API接口文档

1.接口类说明

接口	说明
PushManager	SDK 功能接口，用于调用个推相关功能接口
Tag	标签结构，用于给用户打上标签（如可以用于精准推送）

2.获取PushManager对象

```
public static PushManager getInstance()
```

参数：

- 无

返回：

- 返回PushManager单例对象

说明：

- 用于获取单例 PushManager 对象实例，可以进行推送控制、设置标签、设置别名、设置静默时间等，所有个推提供的接口均是通过该实例调用

3.初始化服务

```
public void initialize(Context context, Class<T> service)
```

参数：

- context：应用的 ApplicationContext
- service: 应用自定义的推送服务载体Service，如果传null，则表示不使用自定义推送服务，而是使用SDK提供的标准Service。

返回：

- 无

说明：

- 初始化个推服务，接口调用后个推服务后台运行，联网成功后，通过广播或者IntentService回调接口返回 CID 信息。

4.注册消息接收服务

```
public void registerPushIntentService(Context context, Class<T> service)
```

参数：

- context：应用的 ApplicationContext
- service: 自定义的 Service

返回：

- 无

说明：

- 注册消息接收 IntentService，个推服务联网成功后，通过该注册的 Service 返回 CID、 在线状态、透传数据等信息。建议在 `initialize` 方法调用后，立即通过 `registerPushIntentService` 方法注册自定义IntentService。
- 如果调用了 `registerPushIntentService` 方法注册自定义IntentService，则SDK仅通过IntentService回调推送服务事件；
- 如果未调用 `registerPushIntentService` 方法进行注册，则原有的广播接收器仍然可以继续使用。

5.停止SDK服务

```
public void stopService(Context context)
```

参数：

- context：应用的 Context

返回：

- 无

说明：

- 接口 PushManager 中的 stopService，停止 SDK 服务，服务不会终止运行， 只是终止推送和联网功能
- 重新调用 initialize 接口初始化推送服务或者调用 turnOnPush 接口开启 push， 即可正常推送， 具体参考 initialize 和 turnOnPush 接口

6.开启推送

```
public void turnOnPush(Context context)
```

参数：

- context：应用的 Context

返回：

- 无

说明：

- 接口 PushManager 中的 turnOnPush， 开启Push推送， 默认是开启状态， 关闭状态则收不到推送。 turnOnPush 默认打开
- 如果已经调用了 stopService 接口停止了 SDK 服务，调用 turnOnPush 或者重新调用 initialize 之后即可正常推送
- 如果已经调用了 turnOffPush 接口关闭了推送， 只有调用 turnOnPush 之后才能正常推送

7.关闭推送

```
public void turnOffPush(Context context)
```

参数：

- context：应用的 Context

返回：

- 无

说明：

- 接口 PushManager 中的 turnOffPush， 关闭Push推送， 关闭后则无法收到推送消息
- 如果已经调用了 stopService 接口停止了 SDK 服务，调用 turnOnPush 或者重新调用 initialize 之后即可正常推送
- 如果已经调用了 turnOffPush 接口关闭了推送， 只有调用 turnOnPush 之后才能正常推送

8.设置标签

```
public int setTag(Context context,Tag[] tag, String sn);
```

参数：

- context：应用的 Context
- tag：用户标签， 具体参考 setName 接口
- sn：用户自定义的序列号，用来唯一标识该动作, 自定义 receiver 中会回执该结果

返回：

- 0：成功
- 20001：tag 数量过大(单次设置的tag数量不超过100)
- 20002：设置频率过快(频率限制每小时一次)
- 20003：标签重复
- 20004：服务初始化失败
- 20005：setTag 异常
- 20006：tag 为空
- 20007：sn为空
- 20008：离线,还未登陆成功
- 20009：该 appid 已经在黑名单列表
- 20010：已存 tag 数目超限

说明：

- 为当前用户设置一组标签，后续推送可以指定标签名进行定向推送
- 标签的设定，一定要在获取到 Clientid 之后才可以设定。标签的设定，服务端限制一天只能成功设置一次

示例：

```
String[] tags = new String[] {"tag1", "tag2", "tag3"};
Tag[] tagParam = new Tag[tags.length];

for (int i = 0; i < tags.length; i++) {
    Tag t = new Tag();
    //name 字段只支持：中文、英文字母（大小写）、数字、除英文逗号以外的其他特殊符号，具体请看代码示例
    t.setName(tags[i]);
    tagParam[i] = t;
}

int i = PushManager.getInstance().setTag(context,tagParam,
System.currentTimeMillis() +"");
String text = "设置标签失败,未知异常";

switch (i) {
    case PushConsts.SETTAG_SUCCESS:
        text = "设置标签成功";
        break;

    case PushConsts.SETTAG_ERROR_COUNT:
        text = "设置标签失败, tag数量过大, 最大不能超过200个";
        break;

    case PushConsts.SETTAG_ERROR_FREQUENCY:
        text = "设置标签失败, 频率过快, 两次间隔应大于1s";
        break;

    case PushConsts.SETTAG_ERROR_REPEAT:
        text = "设置标签失败, 标签重复";
        break;

    case PushConsts.SETTAG_ERROR_UNBIND:
        text = "设置标签失败, 服务未初始化成功";
        break;

    case PushConsts.SETTAG_ERROR_EXCEPTION:
        text = "设置标签失败, 未知异常";
        break;

    case PushConsts.SETTAG_ERROR_NULL:
        text = "设置标签失败, tag 为空";
        break;

    case PushConsts.SETTAG_NOTONLINE:
```

```
            text = "还未登陆成功";
            break;

        case PushConsts.SETTAG_IN_BLACKLIST:
            text = "该应用已经在黑名单中,请联系售后支持!";
            break;

        case PushConsts.SETTAG_NUM_EXCEED:
            text = "已存 tag 超过限制";
            break;

        default:
            break;
    }
```

9.设置静默时间

```
public boolean setSilentTime(Context context,int beginHour,int duration)
```

参数：

- context：应用的 Context
- beginHour：开始时间，设置范围在0-23小时之间，单位 h
- Duration：持续时间，设置范围在0-23小时之间。持续时间为0则不静默，单位 h

返回：

- true：设置成功
- false：设置失败

说明：

- 接口 PushManager 中的 setSilentTime，设置静默时间，静默期间SDK将不再联网

示例：

```
//设置beginHour为15，Duration为10小时，则在15:00-次日1:00这10个小时内SDK将不会联网。
int beginHour = 15;
int durationHour = 10;
boolean result = PushManager.getInstance().setSilentTime(context, beginHour, duration
Hour);
```

10.发送自定义回执

```
public boolean sendFeedbackMessage(Context context,String taskid,String messageid,int
 actionid)
```

参数：

- context：应用的 Context
- taskid：下发任务的任务ID
- messageid：下发任务的消息ID
- actionid：用户自定义的actionid，int类型，取值90001-90999

返回：

- true：上行成功
- false：上行失败；taskid为空或者 messageid 为空 或者 actionid 不在取值范围以内

说明：

- 接口 PushManager 中的 sendFeedbackMessage， 上行第三方自定义回执 actionid

11.绑定别名

```
public boolean bindAlias(Context context,String alias)
```

参数：

- context – 应用的Context
- alias – 别名名称：长度40字节，支持中、英文（区分大小写）、数字以及下划线

返回：

- true：绑定成功
- false：绑定失败

说明：

- 接口 PushManager 中的 bindAlias， 绑定别名
- 同一个别名最多绑定10个 ClientID (适用于允许多设备同时登陆的应用)，当已绑定10个 ClientID 时，再次

调用此接口会自动解绑最早绑定的记录，

- 当ClientID已绑定了别名A，若调用此接口绑定别名B，则与别名A的绑定关系会自动解除。
- 此接口与 unBindAlias 一天内最多调用100次，两次调用的间隔需大于5s。

12.解绑别名

```
public boolean unBindAlias(Context context,String alias,boolean isSelf)
```

参数：

- context：应用上下文
- alias：别名名称
- isSelf：是否只对当前 cid 有效，如果是 true，只对当前cid做解绑；如果是 false，对所有绑定该别名的cid列表做解绑

返回：

- true：解绑定成功
- false：解绑定失败

说明：

- 接口 PushManager 中的 unBindAlias， 解绑定别名
- 此接口与 bindAlias 一天内最多调用100次，两次调用的间隔需大于5s。
- 只能解绑当前手机 ClientID 与别名的关系，不能解绑其他手机上 ClientID 与别名的关系。

13.设置Tag的Name

```
public void setName(String name)
```

参数：

- name：需要传入的Tag名，只支持以下：中文、英文字母（大小写）、数字、除英文逗号以外的其他特殊符号

返回：

- 无

说明：

- 接口 Tag 中的 setName， 设置当前 Tag 的 Name 值

14.获取Clientid

```
public String getClientid(Context context)
```

参数：

- context：应用的Context

返回：

- 当前用户的cid，如果cid不存在，返回null

说明：

- 接口 PushManager 中的 getClientid， 获取当前用户的 clientid

15.获取SDK服务状态

```
public boolean isPushTurnedOn(Context context)
```

参数：

- context：应用的 Context

返回：

- true：当前推送已打开
- false：当前推送已关闭

说明：

- 接口 PushManager 中的 isPushTurnedOn， 获取当前SDK的服务状态

16.获取SDK版本号

```
public String getVersion(Context context)
```

参数：

- context：应用的 Context

返回：

- 当前SDK版本号字符串，如"2.9.5.0"

说明：

- 获取当前 SDK 版本号