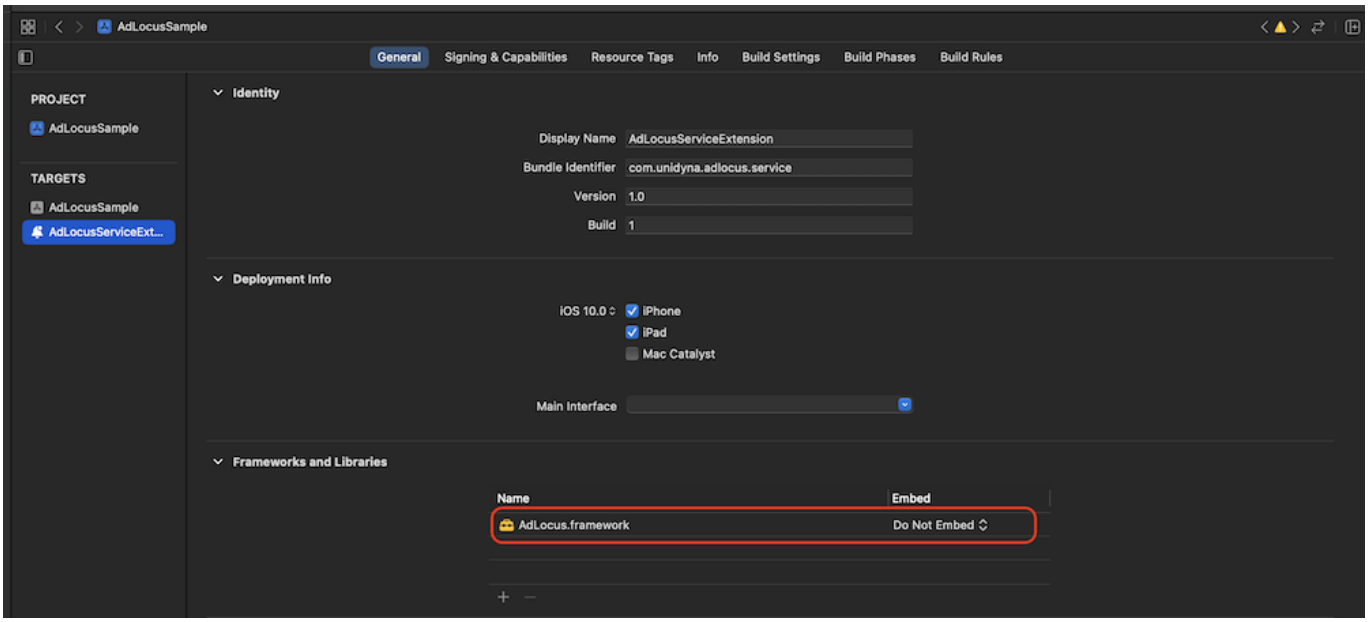
NotificationServiceExtension主要會需要把Adlocus SDK加入，並於接收具有`fcm mutable_content: true` 欄位的推播時, 呼叫Adloucs SDK 函數來取得廣告資訊。



2. 設定NotificationServiceExtension 及 target 的 APP Group (兩個名稱需相同，且需 group. 開頭)



3. 加入 ADLocus SDK Library



4. 程式碼

加入以下程式碼，主要於`func didReceive` 加入呼叫ADLocus SDK

```
import UserNotifications
import AdLocus

class NotificationService: UNNotificationServiceExtension {

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

    override func didReceive(_ request: UNNotificationRequest,
withContentHandler contentHandler: @escaping (UNNotificationContent) ->
Void) {
        self.contentHandler = contentHandler
        bestAttemptContent = (request.content.mutableCopy() as?
UNMutableNotificationContent)

        if let bestAttemptContent = bestAttemptContent {
            // 串接ADLocus取得推播訊息
            ALPushService.didReceiveNotificationExtensionRequest(request,
with: bestAttemptContent, completionHandle: { (attach) in
                if let a = attach {
                    bestAttemptContent.userInfo = a
                    bestAttemptContent.body = a["ad_body"] as? String ??
""

                    if let trackImp = (a["imps_tracking_url"] as? [String]
?? []).first, trackImp != "" {
                        print("tracImp URL: \(trackImp)")
                        self.sendDcm(with: trackImp)
                    }
                }
                contentHandler(bestAttemptContent)
```

```
            })
        }
    }

    override func serviceExtensionTimeWillExpire() {
        if let contentHandler = contentHandler, let bestAttemptContent =
bestAttemptContent {
            contentHandler(bestAttemptContent)
        }
    }

    @objc func sendDcm(with url: String) {
        guard let u = URL(string: url) else { return }
        URLSession.shared.dataTask(with: URLRequest(url: u),
completionHandler: { (data, response, error) in
            var canRetryNewImp: Bool = true
            if error != nil && canRetryNewImp {
                canRetryNewImp = false
                self.perform(#selector(self.sendDcm), with: url,
afterDelay: 60)
            }
        }).resume()
    }
}
```