Apple notification programming guide

uweiyuan

# 概览

客户端开发主要使用的框架是UserNotification UserNotificationUI框架，UIApplication类，推送类型有Remote和Local。详细说明了客户端开发推送涉及到的各个方面。

服务端开发详细地说明了构建一个中间媒介Provider，在Device 和 APNs之间完成串联的全部逻辑。

# 客户端开发

iOS 8 ，我们可以给推送增加用户操作，这样使推送更加具有交互性，并且允许用户去处理用户推送更加的迅速。

iOS 9 ，苹果又再次增加了快速回复功能，进一步的提高了通知的响应性。开发者可以允许用户通过点击推送，并用文字进行回复。

iOS 10，支持3D Touch的设备上，可以按压推送，推送就会被展开，展示出更加详细的用户界面。展示出来的详细界面对用户来说，提供了更加有用的信息。用户可以通过点击下面的按钮，来处理一些事件，并且推送的详细界面也会跟着用户的操作进行更新UI界面。

## Remote Notification

类型：显式和静默

样式：alert(文字)、sound(铃声)、badge(角标)

行为：Action，针对通知消息，可以定义用户的操作(可以打开App，也可以不打开App)

分类：Category，针对富媒体，Action的载体

限制：

Category一次可以注册1or n个，category关联的action，最多可以有4个，但是要根据notification被展示的方式显示个数，banner就只能展示2个

自定义铃声：Linear PCM、MA4(IMA/ADPCM)、alaw，μLaw；将声频文件放到bundle中；要求30s以下，否则就是系统默认的铃声。

可以使用afconvert工具转换格式

例如：afconvert ~/submarine.aiff ~/Desktop/sub.caf -d ima4 -f caff -v

### 权限认证

iOS10+，苹果收敛了推送的权限，在使用推送之前，现需要进行权限的验证，这个部分会有系统UI与用户进行交互。在准备使用推送之前需要调用权限认证接口询问用户的意愿。具体调用：

UNUserNotificationCenter\* center = [UNUserNotificationCenter currentNotificationCenter];

[center requestAuthorizationWithOptions:(UNAuthorizationOptionAlert + UNAuthorizationOptionSound) ompletionHandler:^(BOOL granted, NSError \* \_Nullable error) {

// Enable or disable features based on authorization.

}];

在用户做出认证之后，系统会保存用户的选择，之后再调用此方法是不会有系统UI的提示。

### 注册通知

注册方法： UIApplication实例的registerForRemoteNotification

注册方法回调：

- (void)application:(UIApplication \*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*)deviceToken{}

- (void)application:(UIApplication \*)application didFailToRegisterForRemoteNotificationsWithError:(NSError \*)error{}

一旦App注册成功，即获取Device Token之后，只有当Device Token发生变化，App才会和APNs建立连接。

如果设备没有联网，以上两个方法都不会被调用。

连接到APNs是在指定的port，有些网络可能会限制了这个端口。

### 接收消息

接收消息是iOS系统自行处理的，在底层有两个系统级的service进程处理（apsd(apple push service daemon)和springboard），在接收到指定bundle id的消息之后，由这个两个进行进行检查和转发。

不同iOS版本，接收消息的方法也不同。

APNs推送消息的时候，如果手机关机了，APNs会保留这个推送，稍晚会再次尝试。

#### 显式

iOS 10，App前台运行时，接收展现消息，

- (void)userNotificationCenter:(UNUserNotificationCenter \*)center willPresentNotification:(UNNotification \*)notification withCompletionHandler:(void (^)(UNNotificationPresentationOptions))completionHandler

iOS 10之前，App运行态下：

- (void)application:(UIApplication \*)application didReceiveRemoteNotification:(NSDictionary \*)userInfo{}

#### 静默

- (void)application:(UIApplication \*)application didReceiveRemoteNotification:(NSDictionary \*)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler

iOS7+，需要在Xcode工程中【Capabilities】->【background mode】->【remote notification】

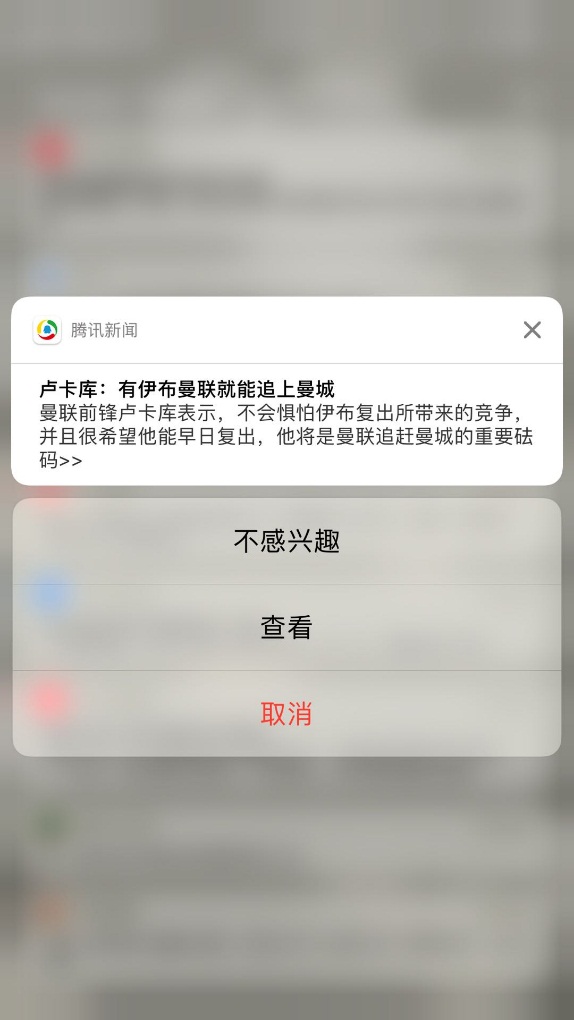
静默推送，系统会在后台唤醒App，执行短时任务。

#### iWatch

运行在watchOS 上的app，不需要显示地注册远程通知，取而代之的是，在watch显示的通知是这些App依赖于它们镜像的iPhone转发通知。当iPhone锁屏或者是屏幕休眠，iWatch在用户的手腕上且没有锁屏的时候，远程推送才会被转发。

### 消息行为

iOS8+，对于每一条消息，开发者都可以预定义用户对消息的操作，行为属性决定是否打开App，具体如下：



#### 自定义Action

由于Action的载体是Category，所以，自定义消息行为的步骤是：

1.创建Action对象,UNNotificationAction(iOS 10+), UIUserNotificationAction(iOS 8+)

2.创建Category对象， UNNotificationCategory， UIUserNotificationCategory

3.设置通知

iOS 10+， UNUserNotificationCenter实例对象的 setNotificationCategories；

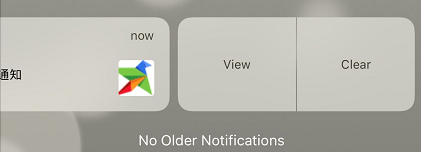
iOS 8+，UIUserNotificationSettings封装Categories，再调用UIApplication 实例对象的registerUserNotificationSettings

Action只在iOS和watchOS中被支持

#### 系统标准Action

DefaultActionIdentifier这个Action是系统默认的，表明用户点击通知中心，打开App的时候，并没有选择任何自定义的Action。

DismissActionIdentifier，这个Action是系统默认的，表明用户没有打开App，是手动地清除了Notification interface(点击clear或者是x)，这个要求在配置Category类型的时候，要指定为CustomDismiss选项。





#### 捕捉Action

当用户在通知栏中点开了消息，会向用户展现之前定义的Action，如上图。根据定义Action的属性不同，以及开发的功能不同，捕获Action的方式有以下两种：

##### Content中捕捉Action

当actions传递给content extension时，它可以延迟推送通知的消失时间。在这段延迟的时间之内，可以处理用户点击按钮的事件，并且更新UI，一切都处理完成之后，我们再去让推送通知消失掉，具体参照：

- (void)didReceiveNotificationResponse:(UNNotificationResponse \*) response completionHandler:(void (^)(UNNotificationContentExtensionResponseOption))completion {

if ([response.actionIdentifier isEqualToString:@"xgaction001"]) {

self.label.text = @"点击了1";

}

dispatch\_after(dispatch\_time(DISPATCH\_TIME\_NOW, (int64\_t)(2 \* NSEC\_PER\_SEC)), dispatch\_get\_main\_queue(), ^{

// 将Action转发给App处理

completion(UNNotificationContentExtensionResponseOptionDismissAndForwardAction);

});

}

##### App中捕捉Action

在App中捕捉Action，要求Content Extension将Action转发给App，且Action属性是打开App的，借助UNUserNotificationDelegate的回调方法就可以捕捉到用户的行为，具体参照：

- (void)userNotificationCenter:(UNUserNotificationCenter \*)center didReceiveNotificationResponse:(UNNotificationResponse \*)response withCompletionHandler:(void (^)(void))completionHandler {

if ([response.actionIdentifier isEqualToString:@"xgaction001"]) {

NSLog(@"click from Action1");

}

completionHandler();

}

### 修改消息

在iOS10+，可以在接收到消息之后展示之前，根据需要对消息的内容进行定制化的修改。为了达到此目的，主要借助Notification Service Extension实现。

主要步骤：

1. 为App新增Target，选择Notification Service Extension
2. 需要使用开发者账号，为新的Target配置开发和发布环境，包括：证书和App信息
3. 在系统生成的UNNotificationServeiceExtension子类中处理APNs推送到客户端的Raw数据。

具体实现代码：

-(void)didReceiveNotificationRequest:(UNNotificationRequest\*)request withContentHandler:(void (^)(UNNotificationContent \* \_Nonnull))contentHandler {

self.contentHandler = contentHandler;

self.bestAttemptContent = [request.content mutableCopy];

self.bestAttemptContent.body = @"custom body";

self.bestAttemptContent.title = @"custom title";

self.bestAttemptContent.badge = @(2);

self.contentHandler(self.bestAttemptContent);

}

以上方法被调用的必要条件：

1. 配置远程通知展示提醒(alert in payload)
2. 远程通知中aps字典必须包含mutable-content：1键值对

不能修改静默推送，不能修改之后sound & badge 的推送

补充：

如果推送中不包含任何提醒(alert)文字，系统会忽略此处的修改，分发原始消息

### 富媒体消息

在iOS10+，可以支持富媒体消息的推送，系统封装是UNNotificationAttachment，每一个App存储Attachment的空间是有限的，不用之后要清除。

推送里面包含的attachment这些文件，是由系统管理的，系统会把这些文件放在单独的一个地方，不在应用的沙盒中，然后统一管理。

富媒体支持的文件类型：

| **Attachment** | **Supported File Types** | **Maximum Size** |
| --- | --- | --- |
| Audio | kUTTypeAudioInterchangeFileFormat  kUTTypeWaveformAudio  kUTTypeMP3  kUTTypeMPEG4Audio | 5 MB |
| Image | kUTTypeJPEG  kUTTypeGIF  kUTTypePNG | 10 MB |
| Movie | kUTTypeMPEG  kUTTypeMPEG2Video  kUTTypeMPEG4  kUTTypeAVIMovie | 50 MB |

#### 接收

同样是借助Notification Service Extension来实现的。具体方法步骤：

1. 从UNNotificationRequest中获取富媒体链接URL
2. 下载URL资源文件，保存在本地
3. 使用资源的本地路径创建UNNotificationAttachment对象。
4. 将UNNotificationAttachment对象数组添加到UNMutableNotificationContent对象中。

- (void)didReceiveNotificationRequest:(UNNotificationRequest \*)request withContentHandler:(void (^)(UNNotificationContent \* \_Nonnull))contentHandler {

self.contentHandler = contentHandler;

self.bestAttemptContent = [request.content mutableCopy];

//1. 下载

NSURL \*url = [NSURL URLWithString:@"http://xg.qq.com/pigeon\_v2/resource/imgcache/image/logo.png"];

NSURLSessionConfiguration \*config = [NSURLSessionConfiguration defaultSessionConfiguration];

NSURLSession \*session = [NSURLSession sessionWithConfiguration:config];

NSURLSessionDataTask \*task = [session dataTaskWithURL:url completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

if (!error) {

//2. 保存数据, 不可以存储到不存在的路径

NSString \*path = [NSSearchPathForDirectoriesInDomains(NSLibraryDirectory, NSUserDomainMask, YES).firstObject

stringByAppendingPathComponent:@"logo.png"];

UIImage \*image = [UIImage imageWithData:data];

CGRect subrect = CGRectMake(0, 0, 55, CGImageGetHeight(image.CGImage));

CGImageRef subImage = CGImageCreateWithImageInRect(image.CGImage, subrect);

NSError \*err = nil;

[UIImageJPEGRepresentation([UIImage imageWithCGImage:subImage], 1) writeToFile:path options:NSAtomicWrite error:&err];

//3. 添加附件

UNNotificationAttachment \*attachment = [UNNotificationAttachment attachmentWithIdentifier:@"remote-atta1" URL:[NSURL fileURLWithPath:path] options:nil error:&err];

if (attachment) {

self.bestAttemptContent.attachments = @[attachment];

}

}

self.contentHandler(self.bestAttemptContent);

}];

[task resume];

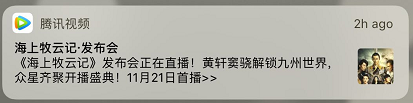
}

补充：

Notification Service Extension只有30s限制,在上述方法中执行的任务都需要尽量完成，如果没有完成，会回调serviceExtensionTimeWillExpire:，这是最后修改的机会。

#### 展示

可以在通知中心为富媒体的展示定制开发UI，借助Notification Content Extension。效果：





主要步骤：

1. 为App新增Target，选择Notification Content Extension
2. 需要使用开发者账号，为新的Target配置开发和发布环境，包括：证书和App信息
3. 在Notification Serveice Extension中处理APNs推送到客户端的Raw数据。将富媒体文件封装到UNNotificationAttachment对象中，将附件添加到通知消息中，然后转发给Content Extension.具体实现：

- (void)didReceiveNotificationRequest:(UNNotificationRequest \*)request withContentHandler:(void (^)(UNNotificationContent \* \_Nonnull))contentHandler {

self.contentHandler = contentHandler;

self.bestAttemptContent = [request.content mutableCopy];

//1. 下载

NSURL \*url = [NSURL URLWithString:@"http://xg.qq.com/pigeon\_v2/resource/imgcache/image/logo.png"];

NSURLSessionConfiguration \*config = [NSURLSessionConfiguration defaultSessionConfiguration];

NSURLSession \*session = [NSURLSession sessionWithConfiguration:config];

NSURLSessionDataTask \*task = [session dataTaskWithURL:url completionHandler:^(NSData \* \_Nullable data, NSURLResponse \* \_Nullable response, NSError \* \_Nullable error) {

if (!error) {

//2. 保存数据, 不可以存储到不存在的路径

NSString \*path = [NSSearchPathForDirectoriesInDomains(NSLibraryDirectory, NSUserDomainMask, YES).firstObject stringByAppendingPathComponent:@"logo.png"];

UIImage \*image = [UIImage imageWithData:data];

CGRect subrect = CGRectMake(0, 0, 55, CGImageGetHeight(image.CGImage));

CGImageRef subImage = CGImageCreateWithImageInRect(image.CGImage, subrect);

NSError \*err = nil;

[UIImageJPEGRepresentation([UIImage imageWithCGImage:subImage], 1) writeToFile:path options:NSAtomicWrite error:&err];

//3. 添加附件

UNNotificationAttachment \*attachment = [UNNotificationAttachment attachmentWithIdentifier:@"remote-atta1" URL:[NSURL fileURLWithPath:path] options:nil error:&err];

if (attachment) {

self.bestAttemptContent.attachments = @[attachment];

}

}

self.contentHandler(self.bestAttemptContent);

}];

[task resume];

}

1. 在Notification Content Extension创建的遵循UNNotificationContentExtension协议的UIViewController子类中定制好UI，在协议的回调方法中接收上一步转发过来的UNNotification对象，实现自定义的展示。具体代码如下：

- (void)didReceiveNotification:(UNNotification \*)notification {

UNNotificationAttachment \*att = notification.request.content.attachments[0];

if ([att.URL startAccessingSecurityScopedResource]) { //通知系统开始使用附件

//访问通知对象中的富媒体附件,并定制UI

[att.URL stopAccessingSecurityScopedResource]; // 完毕之后通知系统

}

}

1. 在info.plist文件中增加UNNotificationExtensionCategory key，value设置为Category的identifier，且是App启动时注册的

补充1：

如果要是支持多个category，需要配置content extension的Info.plist文件，展开NSExtension的NSExtensionAttributes,将UNNotificationExtensionCategory的类型修改为Array ，然后添加notification Category的实例，在bundle中可以添加多个UNNotificationContentExtension实例，但是对于给定的Category，系统希望只有一个extension支持它，所以必须为UNNotificationExtensionCategory配置不同的值集合

补充2：

如果你想把默认文本信息部分隐藏掉，只需要在Notification Content 的info.plist中添加一个键值UNNotificationExtensionDefaultContentHidden设置为YES就可以了

补充3：

不用storyboard来自定义界面，只需要在Notifications Content 的info.plist中把NSExtensionMainStoryboard替换为NSExtensionPrincipalClass，并且value对应你的类名

## Local Notification

本地通知和远程通知看起来是一样的，实现的方式有所不同，其他无太大区别。本地通知最显著的特点就是Scheduled。

除了macOS之外，其他平台都支持。

### 配置本地通知

在不同的版本中，需要不同的操作，基本的步骤都差不多：

iOS 10 +：

1. 创建和配置Notification，使用UNMutableNotification类
2. 创建触发器，不管是UNCalendarNotificationTrigger, UNTimeIntervalNotificationTrigger, or UNLocationNotificationTrigger，用来描述Notification被分发的条件(时间、地理位置)
3. 创建请求，这个对象需要前两部中创建的Notification对象和Trigger对象

在创建请求的时候，给请求一个identifier，之后可以使用这个标识来查询准备中的通知或者是在通知被分发之前取消掉。

iOS 10 earlier，使用UILocalNotification来创建本地通知。不支持Action，不支持Category，有自己比较独立的作用域。

具体使用：

UILocalNotification \*notification = [[UILocalNotification alloc] init];

if (notification) {

// 设置时间

notification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10];

notification.timeZone = [NSTimeZone defaultTimeZone];

notification.alertTitle = @"AlertTitle 1";

notification.alertBody = @"AlertBody 1";

NSDictionary \*infoDict = [NSDictionary dictionaryWithObjectsAndKeys:@"notification\_1", @"id", nil];

notification.userInfo = infoDict;

[[UIApplication sharedApplication] scheduleLocalNotification:notification];

}

// APP运行中收到local notification时调用

- (void)application:(UIApplication \*)application didReceiveLocalNotification:(UILocalNotification \*)notification {

// 可根据notification对象的userInfo等属性进行相应判断和处理

}

其他具体操作的定义都在UIApplication类中。

### 消息行为

只需要将消息的Category identifier赋值给UNMutableNotificationContent对象的属性categoryIdentifier，具体的创建步骤参照Remote Notification中的消息行为即可。

### 消息铃声

消息铃声可以使用系统默认的，或者是自定义的，铃声被抽象为UNNotificationSound类。设置铃声就是将UNNotificationSound对象赋值给UNMutableNotificationContent对象的属性sound。

自定义的铃声要求铃声media file必须App本地。

放置本地铃声的途径有两种：

1.打包的时候，将铃声文件放在main bundle中；

2.将铃声下载到App的Container目录中的Library/Sounds子目录下。

### Schedule Notification

安排Notification请求到系统上，通过使用UNUserNotificationCenter对象的addNotificationRequest:withCompletionHandler:

Ex:创建一个早上7点钟的提醒

UNMutableNotificationContent\* content = [[UNMutableNotificationContent alloc] init];

content.title = [NSString localizedUserNotificationStringForKey:@"Wake up!" arguments:nil];

content.body = [NSString localizedUserNotificationStringForKey:@"Rise and shine! It's morning time!" arguments:nil];

// Configure the trigger for a 7am wakeup.

NSDateComponents\* date = [[NSDateComponents alloc] init];

date.hour = 7;

date.minute = 0;

UNCalendarNotificationTrigger\* trigger = [UNCalendarNotificationTrigger triggerWithDateMatchingComponents:date repeats:NO];

// Create the request object.

UNNotificationRequest\* request = [UNNotificationRequest requestWithIdentifier:@"MorningAlarm" content:content trigger:trigger];

// Schedule notification to system

[[UNUserNotificationCenter currentCenter] addNotificationRequest:request withCompletionHandler:nil];

安排好的通知一直保持活跃，直到他们被系统unschedule，或者是手动取消。

补充：

Alerts: iOS 7之前, alert只能显示107个字符，多余的就会使用…符号代替，iOS7之后可以显示235个字符，如过超过8行也是会截断显示

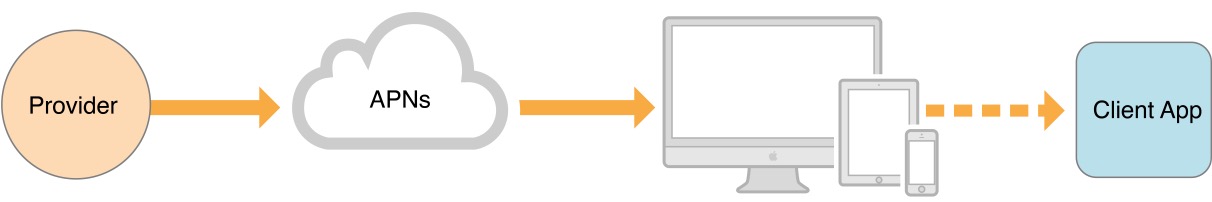
Banners: 62个字符或者是最多2行

Notification Center: 110 个字符或者是4行.

Lock Screen: Same as notification center.

# Provider开发

## 概要



App一旦启动，会建立一个可信加密持久的IP连接到APNs

APNs使用这个长连接分发notification

如果iOS设备未连接网络，APNs会持有通知消息，并在一段时间之后重试

Your Server known as Provider to originate notifications

## Server的任务

1. 从App端接收通过APNs获取的全局唯一的、指定App的device token+有关数据
2. 决定remote notification被发送到device的时机
3. 创建发发送请求到APNs，请求包含notification payload和其他数据，APNs负责分发

对于每一个Remote Notification Request， Provider需要

1. 创建一个包含payload的 JSON字典
2. 添加(payload+deviceToken+其他数据)到HTTP/2的request中(特有的请求格式)
3. 通过一个使用HTTP/2网络协议的持久、安全的通道，发送请求(包含token形式的加密凭证或者是证书)到APNs

## 服务质量(QoS)、存储转发、合并通知

当分发消息的设备处于offline，QoS会保存消息一段时间，当设备状态可用的时候，再次分发。只存储每个设备和每个App的最近的消息。如果设备offline，消息Push Request只会保留最新的，后面的消息会替换掉前面的消息。

为了合并相似的消息，在Request中带上重叠标识(collapse identifier)， 当在HTTP/2请求头中会有apns-collapse-id字段时，APNs会合并这个字段一样的请求。

## 安全架构

APNs强制执行端到端的加密验证和认证：connection trust，device token trust。

Connection trust：(作用于Provider和APNs、APNs和Device之间)

Provider-to-APNs connection trust, 在被认证的Provider和APNs之间建立连接。Provider要自己保证与APNs连接的可信性

APNs-to-device connection trust, 用来保证只有被认证过的设备可以连接到APNs以接收消息。APNs会自动加强与每个设备的连接信任以确保设备的合法性。

Provider与APNs之间的交互必须使用一个有效的authentication key certificate(为了基于token连接信任)或者是SSL certificate(为了基于证书的连接信任)

Device token trust：(作用于每一个端到端的远程通知)

保证消息只在正确的起点（Provider）和终点（Device）被分发

APNs使用connection certificate， CA certificate，cryptographic keys（private+public）来验证Provider和device的connection和identities

APNs使用connection trust + device token trust控制provider和device之间的入口点

Device Token trust ，每一个App拥有唯一的由Apple 分配的token

Provider-to-APNs Connection trust

有两个可用的scheme(方案)用来判定Provider和APNs之间的connection trust

1. Token-based connection trust

一个使用HTTP/2-based API的Provider可以使用JSON web tokens(JWT)来提供有效的凭证来建立APNs之间的连接。Provider提供一个Apple拥有的public key， private key是Provider拥有的， Provider使用private key 创建和签名 JWT authentication tokens，Provider的每一个Push请求都必须包含一个Provider authentication token

1. Certificate-based connection Trust

Provider可以利用一个唯一的Provider certificate + private cryptographic key

Provider certificate(推送证书)，当使用开发者账号为指定App建立push service的时候，Apple就会提供的。它标识一个topic， (topic, 就是bundle ID)

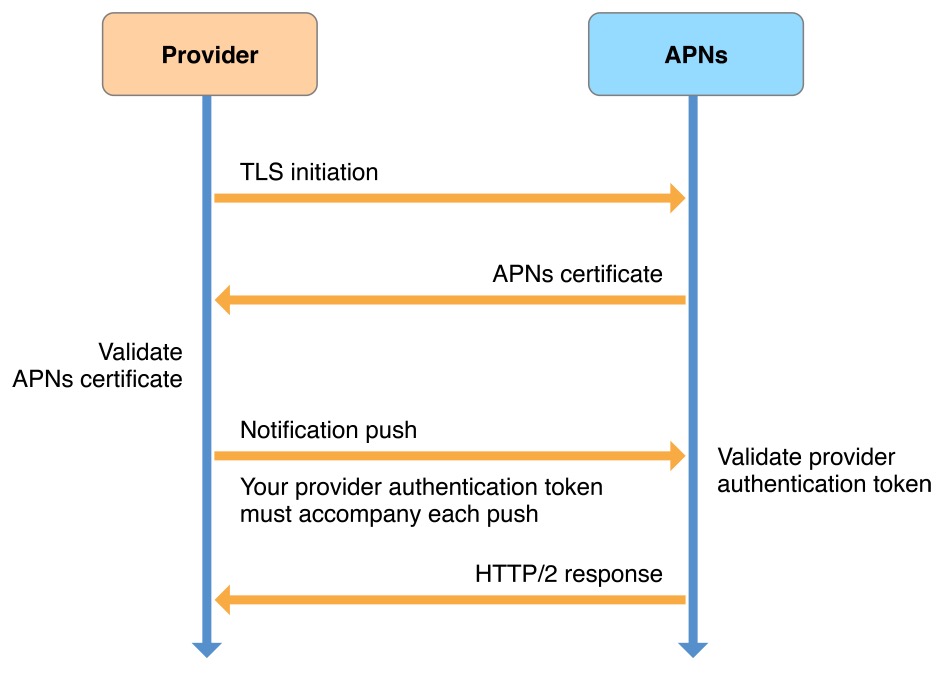
为了建立与APNs基于HTTP/2-based TSL的 session，每一个Provider必须要安装GeoTrust Global CA root证书。macOS默认在keychain中，其他则要自行安装。证书下载地址：[GEOTrust](https://www.geotrust.com/resources/root_certificates/certificates/GeoTrust_Global_CA.pem)

Token-based Provider-to-APNs Trust

Provider使用的证书类型：Apple Push Notification Authentication Key (Sandbox & Production)

证书特性：

* 关联App，适用于App与Watch的连接，还有VoIP
* 通过JWT token-based APNs connection 发送push请求的时候，必须带上authentication token
* APNs authentication key 证书永不过期，但是可以手动revoke

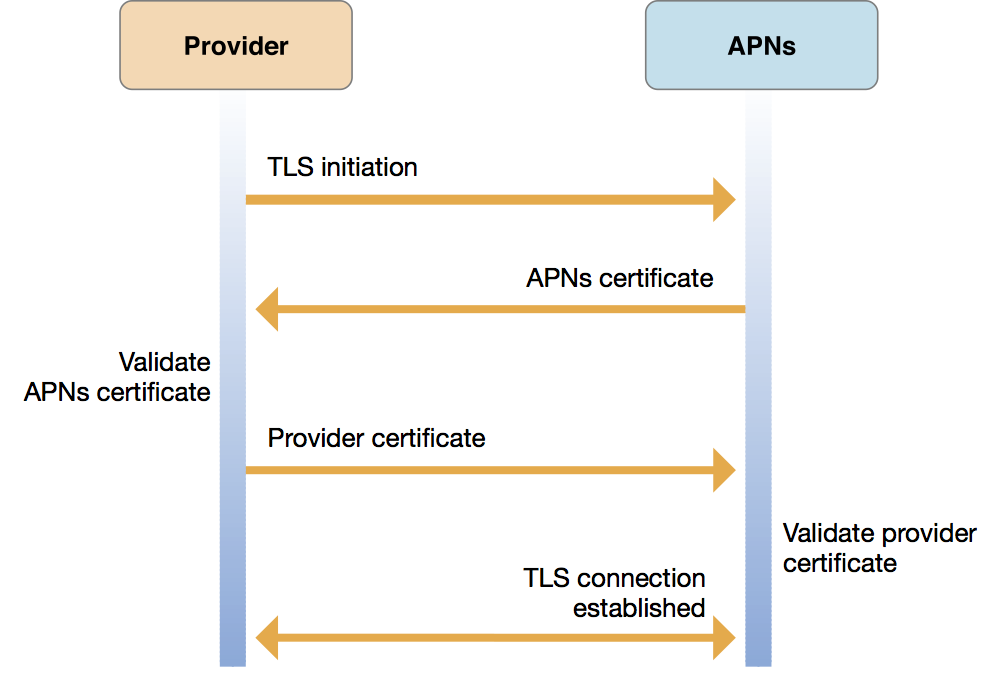


如果证书失效了，第一步的TSL就会失败

Certificate-based Provider-to-APNs Trust

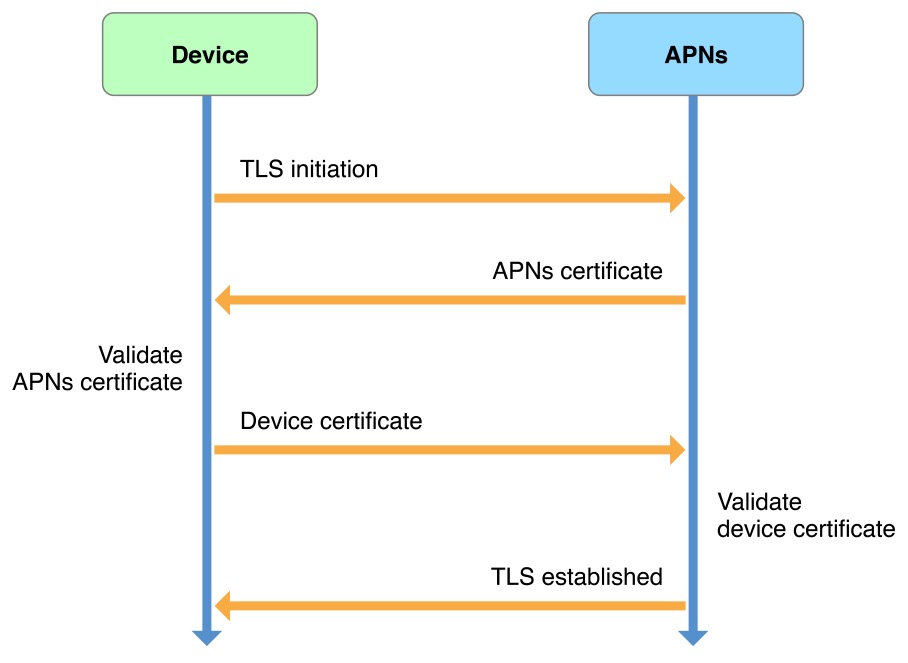
根据开发者创建的推送证书来进行推送，推送证书中要指定bundle ID，依赖开发者配置和提供证书。

这种方式下，APNs维护一个证书吊销列表；如果Provider提供的证书在列表中，APNs revoke Provider的信任(APNs 拒绝TLS initiation连接)

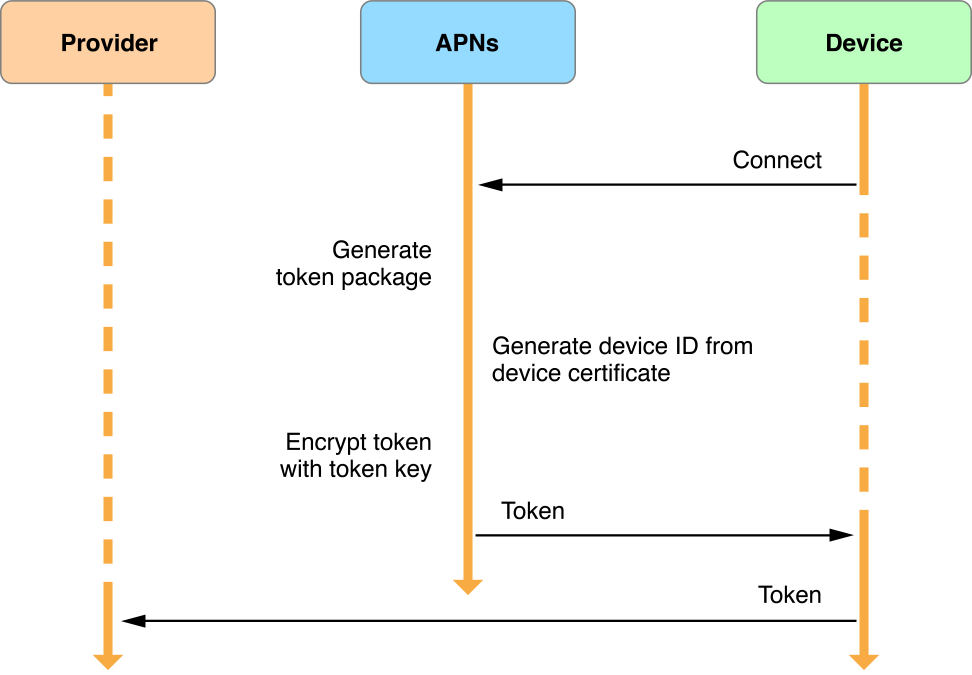


APNs-to-Device Connection Trust and Device Tokens

Device和APNs之间的信任是device’s operating system 在初始化激活device的时候自动建立的(没有App参与的)。每一个device都有一个cryptographic certificate + private cryptographic key，在device被初始化激活的时候获取到的，并被存储到keychain中。在激活的时候，APNs使用certificate + private key 验证device’s connection



通过APNs-Device之间的TSL connection建立，App才可以到APNs注册，并从APNs接收到一个app-specific device token to each app



Token：

是一个不透明的NSData实例对象，包含了一个在指定设备上的指定App的唯一identifier

由Apple 分配

只有APNs可以decode & read token content

系统会为每一个App保存Device Token，直到调用系统注销或者是应用被卸载

Token会失效，APNs发行新token的原因：

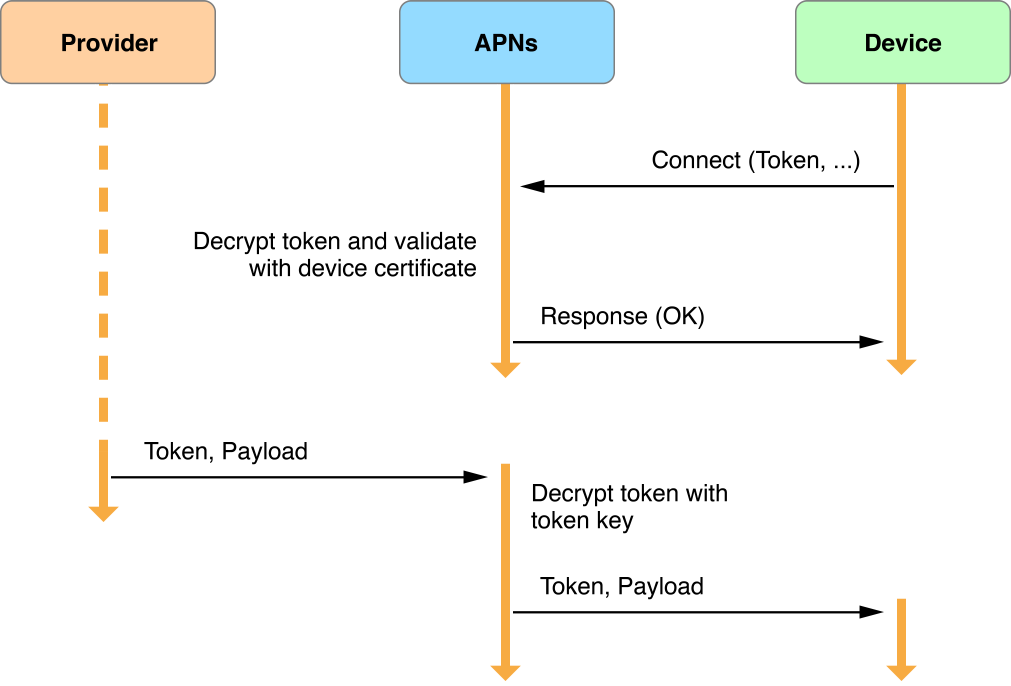
* 用户在新的设备上安装App
* 用户从backup中恢复设备
* 用户重新安装OS
* 其他系统定义的事件(1.调用unregisterNotification接口之后再注册通知，2.清除device data and settings)

为了保护用户的隐私，不要使用token标识设备

*APNs提供的Token是变长的，不要硬编码token的size*

消息的推送路径

App每一次启动，都需要向系统注册remote Notification，一旦注册成功，App就会接收到device token



特点：

Provider向APNs提供的token必须是二进制或者是16进制

APNs使用包含在device token中的device ID判定target device的身份

## 创建Remote Notification Payload

Payload = 自定义数据 + 系统通知用户的风格(Ex：Alert,Badge,Sound)

格式：JSON字典

作为HTTP/2消息的body

限制条件：

常规的remote notification， max size是4kb

VoIP notification，max size是5kb

APNs Binary接口，max size 是2kb

APNS拒绝推送payload size超出限制的notification

APNs不保证Notification分发的安全，不要包含敏感数据，或者是通过其他方式可以检索到的数据

### 创建JSON 字典

Payload数据的JSON 字典中必须包含”aps”字典，aps包含了Apple-defined keys，用于决定系统收到notification怎么样提醒用户

在payload中省去whitespace + line break，从而减小 payload的size

Example 1. The following payload contains an aps dictionary with a simple alert messageThe acme2 key contains an array of app-specific data.

{

"aps" : { "alert" : "Message received from Bob" },

"acme2" : [ "bang", "whiz" ]

}

Example 2. The following payload asks the system to display an alert with a Close button and a single action button. The title and body keys provide the contents of the alert. The “PLAY” string is used to retrieve a localized string from the appropriate Localizable.strings file of the app. The resulting string is used by the alert as the title of an action button. This payload also asks the system to badge the app’s icon with the number 5.

{

"aps" : {

"alert" : {

"title" : "Game Request",

"body" : "Bob wants to play poker",

"action-loc-key" : "PLAY"

},

"badge" : 5

},

"acme1" : "bar",

"acme2" : [ "bang", "whiz" ]

}

Example 3. The following payload specifies that the device should display an alert message, plays a sound, and badges the app’s icon.

{

"aps" : {

"alert" : "You got your emails.",

"badge" : 9,

"sound" : "bingbong.aiff"

},

"acme1" : "bar",

"acme2" : 42

}

Example 4. The following payload uses the loc-key to specify a localized string in the app’s Localizable.strings file. That string is displayed as the message of the alert. The loc-args contains values to substitute into the string before displaying it. The payload also specifies a custom sound to play with the alert.

{

"aps" : {

"alert" : {

"loc-key" : "GAME\_PLAY\_REQUEST\_FORMAT",

"loc-args" : [ "Jenna", "Frank"]

},

"sound" : "chime.aiff"

},

"acme" : "foo"

}

### Configuring a Silent Notification

Silent notification可以唤醒app一段时间**(30s)**，可以让app refresh data in the background

Silent notification不是保持App在后台唤醒的方式，也不是有更高更新的优先级，APNs把silent notification看成一个低优先级，可能“掐死”全部的静默通知，如果静默通知的总量太大了的话，这个限制是动态的，根据条件改变的，不要尝试每个小时发送更多的通知

Silent Notification需要特殊配置payload：

1. aps字典中要包含content-available key，value 是1
2. aps字典中必须不能包含”alert”，”sound”，“badge” keys

回调方法：application:didReceiveRemoteNotification:fetchCompletionHandler:

可以完成：下载操作去获取数据

需要配置：在工程中打开background mode，选择Remote Notification

Listing shows an example of a JSON payload for a silent notification. Configuring a silent notification

{

"aps" : {

"content-available" : 1

},

"acme1" : 42

}

### Assigning custom Actions to remote notification

要求必须在aps字典中带有category key(iOS 8 or later)

Listing Including a category in the payload  
{  
   "aps" : {

      "category" : "NEW\_MESSAGE\_CATEGORY"  
      "alert" : {  
         "body" : "Acme message received from Johnny Appleseed",  
      },  
      "badge" : 3,  
      "sound" : “chime.aiff"  
   },  
   "acme-account" : "jane.appleseed@apple.com",  
   "acme-message" : "message123456"  
}

### Localizing the content of remote notification

1. Provider server 提供localized content //可以指定任意的文本，但是需要发现和跟踪设备当前的语言(app使用NSLocale的preferredLanguages获取，然后传给Provider server，当用户改变语言的时候，app可以收到NSCurrentLocaleDidChangeNotification)，潜在的动态翻译内容

**Getting the current supported language and sending it to the provider  
NSString \*preferredLang = [[NSLocale preferredLanguages] objectAtIndex:0];  
const char \*langStr = [preferredLang UTF8String];  
[self sendProviderCurrentLanguage:langStr]; // custom method**

1. App bundle 存有localized message string //只能bundle中指定语言的已有的文本，这个可以在Localizable.strings资源文件中尽可能的添加多国语言

推送参数化，配合本地国际化资源文件的用法：

本地资源文件中key pair是"REQUEST\_FORMAT" = "%@ and %@ have invited you to play Monopoly";

{

"aps" : {

"alert" : {

"loc-key" : "GAMEEST\_FORMAT",

"loc-args" : [ "Jenna", "Frank"]

}

}

}

最终显示结果是：Jenna and Frank have invited you to play Monopoly

## Communicating with APNs

APNs server 提供的接口可以推送remote notification 到iOS，tvOS，macOS devices，Apple Watch 是通过iOS

API是基于HTTP/2协议的POST请求，包含(a JSON Payload + a device token)

**安全连接到APNs有两种方式：**

1. **provider authentication token**
2. **provider certificates**

### **Provider Authentication Tokens**

**Provider API要支持JSON Web Token(JWT)规则，传递statement + metadata（被称为claim(断言)） + notification 到APNs**

**生成签名的JWT的可用库在**<https://jwt.io>

**Provider authentication token是一个JSON对象，header必须包含以下两个key**

**{**

**”alg": "ES256", //加密token的算法名称字符串**

**"kid": "ABC123DEFG" // 10个字符长的id，从开发者账号获取**

**}**

**Claim payload of token必须包含以下两个key:**

**{**

**"iss": "DEF123GHIJ",//issuer 开发者账号team ID 10字符长**

**"iat": 1437179036 // 标识token生成的时间，从Epoch开始的秒数**

**}**

**创建token之后，使用private key 签名token，然后利用P-256 curve的 ECDSA和SHA-256 hash 算法加密token**

APNs只支持ES256算法签名的povider authentication token，不安全的JWT或者是使用其他算法的JWT，APNs都会返回403

**为了安全，token每隔一段时间就需要生成一个新的，新的token更新了issued at claim key，它的值标识了token生成的时间，如果token的时间戳不在最近的一个小时，APNs都会拒绝推送消息，并返回403**

**如果推测provider token签名被损坏 开发者可以revoke掉证书，重新发行key pairs，然后使用新的private key生成新的token。**

**为了安全，关闭revoke-token的connection，在使用新的签名token进行之前 reconnect**

### ****APNs Provider Certificates****

**这个证书是为了连接到APNs的开发和发布环境**

**使用证书中的拓展名(extension)来标识推送通知的topic，**

**Ex：**

**Extension ( 1.2.840.1136 35.100.6.3.6 )**

**Critical NO**

**Data com.yourcompany.yourexampleapp**

**Data app**

**Data com.yourcompany.yourexampleapp.voip**

**Data voip**

**Data com.yourcompany.yourexampleapp.complication**

**Data complication**

### ****APNs Connection****

**Server：**

**Development server:  api.development.push.apple.com:443**

**Production server:  api.push.apple.com:443**

可以使用2197，这个允许APNs通信穿过防火墙但阻塞其他的HTTPs流量

**Provider server必须支持TSL1.2以上的版本**

**如果使用authentication token推送，必须定时更新这个token，因为APNs authentication token的有效期是1个小时。**

**并发链接的数量取决于Provider certificate或者是authentication token，还有server load**

**使用token进行连接APNs的话，只有当发送了一条带有有效的provider authentication token的消息，一个 stream才允许连接。**

**APNs忽略HTTP/2 RPIORITY frame**

#### **Best Practices for Managing Connections**

1. 保持与APNs的连接
2. 不要反复关闭or打开连接
3. 不要为每一个推送请求生成新的provider authentication token
4. 建立与APNs的多连接来提升性能
5. 如果证书过期，或者是签名provider token的私钥被revoke了，关闭当前所有连接，然后打开新的连接
6. 可以使用一个HTTP/2 PING frame来检查连接的健康

#### **Terminating an APNs Connection**

APNs发送GOAWAY frame来终止一个HTTP/2的连接，这个frame包含一个JSON格式的payload，其中有reason字段

正常的请求失败是不会终止一个连接的

#### **APNs Notification API**

##### **HTTP/2**

###### **HTTP/2 Request to APNs**

**HTTP/2 request fields**

|  |  |
| --- | --- |
| **Name** | **Value** |
| **:method** | **POST** |
| **:path** | **/3/device/<device-token>(16进制)** |

**APNs requires the use of HPACK (header compression for HTTP/2), which prevents repeated header keys and values. APNs maintains a small dynamic table for HPACK. To help avoid filling up the APNs HPACK table and necessitating the discarding of table data, encode headers in the following way—especially when sending a large number of streams:**

* **The :path value should be encoded as a literal header field without indexing**
* **The authorization request header, if present, should be encoded as a literal header field without indexing**
* **The appropriate encoding to employ for the apns-id, apns-expiration, and apns-collapse-id request headers differs depending on whether it is part of the initial or a subsequent POST operation, as follows:**
  + **The first time you send these headers, encode them with incremental indexing to allow the header names to be added to the dynamic table**
  + **Subsequent times you send these headers, encode them as literal header fields without indexing**

Encode all other headers as literal header fields with incremental indexing. For specifics on header encoding, see [tools.ietf.org/html/rfc7541#section-6.2.1](http://tools.ietf.org/html/rfc7541#section-6.2.1) and [tools.ietf.org/html/rfc7541#section-6.2.2](http://tools.ietf.org/html/rfc7541#section-6.2.2).

**APNs除了下表中的Header是可以接受的，其他都会忽略**

| Header | Description |
| --- | --- |
| authorization | **The provider token that authorizes APNs to send push notifications for the specified topics. The token is in Base64URL-encoded JWT format, specified as bearer <provider token>.**  **When the provider certificate is used to establish a connection, this request header is ignored.** |
| apns-id | **A canonical UUID that identifies the notification. If there is an error sending the notification, APNs uses this value to identify the notification to your server.**  **The canonical form is 32 lowercase hexadecimal digits, displayed in five groups separated by hyphens in the form 8-4-4-4-12. An example UUID is as follows:**  **123e4567-e89b-12d3-a456-42665544000**  **If you omit this header, a new UUID is created by APNs and returned in the response.** |
| apns-expiration | **A UNIX epoch date expressed in seconds (UTC). This header identifies the date when the notification is no longer valid and can be discarded.**  **If this value is nonzero, APNs stores the notification and tries to deliver it at least once, repeating the attempt as needed if it is unable to deliver the notification the first time. If the value is 0, APNs treats the notification as if it expires immediately and does not store the notification or attempt to redeliver it.** |
| apns-priority | **The priority of the notification. Specify one of the following values:**   * **10–Send the push message immediately. Notifications with this priority must trigger an alert, sound, or badge on the target device. It is an error to use this priority for a push notification that contains only the content-available key.** * **5—Send the push message at a time that takes into account power considerations for the device. Notifications with this priority might be grouped and delivered in bursts. They are throttled, and in some cases are not delivered.**   **If you omit this header, the APNs server sets the priority to 10.** |
| apns-topic | **The topic of the remote notification, which is typically the bundle ID for your app. The certificate you create in your developer account must include the capability for this topic.**  **If your certificate includes multiple topics, you must specify a value for this header.**  **If you omit this request header and your APNs certificate does not specify multiple topics, the APNs server uses the certificate’s Subject as the default topic.**  **If you are using a provider token instead of a certificate, you must specify a value for this request header. The topic you provide should be provisioned for the your team named in your developer account.** |
| apns-collapse-id | **Multiple notifications with the same collapse identifier are displayed to the user as a single notification. The value of this key must not exceed 64 bytes. For more information, see Quality of Service, Store-and-Forward, and Coalesced Notifications.** |
|  | |

**Body content of request就是一个JSON dictionary，可以压缩且不能超过4k(remote notification)对于VoIP，不能超过5k**

###### ****HTTP/2 Response from APNs****

**APNs response headers**

|  |  |
| --- | --- |
| **Header name** | **Value** |
| **apns-id** | **The apns-id value from the request. If no value was included in the request, the server creates a new UUID and returns it in this header** |
| **:status** | **The HTTP status code** |

Status codes for an APNs response

| Status code | Description |
| --- | --- |
| 200 | Success |
| 400 | Bad request |
| 403 | There was an error with the certificate or with the provider authentication token |
| 405 | The request used a bad :method value. Only POST requests are supported. |
| 410 | The device token is no longer active for the topic. |
| 413 | The notification payload was too large. |
| 429 | The server received too many requests for the same device token. |
| 500 | Internal server error |
| 503 | The server is shutting down and unavailable. |

**一个成功的请求，Response的body是空的**

**一个失败的请求，返回一个JSON字典，以下格式：**

|  |  |
| --- | --- |
| Key | **Description** |
| **reason** | **The error indicating the reason for the failure. The error code is specified as a string** |
| **timestamp** | **If the value in the :status header is 410, the value of this key is the last time at which APNs confirmed that the device token was no longer valid for the topic.**  **Stop pushing notifications until the device registers a token with a later timestamp with your provider.** |

**Values for the APNs JSON reason key**

| **Status code** | **Error string** | **Description** |
| --- | --- | --- |
| **400** | **BadCollapseId** | **The collapse identifier exceeds the maximum allowed size** |
| **400** | **BadDeviceToken** | **The specified device token was bad. Verify that the request contains a valid token and that the token matches the environment.** |
| **400** | **BadExpirationDate** | **The apns-expiration value is bad.** |
| **400** | **BadMessageId** | **The apns-id value is bad.** |
| **400** | **BadPriority** | **The apns-priority value is bad.** |
| **400** | **BadTopic** | **The apns-topic was invalid.** |
| **400** | **DeviceTokenNotForTopic** | **The device token does not match the specified topic.** |
| **400** | **DuplicateHeaders** | **One or more headers were repeated.** |
| **400** | **IdleTimeout** | **Idle time out.** |
| **400** | **MissingDeviceToken** | **The device token is not specified in the request :path. Verify that the :path header contains the device token.** |
| **400** | **MissingTopic** | **The apns-topic header of the request was not specified and was required. The apns-topic header is mandatory when the client is connected using a certificate that supports multiple topics.** |
| **400** | **PayloadEmpty** | **The message payload was empty.** |
| **400** | **TopicDisallowed** | **Pushing to this topic is not allowed.** |
| **403** | **BadCertificate** | **The certificate was bad.** |
| **403** | **BadCertificateEnvironment** | **The client certificate was for the wrong environment.** |
| **403** | **ExpiredProviderToken** | **The provider token is stale and a new token should be generated.** |
| **403** | **Forbidden** | **The specified action is not allowed.** |
| **403** | **InvalidProviderToken** | **The provider token is not valid or the token signature could not be verified.** |
| **403** | **MissingProviderToken** | **No provider certificate was used to connect to APNs and Authorization header was missing or no provider token was specified.** |
| **404** | **BadPath** | **The request contained a bad :path value.** |
| **405** | **MethodNotAllowed** | **The specified :method was not POST.** |
| **410** | **Unregistered** | **The device token is inactive for the specified topic.**  **Expected HTTP/2 status code is 410; see**[**Table 8-4**](https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CommunicatingwithAPNs.html#//apple_ref/doc/uid/TP40008194-CH11-SW15)**.** |
| **413** | **PayloadTooLarge** | **The message payload was too large. See**[**Creating the Remote Notification Payload**](https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/CreatingtheNotificationPayload.html#//apple_ref/doc/uid/TP40008194-CH10-SW1)**for details on maximum payload size.** |
| **429** | **TooManyProviderTokenUpdates** | **The provider token is being updated too often.** |
| **429** | **TooManyRequests** | **Too many requests were made consecutively to the same device token.** |
| **500** | **InternalServerError** | **An internal server error occurred.** |
| **503** | **ServiceUnavailable** | **The service is unavailable.** |
| **503** | **Shutdown** | **The server is shutting down.** |

###### ****HTTP/2 Request/Response Examples for APNs****

****Sample request for a certificate with a single topic****

**HEADERS**

**- END\_STREAM**

**+ END\_HEADERS**

**:method = POST**

**:scheme = https**

**:path = /3/device/00fc13adff785122b4ad28809a3420982341241421348097878e577c991de8f0**

**host = api.development.push.apple.com**

**apns-id = eabeae54-14a8-11e5-b60b-1697f925ec7b**

**apns-expiration = 0**

**apns-priority = 10**

**DATA**

**+ END\_STREAM**

**{ "aps" : { "alert" : "Hello" } }**

****Sample request for a provider authentication token****

**HEADERS**

**- END\_STREAM**

**+ END\_HEADERS**

**:method = POST**

**:scheme = https**

**:path = /3/device/00fc13adff785122b4ad28809a3420982341241421348097878e577c991de8f0**

**host = api.development.push.apple.com**

**authorization = bearer eyAia2lkIjogIjhZTDNHM1JSWDciIH0.eyAiaXNzIjogIkM4Nk5WOUpYM0QiLCAiaWF0I**

**jogIjE0NTkxNDM1ODA2NTAiIH0.MEYCIQDzqyahmH1rz1s-LFNkylXEa2lZ\_aOCX4daxxTZkVEGzwIhALvkClnx5m5eAT6**

**Lxw7LZtEQcH6JENhJTMArwLf3sXwi**

**apns-id = eabeae54-14a8-11e5-b60b-1697f925ec7b**

**apns-expiration = 0**

**apns-priority = 10**

**apns-topic = <MyAppTopic>**

**DATA**

**+ END\_STREAM**

**{ "aps" : { "alert" : "Hello" }**

****Sample request for a certificate with multiple topics****

**HEADERS**

**- END\_STREAM**

**+ END\_HEADERS**

**:method = POST**

**:scheme = https**

**:path = /3/device/00fc13adff785122b4ad28809a3420982341241421348097878e577c991de8f0**

**host = api.development.push.apple.com**

**apns-id = eabeae54-14a8-11e5-b60b-1697f925ec7b**

**apns-expiration = 0**

**apns-priority = 10**

**apns-topic = <MyAppTopic>**

**DATA**

**+ END\_STREAM**

**{ "aps" : { "alert" : "Hello" } }**

****Sample response for a successful request****

**HEADERS**

**+ END\_STREAM**

**+ END\_HEADERS**

**apns-id = eabeae54-14a8-11e5-b60b-1697f925ec7b**

**:status = 200**

****Sample response for a request that encountered an error****

**HEADERS**

**- END\_STREAM**

**+ END\_HEADERS**

**:status = 400**

**content-type = application/json**

**apns-id: <a\_UUID>**

**DATA**

**+ END\_STREAM**

**{ "reason" : "BadDeviceToken"}**

##### **Binary API**

Legacy API，推荐尽量用HTTP/2协议的接口

###### Generral Provider Requirement

1. stream TCP socket
2. Site:gateway.push.apple.com/sandbox.push.apple.com:2195
3. Asynchronous
4. TSL/SSL to establish channel, SSL证书可用开发者帐号制作

要是与APNs建立TSL通道，就需要Provider要安装Entrust Secure CA root certificate，macOS是默认安装在keychian中，其他系统要手动安装，下载证书：[CA](https://www.entrust.com/ssl-certificates/)

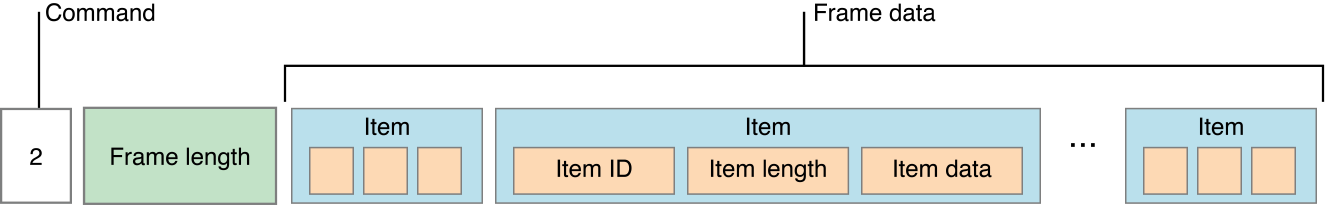
###### Provider Responsibility

* 创建Payload
* 设置badge
* 连接feedback服务，获取反复上报分发失败设备列表，然后停止向这些设备推送消息

###### Binary Interface & Notification Format

接口使用普通的二进制流TCP socket，为了性能上的考虑，在单个传输中批量处理多个通知，通过使用TCP/IP Nagle算法。

Notification Format(big endian)



Top-level fields for remote notifications

| Field name | Length | Discussion |
| --- | --- | --- |
| Command | 1 byte | Populate with the number 2 |
| Frame length | 4 bytes | The size of the frame data. |
| Frame data | variable length | The frame contains the body, structured as a series of items. |

Fields for remote notification frames

| Field name | Length | Discussion |
| --- | --- | --- |
| Item ID | 1 byte | The item identifier, as listed in Table A-3. For example, the item identifier of the payload is 2. |
| Item data length | 2 bytes | The size of the item data. |
| Item data | variable length | The value for the item. |

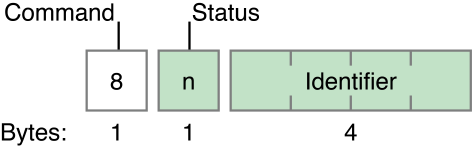
Item identifiers for remote notifications

| Item ID | Item Name | Length | Data |
| --- | --- | --- | --- |
| 1 | Device token | 32bytes | The device token in binary form, as was registered by the device. |
| 2 | Payload | <=2kb | The JSON-formatted payload.  The payload must not be null-terminated. |
| 3 | Notification identifier | 4bytes | 非必须；任意不透明标识通知的；目的是上报错误到Provider；允许在APNs 遇到错误的时候重发通知 |
| 4 | Expiration date | 4 bytes | A UNIX epoch date expressed in seconds (UTC)  标识通知消息不再有效且可取消  != 0, APNs 存储通知，然后在尝试推送一次；== 0，消息过期，APNs不再保存通知消息 |
| 5 | Priority | 1 byte | 10，立即发送，客户端必须配置(alert, sound, or badge)，如果Payload中只有content-available，就会报错  5，设备在某个省电的时候收到推送消息，这样的消息可能会被分群，然后突然爆发，他们是被压制的，导致不会被分发 |

**如果Provider发送的通知请求被APNs接受了，APNs不会返回任何东西；**

**如果Provider发送的通知请求是畸形或不合法，APNs返回相应的错误码并关闭连接，之后使用这个这个连接发送的通知都会被取消，必须重连**

**Format of error-response packet**

****

**Codes in error-response packet**

| Status code | Description |
| --- | --- |
| 0 | No errors encountered |
| 1 | Processing error |
| 2 | Missing device token |
| 3 | Missing topic |
| 4 | Missing payload |
| 5 | Invalid token size |
| 6 | Invalid topic size |
| 7 | Invalid payload size |
| 8 | Invalid token |
| 10 | Shutdown |
| 128 | Protocol error (APNs could not parse the notification) |
| 255 | None (unknown) |

**在错误响应中的通知消息Identifier是最后一个发送成功的消息Identifier**

**发布环境和开发环境的Device token是不一样的**

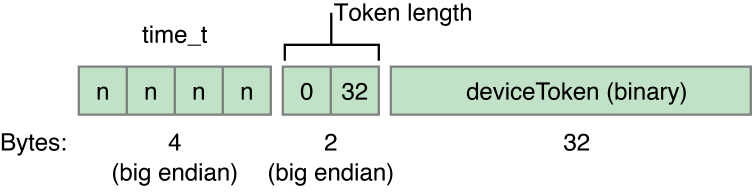
###### The Feedback Service

1. **当因为目标App在设备上不存在而导致推送消息不能被分发，feedback 服务将Device Token添加到列表中。**
2. **过期的推送消息在分发之前不会被认为是失败的，且不会影响feedback服务。**
3. **每天都查询feedback服务，获取Device Token列表，使用时间戳去验证没有被注册的Token，APNs会监控Provider的feedback，抑制推送向不存在App的设备推送消息。**
4. **Feedback服务为每一个App(bundle ID)维护一个单独的列表。**

**服务地址：feedback.push.apple.com/feedback.sandbox.push.apple.com:2196**

**建立连接：TCP TSL/SSL,与推送是一样的**

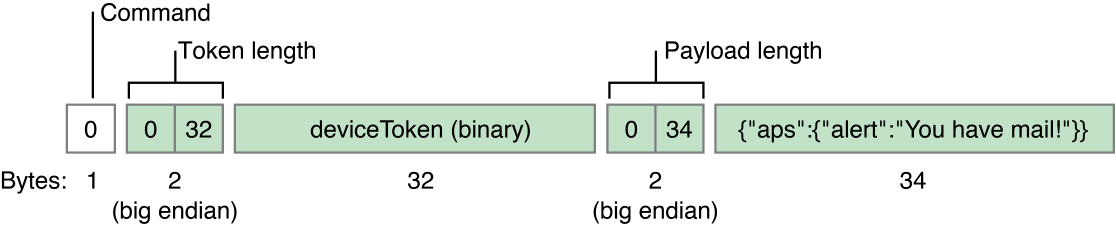
**读取二进制元组：**

****

|  |  |
| --- | --- |
| Timestamp | A timestamp (as a four-byte time\_t value) indicating when APNs determined that the app no longer exists on the device. This value, which is in network order, represents the seconds since 12:00 midnight on January 1, 1970 UTC. |
| Token length | The length of the device token as a two-byte integer value in network order. |
| Device token | The device token in binary format. |

**查询一次，无效的token列表就会被清空**

###### Legacy Notification Format

****

****Sending a notification in the legacy format via the binary interface****

static bool sendPayload(SSL \*sslPtr, char \*deviceTokenBinary, char \*payloadBuff, size\_t payloadLength)

{

bool rtn = false;

if (sslPtr && deviceTokenBinary && payloadBuff && payloadLength)

{

uint8\_t command = 0; /\* command number \*/

char binaryMessageBuff[sizeof(uint8\_t) + sizeof(uint16\_t) +

DEVICE\_BINARY\_SIZE + sizeof(uint16\_t) + MAXPAYLOAD\_SIZE];

/\* message format is, |COMMAND|TOKENLEN|TOKEN|PAYLOADLEN|PAYLOAD| \*/

char \*binaryMessagePt = binaryMessageBuff;

uint16\_t networkOrderTokenLength = htons(DEVICE\_BINARY\_SIZE);

uint16\_t networkOrderPayloadLength = htons(payloadLength);

/\* command \*/

\*binaryMessagePt++ = command;

/\* token length network order \*/

memcpy(binaryMessagePt, &networkOrderTokenLength, sizeof(uint16\_t));

binaryMessagePt += sizeof(uint16\_t);

/\* device token \*/

memcpy(binaryMessagePt, deviceTokenBinary, DEVICE\_BINARY\_SIZE);

binaryMessagePt += DEVICE\_BINARY\_SIZE;

/\* payload length network order \*/

memcpy(binaryMessagePt, &networkOrderPayloadLength, sizeof(uint16\_t));

binaryMessagePt += sizeof(uint16\_t);

/\* payload \*/

memcpy(binaryMessagePt, payloadBuff, payloadLength);

binaryMessagePt += payloadLength;

if (SSL\_write(sslPtr, binaryMessageBuff, (binaryMessagePt - binaryMessageBuff)) > 0)

rtn = true;

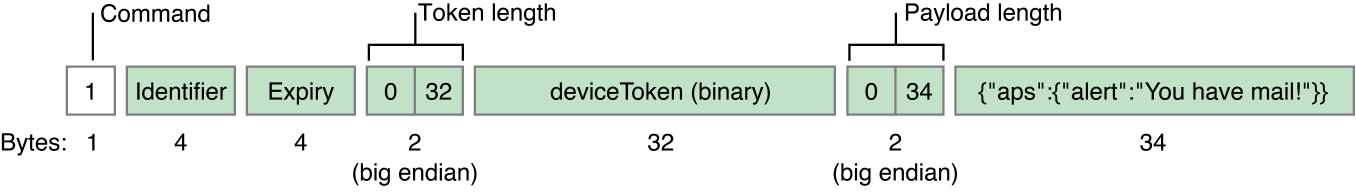
}

return rtn;

}

###### Enhanced Notification Format

* Error Response
* Notification expiration



****Sending a notification in the enhanced legacy format via the binary interface****

static bool sendPayload(SSL \*sslPtr, char \*deviceTokenBinary, char \*payloadBuff, size\_t payloadLength)

{

bool rtn = false;

if (sslPtr && deviceTokenBinary && payloadBuff && payloadLength)

{

uint8\_t command = 1; /\* command number \*/

char binaryMessageBuff[sizeof(uint8\_t) + sizeof(uint32\_t) + sizeof(uint32\_t) + sizeof(uint16\_t) +

DEVICE\_BINARY\_SIZE + sizeof(uint16\_t) + MAXPAYLOAD\_SIZE];

/\* message format is, |COMMAND|ID|EXPIRY|TOKENLEN|TOKEN|PAYLOADLEN|PAYLOAD| \*/

char \*binaryMessagePt = binaryMessageBuff;

uint32\_t whicheverOrderIWantToGetBackInAErrorResponse\_ID = 1234;

uint32\_t networkOrderExpiryEpochUTC = htonl(time(NULL)+86400); // expire message if not delivered in 1 day

uint16\_t networkOrderTokenLength = htons(DEVICE\_BINARY\_SIZE);

uint16\_t networkOrderPayloadLength = htons(payloadLength);

/\* command \*/

\*binaryMessagePt++ = command;

/\* provider preference ordered ID \*/

memcpy(binaryMessagePt, &whicheverOrderIWantToGetBackInAErrorResponse\_ID, sizeof(uint32\_t));

binaryMessagePt += sizeof(uint32\_t);

/\* expiry date network order \*/

memcpy(binaryMessagePt, &networkOrderExpiryEpochUTC, sizeof(uint32\_t));

binaryMessagePt += sizeof(uint32\_t);

/\* token length network order \*/

memcpy(binaryMessagePt, &networkOrderTokenLength, sizeof(uint16\_t));

binaryMessagePt += sizeof(uint16\_t);

/\* device token \*/

memcpy(binaryMessagePt, deviceTokenBinary, DEVICE\_BINARY\_SIZE);

binaryMessagePt += DEVICE\_BINARY\_SIZE;

/\* payload length network order \*/

memcpy(binaryMessagePt, &networkOrderPayloadLength, sizeof(uint16\_t));

binaryMessagePt += sizeof(uint16\_t);

/\* payload \*/

memcpy(binaryMessagePt, payloadBuff, payloadLength);

binaryMessagePt += payloadLength;

if (SSL\_write(sslPtr, binaryMessageBuff, (binaryMessagePt - binaryMessageBuff)) > 0)

rtn = true;

}

return rtn;

}

## ****APS dictionary Keys****

**aps字段，最重要的，它是一个字典类型。主要功能：**

* **Displays an alert message to the user**
* **Applies a badge to the app’s icon**
* **Plays a sound**
* **Delivers the notification silently**

**支持的字段类型是**dictionary (object), array, string, number, and Boolean

可以包含其他自定义的字段，用来传递开发者自定义的数据

### **APS Dictionary Keys**

**aps字典支持的字段是：**

| Key | Value type | Comment |
| --- | --- | --- |
| alert | Dictionary or String | **Include this key when you want the system to display a standard alert or a banner. The notification settings for your app on the user’s device determine whether an alert or banner is displayed.**  **The preferred value for this key is a dictionary, the keys for which are listed in**[**Table 9-2**](https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/PayloadKeyReference.html#//apple_ref/doc/uid/TP40008194-CH17-SW5)**. If you specify a string as the value of this key, that string is displayed as the message text of the alert or banner.**  **The JSON \U notation is not supported. Put the actual UTF-8 character in the alert text instead.** |
| badge | Number | **Include this key when you want the system to modify the badge of your app icon.**  **If this key is not included in the dictionary, the badge is not changed. To remove the badge, set the value of this key to 0.** |
| sound | String | **Include this key when you want the system to play a sound. The value of this key is the name of a sound file in your app’s main bundle or in the Library/Sounds folder of your app’s data container. If the sound file cannot be found, or if you specify default for the value, the system plays the default alert sound.**  **For details about providing sound files for notifications; see Preparing Custom Alert Sounds.** |
| content-available | Number | **Include this key with a value of 1 to configure a silent notification. When this key is present, the system wakes up your app in the background and delivers the notification to its app delegate. For information about configuring and handling silent notifications, see Configuring a Silent Notification.** |
| category | String | **Provide this key with a string value that represents the notification’s type. This value corresponds to the value in the identifier property of one of your app’s registered categories. To learn more about using custom actions, see Configuring Categories and Actionable Notifications.** |
| thread-id | String | **Provide this key with a string value that represents the app-specific identifier for grouping notifications. If you provide a Notification Content app extension, you can use this value to group your notifications together. For local notifications, this key corresponds to the threadIdentifier property of the UNNotificationContent object.** |
| mutable-content | Number | **指定消息是不是可以被修改** |

### ****Alert Keys****

| Key | Value type | Comment |
| --- | --- | --- |
| title | String | A short string describing the purpose of the notification. Apple Watch displays this string as part of the notification interface. This string is displayed only briefly and should be crafted so that it can be understood quickly. This key was added in iOS 8.2. |
| body | String | The text of the alert message. |
| title-loc-key | String or null | The key to a title string in the Localizable.stringsfile for the current localization. The key string can be formatted with %@ and %n$@ specifiers to take the variables specified in the title-loc-args array. See Localizing the Content of Your Remote Notifications for more information. This key was added in iOS 8.2. |
| title-loc-args | Array of strings or null | Variable string values to appear in place of the format specifiers in title-loc-key. See Localizing the Content of Your Remote Notifications for more information. This key was added in iOS 8.2. |
| action-loc-key | String or null | If a string is specified, the system displays an alert that includes the Close and View buttons. The string is used as a key to get a localized string in the current localization to use for the right button’s title instead of “View”. See Localizing the Content of Your Remote Notifications for more information. |
| loc-key | String | A key to an alert-message string in a Localizable.strings file for the current localization (which is set by the user’s language preference). The key string can be formatted with %@ and %n$@specifiers to take the variables specified in the loc-args array. See Localizing the Content of Your Remote Notifications for more information. |
| loc-args | Array of strings | Variable string values to appear in place of the format specifiers in loc-key. See Localizing the Content of Your Remote Notifications for more information. |
| launch-image | String | The filename of an image file in the app bundle, with or without the filename extension. The image is used as the launch image when users tap the action button or move the action slider. If this property is not specified, the system either uses the previous snapshot, uses the image identified by the UILaunchImageFile key in the app’s Info.plist file, or falls back to Default.png. |

# PushKit

## 概览

PushKit 框架可以发送直接让App来处理的指定类型的通知消息，例如VoIP invitation，watchOS complication update，file provider change

与UserNotifications框架不同，PushKit通知不会被展现给用户，也不会有(badge, alert, sound)

优势：

* 如果App没有运行，一旦收到通知，系统会自动启动它，尽管你可以使用静默通知在后台更新App，但是App不能保证当静默通知到达时，一定会被启动
* App被给予runtime来处理通知，即使App正在后台运行
* 只有当PushKit通知到达的时候，设备才会被唤醒，提高电池寿命
* PushKit的消息体能包含更多的数据，相比较于UserNotifications

## 开发指南

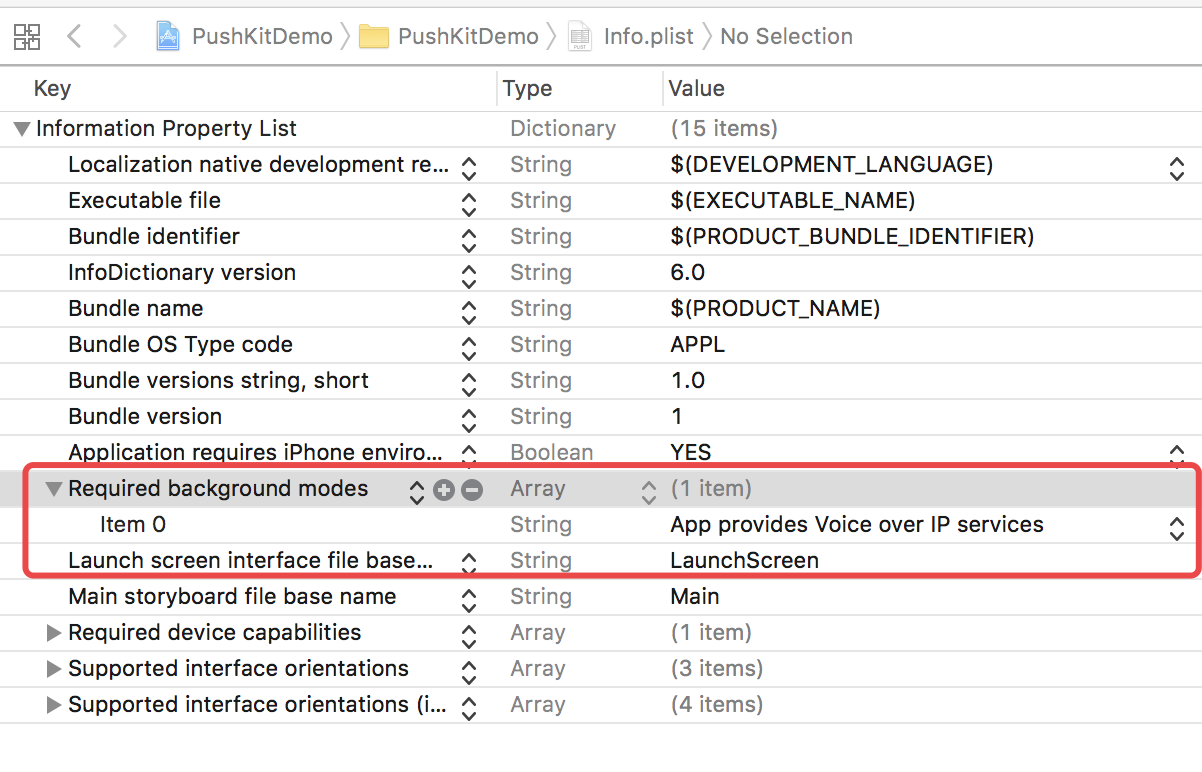
### 客户端

以下是以Xcode9为背景

Step 1, 工程引用PushKit.Framework

Step 2, 选择【Target】-> 【Capabilities】->打开【Background Modes】

Step 3, 在工程配置文件Info.plist文件中的【Background Modes】添加VoIP选项，如图：



Step 4, 注册通知，实现回调方法，具体演示代码(Swift 4):

import PushKit

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate, PKPushRegistryDelegate {

func pushRegistry(\_ registry: PKPushRegistry, didUpdate pushCredentials: PKPushCredentials, for type: PKPushType) {

print("token is \(((((pushCredentials.token as NSData).description as NSString).trimmingCharacters(in: CharacterSet(charactersIn: "<>")) as NSString) as NSString).replacingOccurrences(of: " ", with: ""))")

}

func pushRegistry(\_ registry: PKPushRegistry, didInvalidatePushTokenFor type: PKPushType) {

}

func pushRegistry(\_ registry: PKPushRegistry, didReceiveIncomingPushWith payload: PKPushPayload, for type: PKPushType, completion: @escaping () -> Void) {

print(payload.dictionaryPayload)

}

var window: UIWindow?

fileprivate var voipRegistry:PKPushRegistry!

func registerForVoIPPushes() {

self.voipRegistry = PKPushRegistry(queue: DispatchQueue.main)

self.voipRegistry.desiredPushTypes = [PKPushType.voIP]

self.voipRegistry.delegate = self

}

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

// Override point for customization after application launch.

registerForVoIPPushes();

return true

}

}

在回调方法中拿到Token,然后将其发送到自己开发的服务器，到此客户端的工作就基本完成了。

### 证书制作

在开发者账号下创建证书，选择【Production】->【VoIP Service Certificate】，按照制作推送证书的步骤操作即可。