**修订记录:**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **动作** | **修订内容** | **修订时间** |
| 1 | 修订 | 1.与API同步，增加标签管理与推送的功能；  2.与API同步，群推和单推返回无效id InvalidUser；  3.authToken增加修改方法；  4.修复1.0版本异常情况没有关闭连接问题； | 2019-09-20 |
| 2 | 修订 | 1.增加消息分类（classification）字段 | 2020-04-29 |
| 3 | 修订 | 1. 增加测试模式（pushMode）字段 2. 增加InvalidUser包含的status类型 | 2020-06-04 |

# 1．接入SDK

## 1.1运行环境

该SDK使用Java编写，接入前请确认是否安装Java环境，并在Java环境下运行。

## 1.2 获取SDK并导入

开发者需要注册登录开发平台网站获取应用的appId，appKey，appSecret；

在开发者网站上，下载并解压vivoPush\_sdk\_JAVA.zip；

将文件夹下所有jar文件放入项目工程的libs目录；

刷新工程，确保文件出现在libs目录下。如果没有的话请手动添加；

## 1.3 示例：给测试手机发送一条单推

集成sdk后运行该main函数，标红部分为需要使用者填写的内容。

运行成功后设备会收到推送（需提前确认设备通知栏权限已打开）

|  |
| --- |
| public static void main(String[] args) throws Exception {  Sender sender = new Sender("appSecret");//注册登录开发平台网站获取到的appSecret  Result result = sender.getToken(appId , "appKey");//注册登录开发平台网站获取到的appId和appKey  sender.setAuthToken(result. getAuthToken());  Message singleMessage = new Message.Builder()  //该测试手机设备订阅推送所得的regid，且已添加为测试设备  .regId("regId")  .notifyType(3)  .title("try\_title")  .content("try-content")  .timeToLive(1000)  .skipType(2)  .skipContent("http://www.vivo.com")  .networkType(-1)  .requestId("1234567890123456")  .pushMode(1)  .build();  Result resultMessage = sender.sendSingle(singleMessage);  System.*out*.println(resultMessage); } |

## 1.4 SDK类定义说明

|  |  |
| --- | --- |
| 类名 | 使用说明 |
| Message | 消息对象 |
| Builder | 构建要发送的Message对象 |
| TargetMessage | 构建批量推送的发送目标 |
| TagMessage | 标签相关的消息体 |
| TagGroupMessage | 标签分类的消息体 |
| TagSegMessage | 标签组合相关的消息体 |
| Sender | 发送消息工具类，可以发送鉴权、单推、批量推、全推、标签推消息 |
| TagManage | 创建标签工具类，用于创建，更新标签 |
| TagGroup | 标签分类管理工具类，用于创建，更新标签分类 |
| TagSegment | 标签组合管理工具类，用于创建，更新标签组合 |
| Result | 服务器返回的结果 |
| Validation | 对构建的消息体进行基本参数校验 |
| ExceptionStatusEnum | 消息体参数错误类型 |

# 2．发送消息

1、发送消息依赖Sender类

com.vivo.push.sdk.server.Sender

标签相关的设置依赖以下三个类

com.vivo.push.sdk.server.TagManage

com.vivo.push.sdk.server.TagGroup

com.vivo.push.sdk.server.TagSegment

2、实例化Sender

Sender(String appSecret)，（鉴权时使用）appSecret是在开发者网站上注册时生成的。

sender.setAuthToken(String authToken);，（推送时使用）除鉴权方法外，推送前都需要设置authToken，authToken是在调用鉴权方法后获得。

3、设定连接池参数(可选项)

Sender.initPool(int connection, int route) 设定连接池最大连接数为“connection”，路由最大连接数为“route”，不设定该项则使用默认参数：最大连接数 10，路由最大连接数 5

4、返回结果Result

com.vivo.push.sdk.server.Result

## 2.1鉴权

要进行PUSH推送，任何接入方都要有个鉴权操作。获得authToken用于各类推送。

限制：一天限制调用不超过10000次。

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| getToken(int appId,String appKey) | 根据appId，appKey（用户申请推送业务时生成）进行鉴权操作。 |

**Demo：**

|  |
| --- |
| public void testGetToken() throws Exception {  Sender sender = new Sender(APP\_SECRET);//实例化Sender  sender.initPool(20,10);//设置连接池参数，可选项  Result result = sender.getToken(APP\_ID,APP\_KEY);//发送鉴权请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getAuthToken();//如鉴权请求发送成功，获得authToken  } |

## 2.2单推

接入方携带消息内容以及用户regId（或alias：别名）进行通知消息推送。针对每个用户发送不同的通知。

使用场景：如物流、订单状态、游戏预约状态、行程状态、聊天（如微信、评论）等。

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| sendSingle(Message singleMessage) | 根据消息体singleMessage中设定的regId或alias发送消息到指定设备上。 |

**Demo：**

|  |
| --- |
| public void singeSend() throws Exception {  Sender sender = new Sender(APP\_SECRET);  sender.initPool(20,10);//设置连接池参数，可选项  sender.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Message singleMessage = new Message.Builder().~.build();//构建单推消息体  Result result = sender.sendSingle(singleMessage);//发送单推请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getTaskId();//如单推请求发送成功，将获得该条单推消息的任务编号，即taskId  result. getInvalidUser();//非法用户信息，包括status和userid  result. getInvalidUser().getStatus();//非法用户信息状态，包括三种情况， 1 userId不存在。 2卸载或者关闭了通知。 3 九十天不在线。4非测试用户  result. getInvalidUser().getUserid();//非法的用户信息，即接入方传的regid或者alias  } |

## 2.3批量推送

### 2.3.1保存群推消息

同一条信息覆盖多个用户。此方法需与批量推送用户方法2.3.2配套使用，批量推送用此方法返回的taskId批量发送用户。

使用场景：活动、系统升级提醒等。

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| saveListPayLoad(Message listPayLoad) | 把构建的消息体listPayLoad发送保存至服务器上，返回该消息的taskId |

**Demo：**

|  |
| --- |
| public void saveListPayload() throws Exception {  Sender sender = new Sender(APP\_SECRET);  sender.initPool(20,10);//设置连接池参数，可选项  sender.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Message saveList = new Message.Builder().~.build();//构建要保存的批量推送消息体  Result result = sender. saveListPayLoad(saveList);//发送保存群推消息请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getTaskId();//如请求发送成功，将获得该条消息的任务编号，即taskId  } |

### 2.3.2批量推送用户

批量发送给用户。该方法与保存群推消息方法2.3.1配套使用，首先通过2.3.1获得taskId,然后使用该taskId，根据regIds或者aliases进行批量推送。

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| sendToList(TargetMessage targetMessage) | 把某条消息批量发送给多个用户 |

**Demo：**

|  |
| --- |
| public void listSend() throws Exception {  Sender sender = new Sender(APP\_SECRET);  sender.initPool(20,10);//设置连接池参数，可选项  sender.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Set<String> regid = new HashSet<>();//构建批量推送用户群  regid.add(taskId1);  regid.add(taskId2);  …  TargetMessage targetMessage = new TargetMessage.Builder().~.build();//构建批量推送的消息体  Result result = sender.sendToList(targetMessage);//批量推送给用户  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result. getInvalidUsers();//非法用户信息列表，其中每个非法用户信息包括status和userid两个属性，userid为接入方传的regid或者alias，status有三种情况：1 userId不存在 2卸载或者关闭了通知 3 九十天不在线 4非测试用户  } |

## 2.4全量推送

向所有设备推送某条消息。

使用场景：活动、系统升级提醒等。

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| sendToAll(Message allMessage) | 把构建的消息allMessage发送给所有设备 |

**Demo：**

|  |
| --- |
| public void allSend() throws Exception {  Sender sender = new Sender(APP\_SECRET);  sender.initPool(20,10);//设置连接池参数，可选项  sender.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Message allSendMessage = new Message.Builder().~.build();//构建要全量推送的消息体  Result result = sender. sendToAll(allSendMessage);//发送全量推送消息请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getTaskId();//如请求发送成功，将获得该条消息的任务编号，即taskId  } |

## 2.5标签推

接入方携带消息内容、标签信息进行通知消息推送。根据标签把消息推送给指定用户

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| sendToTag(Message tagMessage) | 把构建的消息发送给标签圈定的用户 |

**Demo：**

|  |
| --- |
| public void tagSend() throws Exception {  Sender sender = new Sender(APP\_SECRET);  sender.initPool(20,10);//设置连接池参数，可选项  sender.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Message tagSendMessage = new Message.Builder().~.build();//构建要全量推送的消息体  Result result = sender. sendToTag(tagSendMessage);//发送标签推送消息请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getTaskId();//如请求发送成功，将获得该条消息的任务编号，即taskId  } |

## 2.6消息体创建

### 2.6.1通知栏消息体

com.vivo.push.sdk.server.Message.Builder

Builder方法列表

|  |  |
| --- | --- |
| 方法 | 说明 |
| Builder() | 构造方法 |
| regId(String value) | 仅单推时需设定（其它推送无需设定该项），单推可通过regId推送给指定用户。应用订阅PUSH服务器得到的id ，长度23个字符（regId，alias 两者需一个不为空，当两个不为空时，取regId） |
| alias(String value) | 仅单推时需设定（其它推送无需设定该项），单推可通过alias推送给指定用户。应用订阅PUSH服务器别名，长度不超过40字符（regId，alias 两者需一个不为空，当两个不为空时，取regId） |
| orTagss(List<String> valus) | 仅标签推送时需设定（其他推送无需设定该项），标签表达式包含三种数组： notTags表示非操作、andTags表示与操作，orTagss表示或操作，例如：  "notTags":["琴","棋","书","画"],  "andTags":["深圳","广东"],  "orTagss":["本科生","研究生"]  可以表示为，不会琴、棋、书、画的广东省深圳市的本科生或研究生的所有用户。  这些标签需要先在平台，接口，或调用该说明文档3部分的方法，创建相应标签后再使用。  具体使用方式可参考下面的DEMO和文档5.3部分 |
| andTags(List<String> valus) |
| notTags(List<String> valus) |
| title(String value) | 必填项，设置通知标题（用于通知栏消息），最大20个汉字（一个汉字等于两个英文字符，即最大不超过40个英文字符） |
| content(String value) | 必填项，设置通知内容（用于通知栏消息） 最大50个汉字（一个汉字等于两个英文字符，即最大不超过100个英文字符） |
| notifyType(int value) | 必填项，设置通知类型，value类型支持以下值：  1：无  2：响铃  3：振动  4：响铃和振动 |
| timeToLive(int value) | 可选项, 消息的生命周期, 若用户离线, 设置消息在服务器保存的时间, 单位: 秒  默认时间：1天re  最长时间：7天  最少时间：单推，60秒  其它，900秒 |
| skipType(int value) | 必填项，设置点击跳转类型，value类型支持以下值：  1：打开APP首页  2：打开链接  3：自定义  4：打开app内指定页面 |
| skipContent(String value) | 可选项，跳转内容  跳转类型为2时，跳转内容最大1000个字符，  跳转类型为3或4时，跳转内容最大1024个字符 |
| networkType(int value) | 可选项，发送推送使用的网络方式，value支持以下值：  -1：方式不限  1：仅在wifi下发送  不填默认为-1 |
| clientCustomMap(String key, String value) | 可选项，客户端自定义键值对，自定义key和value键值对个数不能超过10个，且长度不能超过1024字符, key和value键值对总长度不能超过1024字符。 |
| extra(String callback, String param) | 可选项，仅单推中使用，提供了高级特性（消息送达回执）。  callback参数：不能为null，是第三方接收回执的http接口，最大长度128个字符，vivo推送服务器将已送达或和设备对应的alias或者regId通过调用第三方设置的回调http接口传给开发者服务器。  param参数：可以为null，第三方自定义回执参数，最大长度64个字符 |
| requestId(String value) | 必填项，用户请求唯一标识 最大64字符 |
| classification(int value) | 消息类型 0：运营类消息，1：系统类消息 |
| pushMode(int value) | 推送模式 0：正式推送；1：测试推送，不填默认为0（测试推送，只能给web界面录入的测试用户推送；审核中应用，只能用测试推送） |
| build() | 根据设置的属性, 生成Message对象 |

**Demo：**

|  |
| --- |
| public Message buildMessage() throws Exception {  List<String> andTags = new ArrayList<>();  andTags.add("TAG1");  List<String> orTagss = new ArrayList<>();  orTags.add("TAG2");  List<String> notTags = new ArrayList<>();  notTags.add("TAG3");  Message message = new Message.Builder()  .regId("12345678901234567890123")//仅构建单推消息体需要  .alias(ALIAS) //仅构建单推消息体需要  .orTagss(orTagss) //仅构建标签推消息体需要  .andTags(andTags)//仅构建标签推消息体需要  .notTags(notTags) //仅构建标签推消息体需要  .notifyType(1)  .title("YOUR\_TITLE")  .content("YOUR\_CONTENT")  .timeToLive(1000)  .skipType(2)  .skipContent("http://www.vivo.com")  .networkType(-1)  .clientCustomMap("key1", "value1")  .extra("http://www.vivo.com", "vivo")  .requestId("1234567890123456")  .classification(1)  .build();  Return message;  } |

### 2.6.2批量推送用户消息体

com.vivo.push.sdk.server.TargetMessage.Builder

Builder方法列表

|  |  |
| --- | --- |
| 方法 | 说明 |
| Builder() | 构造方法 |
| regIds(Set<String> regIds) | regId列表，个数大于等于2，小于等于1000，regId长度23个字符（regIds，aliases 两者需一个不为空，两个都不为空时，取regIds） |
| aliases(Set<String> aliases) | 别名列表，个数大于等于2，小于等于1000，长度不超过40字符（regIds，aliases 两者需一个不为空，两个都不为空时，取regIds） |
| taskId(String taskId) | 必填项，公共消息任务号，即调用com.vivo.push.sdk.server.Sender.saveListPayLoad返回的taskId |
| requestId(String requestId) | 必填项，用户请求唯一标识 最大64字符 |
| pushMode(int value) | 推送模式 0：正式推送；1：测试推送，不填默认为0（测试推送，只能给web界面录入的测试用户推送；审核中应用，只能用测试推送） |
| build() | 根据设置的属性, 生成TargetMessage对象 |

**Demo：**

|  |
| --- |
| public TargetMessage buildTargetMessage() throws Exception {  Set<String> regids = new HashSet<>();  regids.add("12345678901234567890123");  regids.add("12345678901234567890321");  Set<String> aliases = new HashSet<>();  aliases.add("ALIAS1");  aliases.add("ALIAS2");  TargetMessage targetMessage = new TargetMessage.Builder()  .regIds(regids)  .aliases(aliases)  .requestId("1234567890123456")  .taskId("123456789012345678")  .build();  Return targetMessage;  } |

## 2.7获取消息推送的统计值

获取批量推送或全量推送返回的taskId对应的统计信息，单次查询的taskIds最多100个。

**接口说明：**

com.vivo.push.sdk.server.Sender

|  |  |
| --- | --- |
| 方法 | 说明 |
| getStatistics(Set<String> taskIds) | 查询taskIds里面所有taskId对应的统计信息，taskIds里最多存放100个taskId |

**Demo：**

|  |
| --- |
| public void testGetStatistics() throws IOException {  Sender sender = new Sender(APP\_SECRET,authToken);  sender.initPool(20,10);//设置连接池参数，可选项  Set<String> taskIds = new HashSet<>();  taskIds.add("123456789012345678");  taskIds.add("123456789087654321");  Result result = sender.getStatistics(taskIds);  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result. getStatistics ();//如请求发送成功，将获得各个taskId对应的统计数据  } |

# 3．标签管理

## 3.1新增标签

为应用方增加标签，用于后续标签推送。

**接口说明：**

com.vivo.push.sdk.server.TagMange

|  |  |
| --- | --- |
| 方法 | 说明 |
| addTag(TagMessage tagMessage) | 根据消息体tagMessage中设定的相关信息创建新标签。 |

**Demo：**

|  |
| --- |
| public void tagAdd() throws Exception {  TagManage tagManage = new TagManage (APP\_SECRET);  tagManage.initPool(20,10);//设置连接池参数，可选项  tagManage.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  TagMessage tagMessage = new TagMessage.Builder()  .name("TAG\_NAME") //必填项，标签名称  .desc("TAG\_DESCRIPTION")//可选项，标签描述  .group("GROUP\_NAME")//可选项，标签分类名称（参考4部分）  .build(); //构建创建标签的消息体  Result result = tagManage.addTag(tagMessage);//发送创建标签请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

## 3.2更新单个标签

更新一个标签的信息,更新标签名字、描述、所属分类。

**接口说明：**

com.vivo.push.sdk.server.TagMange

|  |  |
| --- | --- |
| 方法 | 说明 |
| updateTag (TagMessage tagMessage) | 根据消息体tagMessage中设定的相关信息更新已存在的标签信息。 |

**Demo：**

|  |
| --- |
| public void tagUpdate() throws Exception {  TagManage tagManage = new TagManage (APP\_SECRET);  tagManage.initPool(20,10);//设置连接池参数，可选项  tagManage.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  TagMessage tagMessage = new TagMessage.Builder()  .oldName("OLD\_NAME")//必填项，旧的标签名称  .newName("NEW\_NAME")//必填项，新的标签名称  .desc("TAG\_DESCRIPTION")//可选项，更新标签描述  .group("GROUP\_NAME")//可选项，更新标签分类（参考4部分）  .build(); //构建更新标签的消息体  Result result = tagManage.updateTag(tagMessage);//发送更新标签请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

## 3.3给标签添加用户设备

给标签添加用户设备信息，单次不超过1000个设备。

**接口说明：**

com.vivo.push.sdk.server.TagMange

|  |  |
| --- | --- |
| 方法 | 说明 |
| addMembers(TagMessage tagMessage) | 把tagMessage中指定的用户设备添加到指定的标签中。 |

**Demo：**

|  |
| --- |
| public void tagAddMembers () throws Exception {  TagManage tagManage = new TagManage (APP\_SECRET);  tagManage.initPool(20,10);//设置连接池参数，可选项  tagManage.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Set<String> ids = new HashSet<>();  ids.add("REGID1");  ids.add("REGID2");  TagMessage tagMessage = new TagMessage.Builder()  .name("TAG\_NAME")//必填项，要添加用户设备的标签名称  .type(1)// 必填项，用户类型：1是regId ,2是别名  .ids(ids)//必填项，用户id  .build();//构建给标签添加用户设备的消息体  Result result = tagManage.addMembers (tagMessage);//发送给标签添加用户设备的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getData();//非法用户信息列表，其中每个非法用户信息包括status和userid两个属性，userid为接入方传的regid或者alias，status有三种情况：1是用户不存在，2是用户push开关关闭，3是用户7天未联网  } |

## 3.4移除标签中的用户设备

移除标签中的用户设备信息，单次不超过1000个设备。

**接口说明：**

com.vivo.push.sdk.server.TagMange

|  |  |
| --- | --- |
| 方法 | 说明 |
| removeMembers(TagMessage tagMessage) | 把tagMessage中指定的用户设备从指定标签中移除。 |

**Demo：**

|  |
| --- |
| public void tagRemoveMembers () throws Exception {  TagManage tagManage = new TagManage (APP\_SECRET);  tagManage.initPool(20,10);//设置连接池参数，可选项  tagManage.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  Set<String> ids = new HashSet<>();  ids.add("REGID1");  ids.add("REGID2");  TagMessage tagMessage = new TagMessage.Builder()  .name("TAG\_NAME")//必填项，要移除用户设备的标签名称  .type(1)// 必填项，用户类型：1是regId ,2是别名  .ids(ids)//必填项，用户id  .build();//构建移除标签用户设备的消息体  Result result = tagManage. addMembers (tagMessage);//发送移除标签中的指定用户设备的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  result.getData();//非法用户信息列表，其中每个非法用户信息包括status和userid两个属性，userid为接入方传的regid或者alias，status有三种情况：1是用户不存在，2是用户push开关关闭，3是用户7天未联网  } |

# 4．标签分类管理

## 4.1新增标签分类

为应用方增加标签分类，用于后续标签分类管理。

**接口说明：**

com.vivo.push.sdk.server.TagGroup

|  |  |
| --- | --- |
| 方法 | 说明 |
| addTagGroup(TagGroupMessage groupMessage) | 根据消息体groupMessage中设定的相关信息创建新标签分类。 |

**Demo：**

|  |
| --- |
| public void addGroup() throws Exception {  TagGroup tagGroup = new TagGroup (APP\_SECRET);  tagGroup.initPool(20,10)*;*//设置连接池参数，可选项  tagGroup.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  TagGroupMessage groupMessage = new TagGroupMessage.Builder()  .name("GROUP\_NAME")//必填项，标签分类名称  .type(1)//必填项，标签分类类型：1是普通标签，2是互斥标签  .desc("one group")//可选项，标签分类的描述信息  .build();//构建新增标签分类的消息体  Result result = tagGroup.addTagGroup (groupMessage);//发送创建标签分类的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

## 4.2更新单个标签分类

更新一个标签分类的信息，包括标签分类名称和描述。

**接口说明：**

com.vivo.push.sdk.server.TagGroup

|  |  |
| --- | --- |
| 方法 | 说明 |
| updateTag(TagGroupMessage groupMessage) | 根据消息体groupMessag中设定的相关信息更新已存在的标签分类信息。 |

**Demo：**

|  |
| --- |
| public void updateGroup() throws Exception {  TagGroup tagGroup = new TagGroup (APP\_SECRET);  tagGroup.initPool(20,10)*;*//设置连接池参数，可选项  tagGroup.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  TagGroupMessage groupMessage = new TagGroupMessage.Builder()  .oldName("OLD\_NAME")//必填项，旧的标签分类名称  .newName("NEW\_NAME")//必填项，新的标签分类名称  .desc("TAG\_DESCRIPTION")//可选项，更新标签分类的描述  .build(); //构建更新标签分类的消息体  Result result = tagGroup.updateTagGroup (groupMessage);//发送更新标签分类的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

## 4.3添加标签到标签分类中

把标签添加进标签分类里面，一次可添加一个或者多个标签，最多不超过100个。

**接口说明：**

com.vivo.push.sdk.server.TagGroup

|  |  |
| --- | --- |
| 方法 | 说明 |
| addTagToGroup(TagGroupMessage groupMessage) | 把groupMessag中指定的标签添加到指定的标签分类中。 |

**Demo：**

|  |
| --- |
| public void tagsToGroup() throws Exception {  TagGroup tagGroup = new TagGroup (APP\_SECRET);  tagGroup.initPool(20,10);//设置连接池参数，可选项  tagGroup.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  List<String> tagList = new ArrayList<>();  tagList.add("TAG1");  tagList.add("TAG2");  TagGroupMessage groupMessage = new TagGroupMessage.Builder()  .name("TAG\_GROUP\_NAME")//必填项，需添加标签的标签分类名称  .tagList(tagList)//必填项，标签列表  .build(); //构建添加标签到标签分类的消息体  Result result = tagGroup.addTagToGroup (groupMessage);//发送添加标签到标签分类的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

# 5．标签组合管理

## 5.1新增标签组合

为应用方增加标签组合。 一个标签组合由多个标签组成。

**接口说明：**

com.vivo.push.sdk.server.TagSegment

|  |  |
| --- | --- |
| 方法 | 说明 |
| addTagSegment(TagSegMessage segMessage) | 根据消息体segMessage中设定的相关信息创建新标签组合。 |

**Demo：**

|  |
| --- |
| public void addSegment() throws Exception {  TagSegment tagSegment = new TagSegment (APP\_SECRET);  tagSegment.initPool(20,10)*;*//设置连接池参数，可选项  tagSegment.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  List<String> andTags = new ArrayList<>();  andTags.add("TAG1");  List<String> orTagss = new ArrayList<>();  orTags.add("TAG2");  List<String> notTags = new ArrayList<>();  notTags.add("TAG3");  TagSegMessage segMessage = new TagSegMessage.Builder()  .name("SEGMENT\_NAME")//必填项，标签组合的名称  .andTags(andTags)//与操作  .orTagss(orTags)//或操作  .notTags(notTags)//非操作（具体与，或，非含义，操作参考5.3）  .build();//构建新增标签组合的消息体  Result result = tagSegment.addTagSegment (segMessage);//发送创建标签组合的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

## 5.2更新标签组合

更新一个标签组合的信息。

**接口说明：**

com.vivo.push.sdk.server.TagSegment

|  |  |
| --- | --- |
| 方法 | 说明 |
| addTagSegment(TagSegMessage segMessage) | 根据消息体segMessage中设定的相关信息创建新标签组合。 |

**Demo：**

|  |
| --- |
| public void updateSegment() throws Exception {  TagSegment tagSegment = new TagSegment (APP\_SECRET);  tagSegment.initPool(20,10)*;*//设置连接池参数，可选项  tagSegment.setAuthToken(authToken);//设置推送的必要参数authToken(调用鉴权方法获得)  List<String> andTags = new ArrayList<>();  andTags.add("TAG1");  List<String> orTagss = new ArrayList<>();  orTagss.add("TAG2");  List<String> notTags = new ArrayList<>();  notTags.add("TAG3");  TagSegMessage segMessage = new TagSegMessage.Builder()  .oldName("OLD\_NAME")//必填项，旧的标签组合名称  .newName("NEW\_NAME")//必填项，新的标签组合名称  .andTags(andTags)//与操作  .orTagss(orTags)//或操作  .notTags(notTags)//非操作（具体与，或，非含义，操作参考5.3）  .build();//构建新增标签组合的消息体  Result result = tagSegment.updateTagSegment (segMessage);//发送更新标签组合的请求  result.getResult();//获取服务器返回的状态码，0成功，非0失败  result.getDesc();//获取服务器返回的调用情况文字描述  } |

## 5.3标签组合表达式

|  |
| --- |
| List<String> andTags = new ArrayList<>();  andTags.add("深圳","广东");  List<String> orTagss = new ArrayList<>();  orTagss.add("本科生","研究生");  List<String> notTags = new ArrayList<>();  notTags.add("琴","棋","书","画");  TagSegMessage segMessage = new TagSegMessage.Builder()  .name("SEGMENT\_NAME")//必填项，标签组合的名称  .andTags(andTags)//与操作  .orTagss(orTags)//或操作  .notTags(notTags)//非操作  .build(); |
| 标签表达式包含三种数组： notTags表示非操作、andTags表示与操作，orTagss表示或操作，例如如上面所示创建标签表达式，即代表：  "notTags":["琴","棋","书","画"],  "andTags":["深圳","广东"],  "orTagss":["本科生","研究生"]  可以表示为，不会琴、棋、书、画的广东省深圳市的本科生或研究生的所有用户。  这些标签需要先在平台，接口，或调用该说明文档3部分的方法，创建相应标签后再使用。 |