移动推送

最佳实践

腾讯云

文档目录

最佳实践
iOS 平台角标功能实践

最近更新时间：2023-06-01 15:55:55

移动推送提供一系列角标控制相关方法供您使用，以下列举两个经典场景下的角标实现方法。
本文档为 iOS 平台角标最佳实践，Android 平台角标适配请查看 《Android 角标适配指南》。

场景一：打开应用时清空应用的角标数及通知栏消息

场景描述

- 应用未启动时，角标的数等于当前设备收到该应用的推送条数。
- 应用进入前台时，将角标的数量设置为0，清空通知栏消息，同时将云端角标数设置为0。

实现方法

此类场景建议使用角标自增方案，如下：

1. 通过 API 创建推送 时，设置应用角标数 badge_type = -2，角标数字自动加1。
2. 当 App 启动时，先调用**清空应用角标并清空通知栏**方法来清空本地角标数和通知栏消息，实现代码如下图：

```
/// 启动TPNS
- (void)xgStart {
    /// 控制台打印TPNS日志，开发调试建议开启
    [[XGPush defaultManager] setEnableDebug:YES];

    /// 自定义通知栏消息行为，有自定义消息行为需要使用
    //    [self setNotificationConfigure];

    /// 非广州集群，请开启对应集群配置（广州集群无需使用），此函数需要在startXGWithAccessID函数之前调用
    //    [self configHost];

    /// 启动TPNS服务
    [[XGPush defaultManager] startXGWithAccessID:TPNS_ACCESS_ID accessKey:TPNS_ACCESS_KEY delegate:self];

    /// 角标数目清零,通知中心清空
    if ([XGPush defaultManager].xgApplicationBadgeNumber > 0) {
        TPNS_DISPATCH_MAIN_SYNC_SAFE(^{
            [XGPush defaultManager].xgApplicationBadgeNumber = 0;
        });
    }
}
```

3. 清空云端角标数，实现代码如下图：

```
/// 注册推送服务成功回调
/// @param deviceToken APNs 生成的Device Token
/// @param xgToken TPNS 生成的 Token，推送消息时需要使用此值。TPNS 维护此值与APNs 的 Device Token的映射关系
/// @param error 错误信息，若error为nil则注册推送服务成功
- (void)xgPushDidRegisteredDeviceToken:(nullable NSString *)deviceToken xgToken:(nullable NSString *)xgToken
    error:(nullable NSError *)error {
    NSLog(@"%s, result %@, error %@", __FUNCTION__, error ? @"NO" : @"OK", error);
    NSString *errorStr = !error ? NSLocalizedString(@"success", nil) : NSLocalizedString(@"failed", nil);
    NSString *message = [NSString stringWithFormat:@"%@%@", NSLocalizedString(@"register_app", nil), errorStr];
    [TPNSCommonMethod showAlertViewWithText:message];
    //在注册完成后上报角标数目
    if (!error) {
        //重置服务端自动+1基数
        [[XGPush defaultManager] setBadge:0];
    }
    //设置是否注册成功
    self.isTPNSRegistSuccess = error ? false : true;
}
```

4. 需要更新云端角标数时，需调用下方接口将角标值同步到移动推送服务器，下次推送时以此值为基准。如当前移动推送服务器角标值同步为 n，则下次收到推送时 App 角标值为 n+1。

```
//将角标值同步到 TPNS 服务器，下次推送时以此值为基准
- (void)setBadge:(NSInteger)badgeNumber;
```

注意

1. setBadge 方法依赖移动推送自建通道实现角标同步，应用在退入后台时，移动推送自建通道会断开连接，此时调用 setBadge 会失败，这种情况只能使用自定义角标数方案。
2. 应用进入前台时，移动推送自建通道会尝试恢复连接，在连接恢复之前调用 setBadge 会失败，这种情况只能使用自定义角标数方案。
3. 冷启动应用时需要在注册方法回调中使用 setBadge。
4. 监听移动推送网络连接的状态，详见 如何监听移动推送网络连接的状态？ 。

场景二：角标数包含通知栏消息数与应用内消息数

场景描述

应用的角标数 = 通知栏消息数 + 应用内消息数。用户点击某条通知拉起App时，通知栏消息数会发生变化，此时需要更新App角标数。
在该场景下，开发者需要维护设备的总消息数（包括通知栏消息数和应用内消息数）。

实现方法

1. 获取通知栏消息数，代码如下：

```
//需要获取通知栏的消息数
[[UNUserNotificationCenter currentNotificationCenter]
getDeliveredNotificationsWithCompletionHandler:^(NSArray<UNNotification *> *notifications) {
NSLog(@"notifications count:%d.",notifications.count);
}];
```

2. 假设通知栏消息数为a，应用内消息数为b，设置App角标数代码如下：

```
//设置应用角标数
[XGPush defaultManager].xgApplicationBadgeNumber = a + b;
```

3. 建议使用**自定义角标数**方案，方法如下：
   通过 API 创建 推送时，直接设置应用角标数badge_type >= 0，自定义角标数字为总消息数。如总消息数为10，则设置badge_type = 10。

问题答疑

如何只清空角标数，但是在通知中心保留推送通知？

```
//不在appIcon上显示推送数量，但是在系统通知栏保留推送通知的方法
+ (void)resetBageNumber{
    if(APNS_IS_IOS11_LATER){
        //iOS 11后，直接设置badgeNumber = -1就生效了
        [UIApplication sharedApplication].applicationIconBadgeNumber = -1;
    }else{
        UILocalNotification *clearEpisodeNotification = [[UILocalNotification alloc] init];
        clearEpisodeNotification.fireDate = [NSDate dateWithTimeIntervalSinceNow:(0.3)];
        clearEpisodeNotification.timeZone = [NSTimeZone defaultTimeZone];
        clearEpisodeNotification.applicationIconBadgeNumber = -1;
        [[UIApplication sharedApplication] scheduleLocalNotification:clearEpisodeNotification];
    }
}
```

如何设置角标数，但是在通知中心保留推送通知？

```
#define APNS_IS_IOS11_LATER ([UIDevice currentDevice].systemVersion.floatValue >= 11.0f)
//在appIcon上推送角标数量逻辑，但是在系统通知栏保留推送通知的方法
+ (void)resetBageNumber:(int) number{
/// 如果是非0数，直接设置
if(number){
[XGPush defaultManager].xgApplicationBadgeNumber = number;
return;
}
/// 如果是0，则通过如下逻辑设置
if(APNS_IS_IOS11_LATER){
//iOS 11后，直接设置badgeNumber = -1就生效了
[UIApplication sharedApplication].applicationIconBadgeNumber = -1;
}else{
UILocalNotification *clearEpisodeNotification = [[UILocalNotificationalloc] init];
clearEpisodeNotification.fireDate = [NSDatedateWithTimeIntervalSinceNow:(0.3)];
clearEpisodeNotification.timeZone = [NSTimeZonedefaultTimeZone];
clearEpisodeNotification.applicationIconBadgeNumber = -1;
[[UIApplication sharedApplication] scheduleLocalNotification:clearEpisodeNotification];
}
}
```

如何在通知中心删除特定通知？

```
//objectID：通知下发的消息id，用以辨识通知
- (void)removeNotificationsForObjectID:(ObjectID *)objectID {
    __weak __typeof(self) weakSelf = self;
    [[UNUserNotificationCenter currentNotificationCenter]
getDeliveredNotificationsWithCompletionHandler:^(NSArray<UNNotification *> *notifications) {
        __strong __typeof(weakSelf) self = weakSelf;
        NSMutableArray <NSString *> *identifiersToRemove = [@[] mutableCopy];
        for (UNNotification *notification in notifications) {
            //筛选符合删除条件的通知
            ObjectID *objectIDFromNotification = [self marshalNotification:notification];
            if ([objectID isEqual:objectIDFromNotification]) {
                [identifiersToRemove addObject:notification.request.identifier];
            }
        }
        [[UNUserNotificationCenter currentNotificationCenter]
removeDeliveredNotificationsWithIdentifiers:identifiersToRemove];
    }];
}
```

> **注意**
> 此方法仅支持 iOS 10 及以上的操作系统版本。

如何在 App 没有启动时，只更新应用角标？

使用 自定义角标数方案，创建没有内容，只有角标数的推送。

如何监听移动推送网络连接的状态？

建议使用 v1.3.1.0 及以上版本 SDK 的如下方法：

```
// TPNS网络连接成功
- (void)xgPushNetworkConnected;
// TPNS网络连接断开
- (void)xgPushNetworkDisconnected;
```

通过 API 创建推送 时，如何让角标数不变？

badge_type = -1：角标数字不变。

如何查询App的角标修改方法的函数堆栈？

可以通过 hook 系统类 UIApplication的setApplicationIconBadgeNumber:方法打印函数调用堆栈，从而发现调用者信息，代码如下：
UIApplication+ApplicationIconBadgeNumber.h 文件

```
#import <UIKit/UIKit.h>

NS_ASSUME_NONNULL_BEGIN

@interface UIApplication (ApplicationIconBadgeNumber)
```

```
@end

NS_ASSUME_NONNULL_END
```

UIApplication+ApplicationIconBadgeNumber.m 文件

```
#import "UIApplication+ApplicationIconBadgeNumber.h"
#import <objc/runtime.h>

@implementation UIApplication (ApplicationIconBadgeNumber)

//load类方法(当某个类的代码被读到内存后调用)
+ (void)load
{
SEL origSel = @selector(setApplicationIconBadgeNumber:);
SEL swizSel = @selector(swiz_setApplicationIconBadgeNumber:);
[UIApplication swizzleMethods:[self class] originalSelector:origSel swizzledSelector:swizSel];
}

//交换两个方法的实现
+ (void)swizzleMethods:(Class)class originalSelector:(SEL)origSel swizzledSelector:(SEL)swizSel
{
Method origMethod = class_getInstanceMethod(class, origSel);
Method swizMethod = class_getInstanceMethod(class, swizSel);
BOOL didAddMethod = class_addMethod(class, origSel, method_getImplementation(swizMethod),
method_getTypeEncoding(swizMethod));
if (didAddMethod){
class_replaceMethod(class, swizSel, method_getImplementation(origMethod),
method_getTypeEncoding(origMethod));
}else{
method_exchangeImplementations(origMethod, swizMethod);
}
}

/// hook设置角标
- (void)swiz_setApplicationIconBadgeNumber:(NSInteger)number
{
NSLog(@"调用了swiz_setApplicationIconBadgeNumber方法");
/// 打印当前函数调用堆栈
NSArray *syms = [NSThread  callStackSymbols];
for(int i = 0 ; i < [syms count]; i++){
NSLog(@"<%@ %p> %@ - caller: %@ ", [self class], self, NSStringFromSelector(_cmd),[syms
objectAtIndex:i]);
}
//执行这句的时候跳转到setApplicationIconBadgeNumber方法中
[self swiz_setApplicationIconBadgeNumber:number];
}

@end
```

将上面的两个文件加入到自己的工程中使用。

下图是加入到移动推送 Demo 示例：