extern NSString \*const kAPNetworkDidSetupNotification; // 建立连接

extern NSString \*const kAPNetworkDidCloseNotification; // 关闭连接

extern NSString \*const kAPNetworkDidRegisterNotification; // 注册成功

extern NSString \*const kAPNetworkDidLoginNotification; // 登录成功

extern NSString \*const kAPNetworkDidReceiveMessageNotification; // 收到消息(非APNS) ,JPush 使用的是自己的二进制流,不是XMPP

extern NSString \*const kAPServiceErrorNotification; // 错误提示

实现方法:

NSNotificationCenter \*defaultCenter = [NSNotificationCenter defaultCenter];

[defaultCenter addObserver:self selector:@selector(networkDidSetup:) name:kAPNetworkDidSetupNotification object:nil];

[defaultCenter addObserver:self selector:@selector(networkDidClose:) name:kAPNetworkDidCloseNotification object:nil];

[defaultCenter addObserver:self selector:@selector(networkDidRegister:) name:kAPNetworkDidRegisterNotification object:nil];

[defaultCenter addObserver:self selector:@selector(networkDidLogin:) name:kAPNetworkDidLoginNotification object:nil];

[defaultCenter addObserver:self selector:@selector(networkDidReceiveMessage:) name:kAPNetworkDidReceiveMessageNotification object:nil];

// 以下四个接口是必须调用的

+ (void)setupWithOption:(NSDictionary \*)launchingOption; // 初始化

在- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions 添加

[APService setupWithOption:launchOptions];

+ (void)registerForRemoteNotificationTypes:(int)types; // 注册APNS类型

[APService registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge |

UIRemoteNotificationTypeSound |

UIRemoteNotificationTypeAlert)];

UIRemoteNotificationTypeBadge:

+ (void)registerDeviceToken:(NSData \*)deviceToken; // 向服务器上报Device Token

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

[APService registerDeviceToken:deviceToken];

}

+ (void)handleRemoteNotification:(NSDictionary \*)

remoteInfo; // 处理收到的APNS消息，向服务器上报收到APNS消息

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

[APService handleRemoteNotification:userInfo];

}

**别名 alias**

为安装了应用程序的用户，取个别名来标识。以后给该用户 Push 消息时，就可以用此别名来指定。

每个用户只能指定一个别名。

同一个应用程序内，对不同的用户，建议取不同的别名。这样，尽可能根据别名来唯一确定用户。

系统不限定一个别名只能指定一个用户。如果一个别名被指定到了多个用户，当给指定这个别名发消息时，[服务器端API](http://docs.jpush.cn/pages/viewpage.action?pageId=3309913)会同时给这多个用户发送消息。

举例：在一个用户要登录的游戏中，可能设置别名为 userid。游戏运营时，发现该用户 3 天没有玩游戏了，则根据 userid 调用[服务器端API](http://docs.jpush.cn/pages/viewpage.action?pageId=3309913)发通知到客户端提醒用户。

**标签 tag**

为安装了应用程序的用户，打上标签。其目的主要是方便开发者根据标签，来批量下发 Push 消息。

可为每个用户打多个标签。

不同应用程序、不同的用户，可以打同样的标签。

举例： game, old\_page,  women

# Method - setTagsWithAlias (with Callback)

调用此 API 来同时设置别名与标签，支持回掉函数。

需要理解的是，这个接口是覆盖逻辑，而不是增量逻辑。即新的调用会覆盖之前的设置。

在之前调用过后，如果需要再次改变别名与标签，只需要重新调用此 API 即可。

### 支持的版本

开始支持的版本：1.4.0

### 接口定义

|  |
| --- |
| + (void)setTags:(NSSet \*)tags alias:(NSString \*)alias callbackSelector:(SEL)cbSelector object:(id)theTarget; |

### 参数说明

* alias  
  + nil 此次调用不设置此值。
  + 空字符串 （@""）表示取消之前的设置。
  + 每次调用设置有效的别名，覆盖之前的设置。
  + 有效的别名组成：字母（区分大小写）、数字、下划线、汉字。
  + 限制：alias 命名长度限制为 40 字节。（判断长度需采用UTF-8编码）
* tags  
  + nil 此次调用不设置此值。
  + 空集合（[NSSet set]）表示取消之前的设置。
  + 每次调用至少设置一个 tag，覆盖之前的设置，不是新增。
  + 有效的标签组成：字母（区分大小写）、数字、下划线、汉字。
  + 限制：每个 tag 命名长度限制为 40 字节，最多支持设置 100 个 tag，但总长度不得超过1K字节。（判断长度需采用UTF-8编码）
  + 单个设备最多支持设置 100 个 tag。App 全局 tag 数量无限制。
* callbackSelector  
  + nil 此次调用不需要 Callback。
  + 用于回掉返回对应的参数 alias, tags。并返回对应的状态码：0为成功，其他返回码请参考错误码定义。
  + 回调函数请参考SDK 实现。
* theTarget
  + 参数值为实现了callbackSelector的实例对象。
  + nil 此次调用不需要 Callback。

|  |
| --- |
| - (void)tagsAliasCallback:(int)iResCode tags:(NSSet\*)tags alias:(NSString\*)alias {      NSLog(@"rescode: %d, \ntags: %@, \nalias: %@\n", iResCode, tags , alias);  } |

# Method - setTags

调用此 API 来设置标签，支持回掉函数。

该方法是 setTagsWithAlias (with Callback) 的简化版本，用于只变更标签的情况。

|  |  |
| --- | --- |
| http://docs.jpush.cn/images/icons/emoticons/information.png | **使用建议**  如果待设置的 alias / tags 是动态的，有可能在调用 setTagsWithAlias 时因为 alias / tags 无效而整调用失败。  调用此方法只设置 tags，可以排除可能的无效的 alias 对本次调用的影响。 |

### 支持的版本

开始支持的版本：1.4.0

### 接口定义

|  |
| --- |
| + (void)setTags:(NSSet \*)tags callbackSelector:(SEL)cbSelector object:(id)theTarget; |

### 参数说明

* tags  
  + 空集合（[NSSet set]）表示取消之前的设置。
  + 每次调用至少设置一个 tag，覆盖之前的设置，不是新增。
  + 有效的标签组成：字母（区分大小写）、数字、下划线、汉字。
  + 限制：每个 tag 命名长度限制为 40 字节，最多支持设置 100 个 tag，但总长度不得超过1K字节。（判断长度需采用UTF-8编码）
  + 单个设备最多支持设置 100 个 tag。App 全局 tag 数量无限制。
* callbackSelector  
  + 用于回掉返回对应的参数 alias, tags。并返回对应的状态码：0为成功，其他返回码请参考错误码定义。
  + 回调函数请参考SDK 实现。
* theTarget
  + 参数值为实现了callbackSelector的实例对象。
  + nil 此次调用不需要 Callback。

|  |
| --- |
| - (void)tagsAliasCallback:(int)iResCode tags:(NSSet\*)tags alias:(NSString\*)alias {      NSLog(@"rescode: %d, \ntags: %@, \nalias: %@\n", iResCode, tags , alias);  } |

# Method - setAlias

调用此 API 来设置别名，**支持回掉函数。**

该方法是 setTagsWithAlias (with Callback) 的简化版本，用于只变更别名的情况。

### 支持的版本

开始支持的版本：1.4.0

### 接口定义

|  |
| --- |
| + (void)setAlias:(NSString \*)alias callbackSelector:(SEL)cbSelector object:(id)theTarget; |

### 参数说明

* alias  
  + 空字符串 （@""）表示取消之前的设置。
  + 每次调用设置有效的别名，覆盖之前的设置。
  + 有效的别名组成：字母（区分大小写）、数字、下划线、汉字。
  + 限制：alias 命名长度限制为 40 字节。（判断长度需采用UTF-8编码）
* callbackSelector  
  + 用于回掉返回对应的参数 alias, tags。并返回对应的状态码：0为成功，其他返回码请参考错误码定义。
  + 回调函数请参考SDK 实现。
* theTarget
  + 参数值为实现了callbackSelector的实例对象。
  + nil 此次调用不需要 Callback。

|  |
| --- |
| - (void)tagsAliasCallback:(int)iResCode tags:(NSSet\*)tags alias:(NSString\*)alias {      NSLog(@"rescode: %d, \ntags: %@, \nalias: %@\n", iResCode, tags , alias);  } |

# Method - filterValidTags

用于过滤出正确可用的 tags。

如果总数量超出最大限制则返回最大数量的靠前的可用tags。

|  |  |
| --- | --- |
| http://docs.jpush.cn/images/icons/emoticons/information.png | **使用建议**  设置 tags 时，如果其中一个 tag 无效，则整个设置过程失败。  如果 App 的 tags 会在运行过程中动态设置，并且存在对 JPush SDK tag 规定的无效字符，则有可能一个 tag 无效导致这次调用里所有的 tags 更新失败。  这时你可以调用本方法 filterValidTags 来过滤掉无效的 tags，得到有效的 tags，再调用 JPush SDK 的 set tags / alias 方法。 |

### 支持的版本

开始支持的版本：1.4.0

### 接口定义

|  |
| --- |
| + (NSSet\*)filterValidTags:(NSSet\*)tags; |

### 参数说明

* tags  
  + 原 tag 集合。

### 接口返回

有效的 tag 集合。

# 错误码定义

| **Code** | **描述** | **详细解释** |
| --- | --- | --- |
| **6001** | 无效的设置，tag/alias 不应参数都为 null |  |
| **6002** | 设置超时 | 建议重试 |
| **6003** | alias 字符串不合法 | 有效的别名、标签组成：字母（区分大小写）、数字、下划线、汉字。 |
| **6004** | alias超长。最多 40个字节 | 中文 UTF-8 是 3 个字节 |
| **6005** | 某一个 tag 字符串不合法 | 有效的别名、标签组成：字母（区分大小写）、数字、下划线、汉字。 |
| **6006** | 某一个 tag 超长。一个 tag 最多 40个字符 | 中文 UTF-8 是 3 个字节 |
| **6007** | tags 数量超出限制。最多 100个 | 这是一台设备的限制。一个应用全局的标签数量无限制。 |
| **6008** | tag/alias 超出总长度限制。总长度最多 1K 字节 |  |

用户指定页面使用时长

## API

### 接口定义

|  |
| --- |
| + (void)startLogPageView:(NSString\*)pageName;  + (void)stopLogPageView:(NSString\*)pageName;  + (void)beginLogPageView:(NSString\*)pageName duration:(int)seconds; |

**参数说明**

* pageName 需要统计页面自定义名称
* duration      自定义的页面时间

### 调用说明

 应在所有的需要统计得页面得 viewWillAppear 和 viewWillDisappear 加入 startLogPageView 和 stopLogPageView 来统计当前页面的停留时间。

|  |  |
| --- | --- |
| http://docs.jpush.cn/images/icons/emoticons/information.png | 或者直接使用 beginLogPageView 来自定义加入页面和时间信息。 |

**代码示例**

|  |
| --- |
| - (void)viewWillAppear:(BOOL)animated      {          [super viewWillAppear:animated];          [APService startLogPageView:@"PageOne"];      }  - (void)viewWillDisappear:(BOOL)animated      {          [super viewWillDisappear:animated];          [APService stopLogPageView:@"PageOne"];      }  －(void)trackView     {         [APService beginLogPageView:@"PageTwo" duration:10];     } |