Tencent Push Notification Service Best Practices Product Documentation

Contents

Best Practices
iOS Badge Feature Practice

Last updated : 2021-07-27 11:03:45

TPNS provides a series of badge control methods. The following introduces how to implement the badge feature in two classic scenarios.
This document focuses on iOS badge best practices. For Android badge adaptation, see here.

Scenario 1: Clearing the Application's Badge Number and Notification Bar Messages when Opening the Application

Scenario description

- When the application is not started, make the badge number equal the number of the application's push messages received by the current device.
- When the application enters the background, set the badge number to 0, clear notification bar messages, and set the cloud badge number to 0.

Implementation method

For such a scenario, the badge number auto increase scheme is recommended. The process is as follows:
Step 1: when a push task is created via the push API, set `badge_type` to `-2` to enable the application's badge number to automatically increase by 1.
Step 2: when the application is started, call the "clearing the application's badge number and notification bar messages" method to clear the local badge number and notification bar messages. The implementation code is as follows.

Step 3: clear the cloud badge number. The implementation code is as follows.

Step 4: if the cloud badge number needs to be updated, call the following API to sync the badge value to the TPNS server, and the badge value will be used as the benchmark for the next push. For example, if the current TPNS server badge value is synchronized as N, then the application's badge value will be N+1 when the next push is received.

```
//Sync the badge value to the TPNS server, and the value will be used as the benchmark for the ne
xt push
- (void)setBadge:(NSInteger)badgeNumber;
```

Note :

1. The `setBadge` method relies on the TPNS channel to implement badge value sync. When the application enters the background, the TPNS channel will be disconnected, and calling the `setBadge` method will fail. In this case, only the custom badge number scheme is applicable.
2. When the application enters the foreground, the TPNS channel will try to reconnect, and calling the `setBadge` method will fail before the connection is resumed. In this case, only the custom badge number scheme is applicable.
3. When the application is cold started, `setBadge` must be used in the callback of the registration method.
4. For more information on how to listen for TPNS network connection status, please see here.

Scenario 2: Enabling the Badge Number to Be the Sum of Notification Bar Messages and In-App Messages

Scenario description

Application's badge number = Number of notification bar messages + Number of in-app messages. When a user clicks a notification to start the application, the number of notification bar messages changes, and the application's badge number needs to be updated.
In this scenario, developers need to maintain the device's total number of messages (including the number of notification bar messages and the number of in-app messages).

Implementation method

1. Obtain the number of notification bar messages. The code is as follows:

```
//Obtain the number of notification bar messages
[[UNUserNotificationCenter currentNotificationCenter] getDeliveredNotificationsWithCompletionHandler:^(NSArray<UNNotification *> *notifications) {
NSLog(@"notifications count:%d.",notifications.count);
}];
```

2. Assume that the number of notification bar messages is a and the number of in-app messages is b. The code for setting the application's badge number is as follows.

```
//Set the application's badge number
[XGPush defaultManager].xgApplicationBadgeNumber = a + b;
```

3. Use the **custom badge number** scheme (recommended). The method is as follows:
   When a push task is created via the push API, set `badge_type` to the total number of messages (a custom badge number that is equal to or greater than 0). For example, if the total number of messages is 10, set `badge_type` to `10`.

FAQs

How to clear the badge number but retain the push notifications in the notification center?

```
//Do not display the number of pushes in the application icon, but retain the push notifications
in the system notification bar
+ (void)resetBageNumber{
if(APNS_IS_IOS11_LATER){
//For iOS 11 or later, you only need to set `badgeNumber` to `-1`
[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];
}
}
```

How can I set the badge number but retain the push notifications in the notification center?

```
#define APNS_IS_IOS11_LATER ([UIDevice currentDevice].systemVersion.floatValue >= 11.0f)
// Display the badge number in the application icon, but retain the push notifications in the sys
tem notification bar.
+ (void)resetBageNumber:(int) number{
/// If the number is not `0`, set directly.
if(number){
[XGPush defaultManager].xgApplicationBadgeNumber = number;
return;
}
/// If the number is `0`, set by the following logic:
if(APNS_IS_IOS11_LATER){
//For iOS 11 or later, you only need to set `badgeNumber` to `-1`
[UIApplication sharedApplication].applicationIconBadgeNumber = -1;
}else{
UILocalNotification *clearEpisodeNotification = [[UILocalNotificationalloc] init];
clearEpisodeNotification.fireDate = [NSDatedateWithTimeIntervalSinceNow:(0.3)];
```

```
clearEpisodeNotification.timeZone = [NSTimeZonedefaultTimeZone];
clearEpisodeNotification.applicationIconBadgeNumber = -1;
[[UIApplication sharedApplication] scheduleLocalNotification:clearEpisodeNotification];
}
}
```

How to delete a specific notification from the notification center?

```
//objectID: push ID of the notification to delete
- (void)removeNotificationsForObjectID:(ObjectID *)objectID {
__weak __typeof(self) weakSelf = self;
[[UNUserNotificationCenter currentNotificationCenter] getDeliveredNotificationsWithCompletionHand
ler:^(NSArray<UNNotification *> *notifications) {
__strong __typeof(weakSelf) self = weakSelf;
NSMutableArray <NSString *> *identifiersToRemove = [@[] mutableCopy];
for (UNNotification *notification in notifications) {
//Filter the notification that meets the deletion criteria
ObjectID *objectIDFromNotification = [self marshalNotification:notification];
if ([objectID isEqual:objectIDFromNotification]) {
[identifiersToRemove addObject:notification.request.identifier];
}
}
[[UNUserNotificationCenter currentNotificationCenter] removeDeliveredNotificationsWithIdentifier
s:identifiersToRemove];
}];
}
```

Note :

This method applies only to iOS 10 and later.

How to update only the application's badge number when the application is not started?

Use the custom badge number scheme to create a push with only a badge number but without content.

How to listen for TPNS network connection status?

For SDK v1.3.1.0 or later, the recommended method is as follows:

```
// TPNS network connected
- (void)xgPushNetworkConnected;
// TPNS network disconnected
- (void)xgPushNetworkDisconnected;
```

How to keep the badge number unchanged when I create a push via the push API?

Set `badge_type` to `-1` to make the badge number remain unchanged.

How can I query the function stack of the app's badge modification method?

You can use the `setApplicationIconBadgeNumber:` method of the `UIApplication` hook system class to print the function call stack to find out the caller information. The code is as follows:

`UIApplication+ApplicationIconBadgeNumber.h` file

```
#import <UIKit/UIKit.h>
NS_ASSUME_NONNULL_BEGIN
@interface UIApplication (ApplicationIconBadgeNumber)
@end
NS_ASSUME_NONNULL_END
```

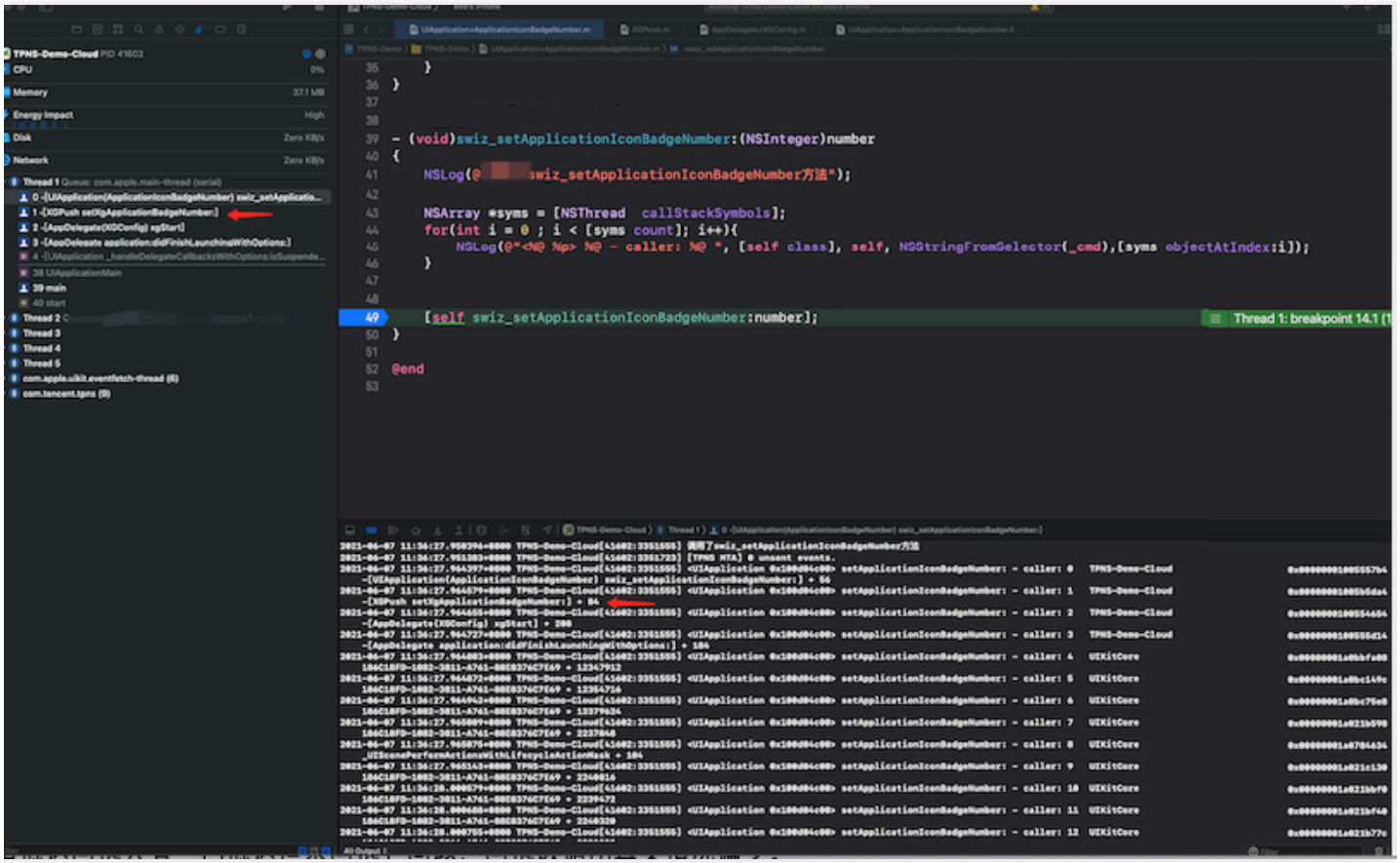
`UIApplication+ApplicationIconBadgeNumber.m` file

```
#import "UIApplication+ApplicationIconBadgeNumber.h"
#import <objc/runtime.h>
@implementation UIApplication (ApplicationIconBadgeNumber)
// Load class method (called when the code of a certain class is read into memory)
+ (void)load
{
SEL origSel = @selector(setApplicationIconBadgeNumber:);
SEL swizSel = @selector(swiz_setApplicationIconBadgeNumber:);
[UIApplication swizzleMethods:[self class] originalSelector:origSel swizzledSelector:swizSel];
}
// Swap the implementations of the two methods.
+ (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);
}
}
/// Set the badge using hook.
- (void)swiz_setApplicationIconBadgeNumber:(NSInteger)number
{
NSLog(@" called the `swiz_setApplicationIconBadgeNumber` method.");
/// Print the current function call stack.
```

```
NSArray *syms = [NSThread callStackSymbols];
for(int i = 0 ; i < [syms count]; i++){
NSLog(@"<%@ %p> %@ - caller: %@ ", [self class], self, NSStringFromSelector(_cmd),[syms objectAtI
ndex:i]);
}
// Jump to the `setApplicationIconBadgeNumber` method when this statement is executed.
[self swiz_setApplicationIconBadgeNumber:number];
}
@end
```

Add the above two files to your project.

The following figure is an example of adding them to the TPNS demo:



IM Offiline Push

Last updated : 2021-08-06 12:12:38

TPNS provides Instant Messaging (IM) with the offline push capability. For an instant messaging app or an app with instant messaging features, you are advised to integrate it with IM for instant messaging capabilities in different scenarios. You can activate IM in the IM console.

You can integrate an instant messaging app with IM and the SDK of TPNS for a complete push service. This helps improve the push delivery rate and implement diversified push scenarios without the need to integrate with different vendor channels separately. End users would like to be able to get the latest messages at any time, but the performance and power of mobile devices are limited. When the app is running in the background, to avoid excessive resource consumption caused by maintaining a persistent connection, you can use TPNS for offline message notifications. It provides a more stable system-level persistent connection, allowing end users to receive push messages at any time and greatly reducing resource consumption.

- Guide for integrating IM with TPNS offline push for Android: Offline Push (Android)
- Guide for integrating IM with TPNS offline push for iOS: Offline Push (iOS)