移动推送

API 文档

产品文档

腾讯云

文档目录

API 文档

简介

最近更新时间：2024-01-17 14:23:45

REST API 是移动推送提供给 App 后台的 HTTP 管理接口，供开发者远程调用推送服务。
REST API 支持 HTTP 和 HTTPS 协议，为了提高安全性，建议使用 HTTPS 协议。

前提条件

如需调用 REST API，请确保已在移动推送控制台创建应用，详情请参见 创建产品和应用。

API 概览

最近更新时间：2024-01-17 14:23:45

推送相关接口

接口名称	接口功能
v3/push/app	推送通知与应用内消息接口
v3/push/package/upload	号码包上传接口 token包上传接口
v3/push/plan/add_plan_push	创建推送计划接口

标签相关接口

接口名称	接口功能
v3/device/tag	标签绑定与解绑接口
v3/device/tag/delete_all_device	删除标签下所有设备接口

账号相关接口

接口名称	接口功能
v3/device/account/batchoperate	账号绑定与解绑接口
v3/device/account/query	账号设备绑定查询接口

统计相关接口

接口名称	接口功能
v3/statistics/get_push_stat_overview	推送统计按天聚合统计
v3/statistics/get_device_stat_overview	设备统计查询

v3/statistics/get_push_record	特定时间段所有推送信息查询
v3/statistics/get_push_record	单个任务推送信息查询
v3/statistics/get_push_task_stat_channel	单条任务推送统计信息查询
v3/statistics/get_push_group_stat	多条任务统计数据聚合查询

用户属性相关接口

接口名称	接口功能
v3/device/set_custom_attribute	用户属性相关接口，用于 token 级别的个性化属性配置，包括增加、删除、更新、查询功能。

调用方式
签名认证（推荐）

最近更新时间：2024-01-17 14:24:51

概述

本文主要为您介绍移动推送签名认证方法。
采用 HMAC-SHA256 算法，根据 SecretKey 生产签名信息。通过校验签名进行鉴权，安全性更好，推荐使用。

参数说明

参数	说明
AccessId	移动推送后台分配的应用 ID，请前往 移动推送控制台 > **配置管理** > **基础配置**获取
SecretKey	移动推送后台分配的 SecretKey，与 AccessId 对应，请前往 移动推送控制台 > **配置管理** > **基础配置**获取
Sign	接口签名方式
TimeStamp	请求时间戳，单位s

签名生成方式

1. 通过请求时间戳 + AccessId + 请求 body 进行字符拼接，得到原始的待签名字符串：
`待签名字符串 = ${TimeStamp} + ${AccessId} + ${请求body}`
2. 通过 SecretKey 作为密钥，对原始待签名字符串进行签名，生成得到签名：
`Sign = Base64(HMAC_SHA256(SecretKey, 待签名字符串))`

HTTP 协议拼装方式

HTTP 协议 header 中 除了通用头部协议外，需要携带当前请求时间戳、 AccessId、 以及签名 Sign 信息，具体参数如下：

Header 中参数 Key	含义	是否必须
Sign	请求签名	是

AccessId	应用 ID	是
TimeStamp	请求时间戳	是

具体 HTTP 请求报文如下：

```
POST /v3/push/app HTTP/1.1
Host: api.tpns.tencent.com
Content-Type: application/json
AccessId: 1500001048
TimeStamp: 1565314789
Sign: Y2QyMDc3NDY4MmJmNzhiZmRiNDNlMTdkMWQ1ZDU2YjNlNWI3ODlhMTY3MGZjMTUyN2VmNTRjNjVkM
{"audience_type": "account","message": {"title": "test title","content": "test cont
```

签名生成示例

1. 生成待拼接签名字符串如下：

```
待加密字符串=1565314789150000104 8{"audience_type": "account","message": {"title":
"test title","content": "test content","android": { "action": {"action_type":
3,"intent": "xgscheme://com.xg.push/notify_detail?param1=xg"}}},"message_type":
"notify","account_list": ["5822f0eee44c3625ef0000bb"] }
```

说明：

待签名字符串中的 `${请求body}` 和发送数据必须保持一致，包括空格、数据编码等。

2. 根据密钥通过 HMAC-SHA256 算法，生成十六进制 hash，其中示例对应 `secretKey =1452fcebae9f3115ba794fb0fff2fd73` 。

```
hashcode= hmac-sha256(SecretKey, 待签名字符串)
得到 hashcode="09f07d2a518a8814e369dcd9534f10b8a29d128531a19adaa28cb247605c1e84"
```

3. 对 hashcode 进行 base64 编码，得到签名串如下：

```
得到 Sign=Base64(hashcode)
Sign="MDlmMDdkMmE1MThhODgxNGUzNjlkY2Q5NTM0ZjEwYjhhMjlkMTI4NTMxYTE5YWRhYTI4Y2IyN
Dc2MDVjMWU4NA=="
```

各语言签名代码示例

Python2

Python3

Java

Golang

C#

PHP

```
#!/usr/bin/env python
import hmac
import base64
from hashlib import sha256

s = '15653147891500001048{"audience_type": "account","message": {"title": "test tit
key = '1452fcebae9f3115ba794fb0fff2fd73'
hashcode = hmac.new(key, s, digestmod=sha256).hexdigest()
print base64.b64encode(hashcode)
```

```
import hmac
import base64
from hashlib import sha256

s = '15653147891500001048{"audience_type": "account","message": {"title": "test tit
key = '1452fcebae9f3115ba794fb0fff2fd73'
hashcode = hmac.new(bytes(key, "utf-8"), bytes(s, "utf-8"),
                        digestmod=sha256).hexdigest()
print(base64.b64encode(bytes(hashcode, "utf-8")))
```

```
package com.tencent.xg;

import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.codec.binary.Hex;

public class SignTest {
    public static void main(String[] args) {
        try {
            String stringToSign = "15653147891500001048{\\"audience_type\\": \\"acc
            String appSecret = "1452fcebae9f3115ba794fb0fff2fd73";

            Mac mac;
```

```
            mac = Mac.getInstance("HmacSHA256");
            mac.init(new SecretKeySpec(appSecret.getBytes("UTF-8"), "HmacSHA256"));
            byte[] signatureBytes = mac.doFinal(stringToSign.getBytes("UTF-8"));

            String hexStr = Hex.encodeHexString(signatureBytes);
            String signature = Base64.encodeBase64String(hexStr.getBytes());

            System.out.println(signature);
        } catch (NoSuchAlgorithmException | InvalidKeyException | UnsupportedEncodi
            e.printStackTrace();
        }
    }
}
```

```
import (
   "crypto/hmac"
   "crypto/sha256"
   "encoding/base64"
   "encoding/hex"
   "testing"
)

func TestSign(t *testing.T) {
   requestBody := "156531478915000010 48{\\"audience_type\\": \\"account\\",\\"messa
   secretKey := "1452fcebae9f3115ba794fb0fff2fd73"

   h := hmac.New(sha256.New, []byte(secretKey))
   h.Write([]byte(requestBody))
   sha := hex.EncodeToString(h.Sum(nil))
   sign := base64.StdEncoding.EncodeToString([]byte(sha))
   println(sign)
}
```

```
using System;
using System.Security.Cryptography;
using System.Text;
namespace tpns_server_sdk_cs
{
    class GenSign {

        //Main Method
        static public void Main(String[] args)
        {
            string reqBody =
```

```
                "{\\"audience_type\\": \\"account\\",\\"message\\": {\\"title\\": \
            string genSign = GenSign.genSign("1621307510", "1500004469", reqBody, "
            Console.WriteLine(genSign);
        }
        public static string HmacSHA256(string key, string data)
        {
            string hash;
            Byte[] code = System.Text.Encoding.UTF8.GetBytes(key);
            using (HMACSHA256 hmac = new HMACSHA256(code))
            {
                Byte[] d = System.Text.Encoding.UTF8.GetBytes(data);
                //Byte[] hmBytes = hmac.ComputeHash(encoder.GetBytes(data));
                Byte[] hmBytes = hmac.ComputeHash(d);
                hash = ToHexString(hmBytes);
            }
            return hash;
        }
        public static string ToHexString(byte[] array)
        {
            StringBuilder hex = new StringBuilder(array.Length * 2);
            foreach (byte b in array)
            {
                hex.AppendFormat("{0:x2}", b);
            }
            return hex.ToString();
        }

        public static string genSign(string timeStampStr, string accessId, string r
        {
            string data = timeStampStr + accessId + requestBody;
            string hash = HmacSHA256(keySecret, data);
            string sign = Base64Encode(hash);
            Console.WriteLine("timeStampStr: "  + timeStampStr + " accessId:" + acc
            return sign;
        }

        public static string Base64Encode(string plainText) {
            var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(plainText);
            return System.Convert.ToBase64String(plainTextBytes);
        }
    }
}


```

```
<?php
```

```
$accessId = "1500001048";
$secretKey = "1452fcebae9f3115ba794fb0fff2fd73";
$timeStamp = "1565314789";
$requestBody = "{\\"audience_type\\": \\"account\\",\\"message\\": {\\"title\\": \\
$hashData = "{$timeStamp}{$accessId}{$requestBody}";
echo "reqBody: " . $hashData . "\\n";
//获取 sha256 and hex 结果
$hashRes = hash_hmac("sha256", $hashData, $secretKey, false);
//进行 base64
$sign = base64_encode($hashRes);
echo $sign . "\\n";
?>
```

Basic Auth 认证

最近更新时间：2024-01-17 14:24:51

本文主要介绍移动推送 Basic Auth 的鉴权认证方法。
采用 AccessId 和 SecretKey 进行 Basic Auth 认证鉴权，密钥容易被获取，安全性不高，推荐使用 签名认证。

获取密钥

1. 登录 移动推送控制台，选择左侧菜单**配置管理 > 基础配置**。
2. 在应用信息一栏中，获取应用 `Access ID` 和 `SECRET KEY` 。

生成签名

1. 拼接请求字符串
此步骤生成请求字符串。
生成算法为： `base64(Access ID:SECRETKEY)` ，即对 `Access ID` 加上冒号，加上 `SECRETKEY` 拼装起来的字符串，再做 `base64` 转换。
示例如下：

```
base64(150000****:cf43dac624820*****c1fe5fc993)
```

2. 进行 Basic Auth 认证鉴权
采用基础鉴权的方式，在 HTTP Header（头）里加一个字段（ Key/Value 对）：

```
Authorization: Basic base64_auth_string
```

示例如下：

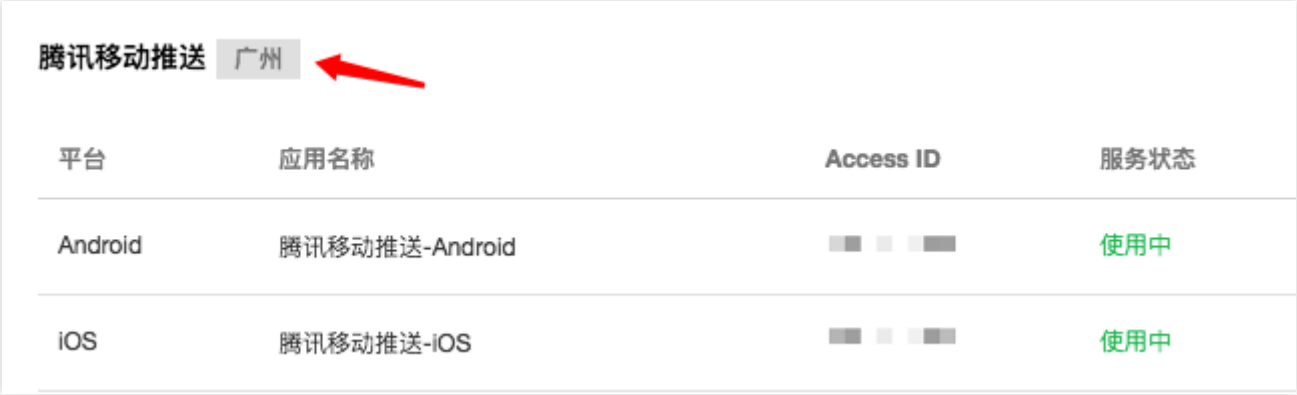
```
Authorization:Basic base64(150000****:cf43dac624820*****c1fe5fc993)
```

请求结构

最近更新时间：2024-01-17 14:24:51

服务地址

移动推送目前部署有广州、上海、中国香港、新加坡（亚太东南）四个服务接入点，不同服务接入点之间数据完全隔离，您可根据创建产品时选择的「服务接入点」选择服务地址。



目前支持的服务地址列表如下：

服务接入点名称	服务地址
广州服务接入点	https://api.tpns.tencent.com/
上海服务接入点	https://api.tpns.sh.tencent.com/
中国香港服务接入点	https://api.tpns.hk.tencent.com/
新加坡服务接入点	https://api.tpns.sgp.tencent.com/

通信协议

腾讯云 API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

请求方法

支持的 HTTP 请求方法：POST

POST 请求支持的 Content-Type 类型：

1. application/json。
2. multipart/form-data（仅部分接口支持）。

字符编码

均使用 `UTF-8` 编码。

推送相关接口
推送接口

最近更新时间：2024-01-25 11:53:41

接口说明

请求方式：POST。

```
服务地址/v3/push/app
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：Push API 是所有推送接口的统称。Push API 有多种推送目标，推送目标见下文。

所有请求参数通过 JSON 封装上传给后台，后台通过请求参数区分不同的推送目标。如有疑问请参见 服务端错误码。

注意：

腾讯云移动推送结合业务优化的需求，对**全量推送、标签推送和号码包推送**的频率进行限制，均调整为 1条/秒，官网控制台将与 API 同步调整。

如果超过此频率可能会引起推送异常，如相关推送需更高频率，您可以联系 在线客服。

必要参数

推送必要参数指一条推送消息必需携带的参数。

参数名	类型	是否必需	描述
audience_type	String	是	推送目标： all：全量推送 tag：标签推送 token：单设备推送 token_list：设备列表推送 account：单账号推送 account_list：账号列表推送 package_account_push：号码包推送 package_token_push：token 文件包推送
message	Object	是	消息体，请参见 消息体类型
message_type	String	是	消息类型：

			notify：通知 message：透传消息/静默消息
environment	String	是（仅 iOS 平台使用）	用户指定推送环境，仅限 iOS 平台推送使用： product： 推送生产环境 dev： 推送开发环境 注册打包的环境与推送环境需要保存一致，请参见推送环境选择说明
upload_id	Integer	是（仅号码包推送\\token 文件包推送时使用）	号码包或 token 包的上传 ID

audience_type：推送目标

推送目标，表示一条推送可以被推送到哪些设备。

Push API 提供了多种推送目标的，例如全量、标签、单设备、设备列表、单账号、账号列表。

推送目标	描述	必需参数及使用说明
all	全量推送	无
tag	标签推送	tag_rules（推荐使用）： 标签组合推送，可设置'与'、'或'、'非'组合规则 **注意：**当与 tag_list 二者共存时，tag_list 字段自动无效，参数说明请查看tag_rules 参数说明 tag_list（后续不再更新）： 推送 tag1 和 tag2 的设备{"tags":["tag1","tag2"],"op":"AND"} 推送 tag1 或 tag2 的设备{"tags":["tag1","tag2"],"op":"OR"} 标签列表不能超过512个字符
token	单设备推送	token_list 如果该参数包含多个 token 只会推送第一个 token 格式 eg：[“token1”] token 字符串长度不能超过36
token_list	设备列表群推	token_list 最多1000个 token 格式 eg：["token1","token2"] token 字符串长度不能超过36 **注意：**若列表超过1000个 token，推送会失败，如需推送更大批量 token，建议您使用上传 Token 包推送
account	单账号推送	account_list 该参数有多个账号时，仅推送第一个账号 格式 eg：["account1"]

account_list	账号列表群推	account_list 最多1000个 account 格式 eg：["account1","account2"] **注意**：若列表超过1000个 account，推送会失败，如需推送更大批量 account，建议您使用号码包推送
package_account_push	号码包推送	上传号码包推送必填
package_token_push	token 文件包推送	上传 token 文件包推送必填

全量推送：推送给全部设备。

```
{
  "audience_type": "all"
}
```

标签推送（tag_rules 方式）：广东和湖南，并且是20200408当天活跃过的男性用户。

```
 {
"audience_type": "tag",
"tag_rules": [
      {
          "tag_items": [
              {
                  "tags": [
                      "guangdong",
                      "hunan"
                  ],
                  "is_not": false,
                  "tags_operator": "OR",
                  "items_operator": "OR",
                  "tag_type": "xg_auto_province"
              },
              {
                  "tags": [
                      "20200408"
                  ],
                  "is_not": false,
                  "tags_operator": "OR",
                  "items_operator": "AND",
                  "tag_type": "xg_auto_active"
              },
              {
                  "tags": [
                      "male"
```

```
                    ],
                    "is_not": false,
                    "tags_operator": "OR",
                    "items_operator": "AND",
                    "tag_type": "xg_user_define"
                }
            ],
            "operator": "OR",
            "is_not": false
        }
    ]
}
```

单设备推送：推送给 token 为 token1 的设备。

```
{
  "audience_type": "token",
  "token_list": [
      "token1"
  ]
}
```

设备列表推送，推送给 token 为 token1 和 token2 的设备。

```
{
  "audience_type": "token_list",
  "token_list": [
      "token1",
      "token2"
  ]
}
```

单账号推送：推送给账号为 account1 的设备。

```
{
  "audience_type": "account",
  "account_list": [
      "account1"
  ]
}
```

账号列表推送：推送账号为 account1 和 account2 的设备。

```
{
  "audience_type": "account_list",
  "account_list": [
      "account1",
      "account2"
```

```
    ]
}
```

message_type：消息类型

针对不同平台，消息类型稍有不同，具体参照下表：

消息类型	描述	支持平台	特性说明
notify	通知栏消息	Android & iOS	通知栏展示消息 **注意**：该字段与 content-available: 1 互斥，请勿同时使用。
message	透传消息/静默消息	Android（透传消息） iOS（静默消息）	通知栏不展示消息。 **注意**：因厂商限制，Android 透传消息只通过移动推送自建通道下发，无法通过厂商通道下发

message：消息体

消息体，即下发到客户端的消息。

Push API 对 iOS 和 Android 两个平台的消息有不同处理，需要分开来实现对应平台的推送消息，推送的消息体是 JSON 格式。

Android 普通消息

Android 平台具体字段如下表：

字段名	类型	父项目	默认值	必需	参数描述
title	String	message	无	是	消息标题
content	String	message	无	是	消息内容
accept_time	Array	message	无	否	消息将在哪些时间段允许推送给用户。 单个元素由 "start" 和 "end" 组成的起止时间对组成 "start" 和 "end" 由 hour （小时）和 min（分钟）描述对应时刻，详细参考具体示例。 **注意**：因厂商限制， 仅对移动推送自建通道有效
thread_id	String	message	无	否	通知分组折叠的组别识别名 **注意**：因厂商限制， 仅对移动推送自建通道有效

thread_sumtext	String	message	无	否	通知分组折叠后显示的摘要，thread_id 非空时有效 **注意：**因厂商限制，仅对移动推送自建通道有效
xg_media_resources	String	message	无	否	通知栏大图片 url 地址，仅对移动推送自建通道和小米通道生效； **注意：**如需使用小米通道大图通知功能，需先调用小米图片上传接口上传图片文件，获取小米指定的图片地址 pic_url，再填入移动推送推送对应的参数 xg_media_resources 中。
xg_media_audio_resources	String	message	无	否	音频富媒体元素地址。 支持 mp3 格式音频，建议大小在5MB以内。 **注意：**仅移动推送自建通道支持该参数下发，其他通道不下发该参数
android	Object	message	无	否	安卓通知高级设置结构体，请参见 Android 结构体说明

Android 结构体说明

字段名	类型	父项目	默认值	必需	参数描述
n_ch_id	String	android	无	否	通知渠道 ID（仅移动推送自建通道生效），请参见 创建通知渠道
n_ch_name	String	android	无	否	通知渠道名称（仅移动推送自建通道生效），请参见 创建通知渠道
xm_ch_id	String	android	无	否	小米渠道 ID（仅小米推送通道生效）
fcm_ch_id	String	android	无	否	FCM 渠道 ID（仅 FCM 推送通道生效）
hw_biz_type	Integer	android	0	否	是否开启华为快通知： 1：开启 0：关闭 **注意**：仅华为通道有效，其需要联系华为商务开通。
hw_ch_id	String	android	无	否	华为渠道 ID（仅 华为推送通道生效）
hw_category	String	android	无	否	华为消息类型标识，确定消息提醒方式，对特定类型消息加快发送，参数详情请参见 华为的请求参数

					说明 的 category 参数。 IM：即时聊天 VOIP：音视频通话 SUBSCRIPTION：订阅
hw_importance	Integer	android	无	否	消息的提醒级别，取值如下： 1：表示通知栏消息预期的提醒方式为静默提醒，消息到达手机后，无铃声震动 2：表示通知栏消息预期的提醒方式为强提醒，消息到达手机后，以铃声、震动提醒用户。终端设备实际消息提醒方式将根据 hw_category 字段取值或者 智能分类 结果进行调整。
oppo_ch_id	String	android	无	否	OPPO渠道 ID（仅 OPPO 推送通道生效）
vivo_ch_id	String	android	0	否	vivo 渠道 ID：“0”代表运营消息，“1”代表系统消息（仅 vivo 推送通道生效）
n_id	Integer	android	0	否	（该字段已废弃，后续会下线，如需使用覆盖功能请使用覆盖参数：collapse_id） 通知消息对象的唯一标识（移动推送自建通道） （1）大于0：会覆盖先前相同 id 的消息 （2）等于0：展示本条通知且不影响其他消息 （3）等于-1：将清除先前所有消息，仅展示本条消息
builder_id	Integer	android	0	否	本地通知样式标识
badge_type	Integer	android	-1	否	通知角标： -2：自动增加1，支持华为设备 -1：不变，支持华为、vivo 设备 [0, 100)：直接设置，支持华为、vivo 设备 **注意**：不同厂商设备的角标适配能力不同，各参数值实现效果请参见角标适配指南
ring	Integer	android	1	否	是否有铃声： 0：没有铃声 1：有铃声
ring_raw	String	android	无	否	指定 Android 工程里 raw 目录中的铃声文件名，不需要后缀名。 **说明**：自定义铃声仅华为、小米、FCM 和移动推送自建通道支持，需配合n_ch_id字段使用，配置步骤可参考 如何设置自定义铃声。
vibrate	Integer	android	1	否	是否使用震动： 0：没有震动

					1：有震动
lights	Integer	android	1	否	是否使用呼吸灯： 0：不使用呼吸灯 1：使用呼吸灯
clearable	Integer	android	1	否	通知栏是否可清除
icon_type	Integer	android	0	否	指定通知栏缩略图显示的是应用内图标还是网络资源图标 ： 0：应用内图标，仅对移动推送自建通道有效 1：网络资源图标，仅移动推送自建通道、FCM、华为、荣耀通道支持
icon_res	String	android	无	否	指定通知栏缩略图的具体图片资源 ： 当 icon_type = 0：填写 Android 应用内的图片资源文件名称（不带文件后缀），仅对移动推送自建通道有效 当前 icon_type = 1：填写缩略图 url 地址，缩略图格式要求可参见 富媒体通知文档，仅移动推送自建通道、FCM、华为、荣耀通道支持
style_id	Integer	android	1	否	设置是否覆盖指定编号的通知样式
small_icon	String	android	无	否	消息在状态栏显示的图标，若不设置，则显示应用图标
icon_color	Integer	android	0	否	通知栏小图标染色 仅移动推送自建通道有效 需要使用 RGB 颜色的十进制值，例如 RGB 颜色 #01e240，请填入123456
action	Object	android	有	否	设置点击通知栏之后的行为，默认为打开 App，详情参考 action 参数说明
custom_content	String	android	无	否	用户自定义的参数（需要序列化为 JSON String） 获取方式详见通知点击跳转-客户端获取参数 **说明：** 华为官方通知：「2021年9月30日起停用V2协议」。移动推送已将华为推送协议升级到V5，V5协议不支持通过【附加参数】字段携带自定义参数。如果您集成了华为厂商通道，建议您改用 Intent 方式携带自定义参数，否则将导致自定义参数不能成功通过华为推送通道下发。
show_type	Integer	android	2	否	应用前台时，是否展示通知 。 默认展示，仅对移

					动推送自建通道、FCM 通道有效 1：不展示 2：展示 **说明**：若取值为1且应用在前台，终端用户对该条推送无感知，但有抵达数据上报

说明：

华为官方通知：「2021年9月30日起停用V2协议」。移动推送已将华为推送协议升级到V5，V5协议不支持通过【附加参数】字段携带自定义参数。如果您集成了华为厂商通道，建议您改用 Intent 方式携带自定义参数，否则将导致自定义参数不能成功通过华为推送通道下发。

action

参数说明

字段名	类型	父项目	默认值	必需	参数描述
action_type	Integer	action	1	否	点击动作类型， 1：打开 activity 或 App 本身 2：打开浏览器 3：打开 App 自定义页面（推荐使用，参考 使用 Inten 方式跳转指引）
activity	String	action	无	action_type 为1，且需要打开 activity 时必选	activity 完整名称，例如 com.x.y.PushActivity
aty_attr	Object	action	无	action_type 为1，且需要打开 activity 时可选	activity 属性 if：Intent 的 Flag 属性，类型为 Integer pf：PendingIntent 的 Flag 属性，类型为 Integer
browser	Object	action	无	action_type 为2时必选	打开浏览器操作， url：网页地址，仅支持 http、https，类型为 String confirm：是否需要用户确认，类型为 Integer （1）1：需要确认 （2）0：不需要确认
intent	String	action	无	action_type 为3时必选	自定义 scheme，例如 `xgscheme://com.tpns.push/notify_detail`

完整的消息示例如下：

```
{
    "title": "xxx",
    "content": "xxxxxxxxx",
    "xg_media_resources": "xxx" , //此处填富媒体元素地址，例如https://www.xx.com/img/bd
    "xg_media_audio_resources":"xxx", //此处填音频富媒体元素地址，例如http://sc1.111ttt.
    "thread_id":"活动_id",
    "thread_sumtext":"运营活动",
    "accept_time": [
        {
            "start": {//时间段起始时间,
                "hour": "13",//起始时间 小时值，取值 [0:24)
                "min": "00"// 起始时间 分钟值，取值[0:60)
            },
            "end": {//时间段结束时间
                "hour": "14",//结束时间 小时值，取值 [0:24)
                "min": "00" //结束时间 分钟值,取值[0:60)

            }
        },
        {
            "start": {
                "hour": "00",
                "min": "00"
            },
            "end": {
                "hour": "09",
                "min": "00"
            }
        }
    ],
    "android": {
                "n_ch_id": "default_message",
                "n_ch_name": "默认通知",
                "n_id": 0,
                "builder_id": 0,
                "ring": 1,
                "ring_raw": "ring",
                "badge_type":-1,
                "vibrate": 1,
                "lights": 1,
                "clearable": 1,
                "icon_type": 0,
                "icon_res": "xg",
                "style_id": 1,
                "small_icon": "xg",
```

```
                "action": {
                        "action_type": 1,// 动作类型，1，打开activity或app本身；2，打开浏
                        "activity": "com.x.y.PushActivity",
                        "aty_attr": {// activity属性，只针对action_type=1的情况
                                "if": 0, // Intent的Flag属性
                                "pf": 0  // PendingIntent的Flag属性
                        },
                        "browser": {
                                "url": "https://cloud.tencent.com ", // 仅支持http、
                                "confirm": 1 // 是否需要用户确认
                        },
                        "intent": "xgscheme://com.tpns.push/notify_detail" //SDK版本
                },
                "custom_content":"{\\"key\\":\\"value\\"}"
        }
}
```

iOS 通知消息

iOS 平台具体字段如下表：

字段名	类型	父项目	默认值	必需	参数描述
title	String	message	无	是	消息标题，此字段会覆盖 alert 下的 title 中的内容。
content	String	message	无	是	消息内容，此字段会覆盖 alert 下的 body 中的内容。
thread_id	String	message	无	否	通知分组折叠的组别识别名
ios	Object	message	无	是	iOS 消息结构体，请参见 iOS 字段说明
show_type	Integer	message	2	否	应用前台时，是否展示通知 。 1：不展示 2：展示 说明：若取值为1且应用在前台，终端用户对该条推送无感知，但有抵达数据上报
xg_media_resources	String	message	无	否	图片、音视频富媒体元素 url 地址，详情请参见 富媒体通知

iOS 字段说明

字段名	类型	父项	默认	必	参数描述

		目	值	需	
aps	Object	ios	无	是	苹果推送服务（APNs）特有的消息体字段，请参见 aps 参数说明，其他详细介绍请参见官网 Payload。
custom_content	String	ios	无	否	自定义下发的参数，需要序列化为 json string

aps 参数说明

aps 参数说明

字段名	类型	父项目	默认值	必需	参数描述
alert	Object	aps	无	是	包含标题和消息内容
badge_type	Integer	aps	无	否	用户设置角标数字： -1：角标数字不变 -2：角标数字自动加1 >=0：设置自定义角标数字
category	String	aps	无	否	下拉消息时显示的操作标识
mutable-content	Integer	aps	1	否	通知拓展参数。 推送的时候携带 "mutable-content":1，说明是支持 iOS 10 的 Service Extension 开启后，推送详情中会有抵达数据上报，使用该功能前请按照 通知服务扩展的使用说明 实现 Service Extension 接口，如果不携带此字段则没有抵达数据上报
sound	String	aps	无	否	sound 字段使用情况如下： 播放系统默认提示音，"sound":"default" 播放本地自定义铃声，"sound":"chime.aiff" 静音效果，"sound":"" 或者是去除 sound 字段。 自定义铃声说明：格式必须是 Linear PCM、MA4（IMA/ADPCM）、alaw，μLaw 的一种，将声频文件放到项目 bundle 目录中，且时长要求30s以下，否则就是系统默认的铃声。
interruption-level	String	aps	active	否	仅对 iOS 15 以后的设备生效，需要在Capability 中打开Time Sensitive Notifications选项，有4个值可以选择设置： passive：被动通知，即并不需要及时关注的通知。 active：主动通知，默认的通知。

					time-sensitive：时效性通知，需要人们立刻注意的通知。 critical：重要通知，需要立刻关注的通知。

完整的消息示例如下：

```
{
    "title": "xxx",
    "content": "xxxxxxxxx",
    "thread_id":"活动_id",
    "xg_media_resources":"https://www.xx.com/img/bd_logo1.png",
    "show_type":1,
    "ios":{
        "aps": {
            "alert": {
                "subtitle": "my subtitle"
            },
            "badge_type": 5,
            "category": "INVITE_CATEGORY",
            "sound":"default",
            "interruption-level":"time-sensitive",
            "mutable-content":1
        },
       "custom_content":"{\\"key\\":\\"value\\"}"
    }
}
```

Android 透传消息

透传消息，Android 平台特有，即不显示在手机通知栏中的消息，可以用来实现让用户无感知的向 App 下发带有控制性质的消息。

注意：

因厂商限制，Android 透传消息只通过移动推送自建通道下发，无法通过厂商通道下发。

Android 平台具体字段如下表：

字段名	类型	父项目	默认值	是否必需	参数描述
title	String	message	无	是	命令描述
content	String	message	无	是	命令内容
android	Object	message	无	否	安卓消息结构体
accept_time	Array	message	无	否	消息将在哪些时间段允许推送给用户。

					单个元素由 "start" 和 "end" 组成的起止时间对组成。 "start" 和 "end" 由 hour （小时）和 min（分钟）描述对应时刻，详细参考具体示例。 **注意**：因厂商限制， 仅对移动推送自建通道有效。
custom_content	String	android	无	否	需要序列化为 json string

具体完整示例：

```
{
    "title": "this is title",
    "content": "this is content",
    "android": {
      "custom_content":"{\\"key\\":\\"value\\"}"
    },
    "accept_time": [
        {
            "start": {
                "hour": "13",
                "min": "00"
            },
            "end": {
                "hour": "14",
                "min": "00"
            }
        },
        {
            "start": {
                "hour": "00",
                "min": "00"
            },
            "end": {
                "hour": "09",
                "min": "00"
            }
        }
    ]
}
```

iOS 静默消息

静默消息，iOS 平台特有，类似 Android 中的透传消息，消息不展示，当静默消息到达终端时，iOS 会在后台唤醒 App 一段时间（小于30s），让 App 来处理消息逻辑。

具体字段如下表：

字段名	类型	父项目	默认值	是否必要	参数描述
ios	Object	message	无	是	ios 消息结构体
aps	Object	ios	无	是	苹果推送服务（APNs）特有的，其中最重要的键值对如下： content-available：标识消息类型（必须为1），类型为 Integer 不能包含 alert、sound、badge_type 字段，详细介绍请参见 Payload 注意：content-available: 1与message_type:"notify"字段互斥，请勿同时使用
custom_content	String	ios	无	否	需要序列化为 json string

具体完整示例：

```
{
    "ios":{
        "aps": {
            "content-available": 1
        },
      "custom_content":"{\\"key\\":\\"value\\"}"
    }
}
```

可选参数

Push API 可选参数是除了 `audience_type` 、 `message_type` 、 `message` 以外，可选的高级参数。

参数名	类型	父项目	必需	默认值	描述
expire_time	Integer	无	否	86400（24小时）	消息离线存储时间（单位为秒），最长72小时 若 expire_time = 0，则表示实时消息 若 expire_time 大于0，且小于800s，则系统会重置为800s

					若expire_time >= 800s，按实际设置时间存储，最长72小时设置的最大值不得超过259200，否则会导致推送失败如需调整离线消息时间，请联系 在线客服 申请
send_time	String	无	否	当前系统时间	指定推送时间，可选择未来90天内的时间： 格式为 yyyy-MM-DD HH:MM:SS 若小于服务器当前时间，则会立即推送 仅全量推送、号码包推送和标签推送支持此字段
multi_pkg	Boolean	无	否	false	多包名推送：当 App 存在多个渠道包（例如应用宝、豌豆荚等），并期望推送时所有渠道的 App 都能收到消息，可将该值设置为 true。 **注意：**该参数默认控制移动推送自建通道的多包名推送，需要实现厂商通道多包名推送详见 厂商通道多包名配置 文档
loop_param	Object	无	否	0	仅全量推送、号码包推送和标签推送支持此字段，详情见下文 loop_param 参数说明
group_id	String	无	否	tpns_yyyymmdd，yyyymmdd 代表推送日期	**该字段已废弃，后续会下线，若需要使用聚合统计请使用推送计划字段：plan_id**
plan_id	String	无	否	无	推送计划 ID，推送计划创建及使用方式可 参考文档
tag_rules	Array	无	仅标签推送必需	无	标签组合推送，可设置'与'、'或'、'非'组合规则 注意：当与 tag_list 二者共存时，tag_list 字段自动无效，参数说明请查看 tag_rules 参数说明
account_list	Array	无	单账号推送、	无	若单账号推送： 要求 audience_type = account 参数格式：["account1"]

			账号列表推送时必需		若账号列表推送： 参数格式： ["account1","account2"] 最多1000 个 account
account_push_type	Integer	无	账号推送时可选	0	0：往账号的最新的 device 上推送信息 1：往账号关联的所有 device 设备上推送信息
account_type	Integer	无	否	0	账号类型，需要与推送的账号所属类型一致，取值可参考 账号类型取值表
token_list	Array	无	单设备推送、设备列表推送时必需	无	若单设备推送： 要求 audience_type=token 参数格式：["token1"] 若设备列表推送： 参数格式：["token1","token2"] 最多1000个 token
ignore_invalid_token	int	无	否	0	0：代表如果有无效的token则这个接口调用失败 1：代表忽略无效的token继续进行下发 **注意**：仅对 token 列表推送和单 token 推送有效
push_speed	Integer	无	否	无	推送限速设置每秒 X 条，X 取值参数范围1000 - 50000 仅全量推送、号码包推送和标签推送有效
collapse_id	Integer	无	否	系统默认分配一个 collapse_id	消息覆盖参数，在前一条推送任务已经调度下发后，如果第二条推送任务携带相同的 collapse_id 则会停止前一条推送中尚未下发的移动推送自建通道数据，同时会覆盖展示第一条推送任务的消息。 已完成任务的 collapse_id 可以通过 单个任务推送信息查询接口 获取。

					目前仅支持全推、标签推送、号码包推送。
channel_rules	Array	无	否	无	推送通道选择策略。 可自定义该条推送允许通过哪些通道下发，默认允许通过所有通道下发，详细推送策略参考 通道策略 channel_rules 数组单元素数据结构见下 channel_rules 参数说明
tpns_online_push_type	Integer	无	否	0	在线设备是否通过移动推送自建通道下发： 0：默认在线走移动推送自建通道下发 1：在线不优先走移动推送自建通道下发
force_collapse	Boolean	无	否	false	对于不支持消息覆盖的 OPPO、vivo 通道的设备，是否进行消息下发。 false：不下发消息 true：下发消息

说明：

对于 collapse_id，有以下使用条件：

暂不支持用户自定义此参数，需要移动推送生成的 collapse_id。

目前仅支持移动推送自建通道、APNS 通道、小米通道、魅族通道以及华为系统版本 EMUI10 及以上的设备。

对于华为通道，覆盖消息时携带自定义参数需要使用 intent 方式，如使用 custom_content 方式携带自定义参数，接口层会进行拦截。

目前 OPPO 通道 vivo 通道不支持覆盖消息。当新创建覆盖消息时可通过 force_collapse 字段设置为 false 来关闭 vivo、OPPO 通道的下发。

tag_rules 参数说明

字段	类型	父项目	必填	描述
tag_items	Array	tag_rules	是	标签规则，参见 tag_items 说明
operator	String	tag_rules	是	tag_rules 数组内各元素的运算符，第一个 tag_rules 元素的 operator 为无效数据，第二个 tag_rules 元素的 operator 作为第一个和第二个 tag_rules 元素之间的运算符。

				OR：或运算 AND：且运算
is_not	Boolean	tag_rules	是	是否对 tag_items 数组的运算结果进行非运算。 true：进行非运算 false：不进行非运算。

tag_items 说明

字段	类型	父项目	必填	描述
tags	Array	tag_items	是	具体标签值，类型：string，如 tag1，guangdong 等。
is_not	Boolean	tag_items	是	是否对 tag 数组的运算结果进行非运算。 true：非运算 false：不进行非运算
tags_operator	String	tag_items	是	tags 内标签对应的运算符。 OR：或运算 AND：且运算。
items_operator	String	tag_items	是	tag_items 数组内各元素的运算符，第一个 tag_items 元素的 items_operator 为无效数据，第二个 tag_items 元素的 items_operator 作为第一个和第二个 tag_items 元素之间的运算符。 OR ：或运算 AND ：且运算 **注意**：不同规则之间运算符逻辑优先级「AND」>「OR」
tag_type	String	tag_items	是	参见 tag_type 取值表

tag_type 取值表

标签名称	tag_type 取值	标签名举例
自定义标签	xg_user_define	tag1，tag2 等
App版本	xg_auto_version	1.1.0，1.2.0.1等
设备省份信息	xg_auto_province	guangdong，shanghai 等
活跃信息	xg_auto_active	20200131，20200201等
XG SDK版本	xg_auto_sdkversion	1.1.5.2，1.1.5.3等
系统语言	xg_auto_systemlanguage	zh，en 等

手机品牌	xg_auto_devicebrand	Xiaomi，vivo 等
手机机型	xg_auto_deviceversion	MI 9 SE，vivo X9Plus 等
国家	xg_auto_country	CN，SG 等

说明：

详细使用方法可参见 标签推送示例。

channel_rules 参数说明

字段名	类型	父项目	是否必填	参数说明
channel	String	channel_rules	是	下发推送通道。 xg：自建通道 hw：华为通道 xm：小米通道 mz：魅族通道 vivo：vivo 通道 oppo：OPPO 通道 apns：APNs 通道 honor：荣耀通道 fcm：FCM通道
disable	Boolean	channel_rules	是	是否关闭 channel 中对应的通道， 默认打开通道。 true：关闭 false：打开

loop_param 参数说明

字段名	类型	父项目	是否必填	注释
startDate	String	loop_param	是	循环区间开始日期，可选择未来90天内的时间。格式YYYY-MM-DD，例如2019-07-01
endDate	String	loop_param	是	循环区间截止日期，可选择未来90天内的时间。格式YYYY-MM-DD，例如2019-07-07
loopType	Integer	loop_param	是	循环类型 天： 1 周： 2 月： 3

loopDayIndexs	Array	loop_param	是	按天循环：填写[0]，表示每天的任务，按周循环：填周几[0-6]，如[0, 1, 2]表示每周的星期天，周一，周二进行推送，按月循环：填写日期，如[1,10,20],每个月1,10,20号
dayTimes	Array	loop_param	是	具体推送时间，格式 HH:MM:SS，例如["19:00:00", "20:00:00"]，表示每天的19点，20点进行推送

应答参数

字段名	类型	注释
seq	Integer	与请求包一致（如果请求包无该字段，则该字段返回为0）
push_id	String	推送 ID **注意**：如果您是循环推送类型，则会返回多个 pushid 放在一个数组类型里
invalid_targe_list	Array	仅对 token 列表推送和单 token 推送 且 ignore_invalid_token 值为1时返回，该字段会存储被过滤的无效 token ，有效 token 会正常下发
ret_code	Integer	错误码，详细参照错误码对照表
environment	String	用户指定推送环境，仅支持 iOS product： 生产环境 dev： 开发环境
err_msg	String	请求出错时的错误信息
result	String	请求正确时 若有额外数据要返回，则结果封装在该字段的 json 中 若无额外数据，则可能无此字段

示例说明

Android 账号推送请求消息

```
{
    "audience_type": "account",
    "account_list": [
      "account1"
  ],
    "multi_pkg":true,
    "push_speed":50000,
```

```
    "channel_rules": [
        {
            "channel": "mz",
            "disable": true
        },
        {
            "channel": "xm",
            "disable": false
        }
    ],
    "message_type": "notify",
    "message": {
    "title": "测试标题",
    "content": "测试内容",
    "xg_media_resources": "xxx1" , //此处填富媒体元素地址，例如https://www.xx.com/img/bc
    "xg_media_audio_resources":"xxx", //此处填音频富媒体元素地址，例如http://sc1.111ttt.
    "accept_time": [
        {
            "start": {//时间段起始时间,
                "hour": "13",//起始时间 小时值，取值 [0:24)
                "min": "00"// 起始时间 分钟值，取值[0:60)
            },
            "end": {//时间段结束时间
                "hour": "14",//结束时间 小时值，取值 [0:24)
                "min": "00" //结束时间 分钟值,取值[0:60)

            }
        },
        {
            "start": {
                "hour": "00",
                "min": "00"
            },
            "end": {
                "hour": "09",
                "min": "00"
            }
        }
    ],
    "android": {
                "n_ch_id": "default_message",
                "n_ch_name": "默认通知",
                "n_id": 0,
                "builder_id": 0,
                "ring": 1,
                "ring_raw": "ring",
                "badge_type":-1,
```

```
                "vibrate": 1,
                "lights": 1,
                "clearable": 1,
                "icon_type": 0,
                "icon_res": "xg",
                "style_id": 1,
                "small_icon": "xg",
                "action": {
                        "action_type": 1,// 动作类型，1，打开 activity 或 app 本身；2，
                        "activity": "xxx",
                        "aty_attr": {// activity属性，只针对action_type=1的情况
                                "if": 0, // Intent的Flag属性
                                "pf": 0  // PendingIntent的Flag属性
                        },
                        "browser": {
                                "url": "xxxx ", // 仅支持http、https
                                "confirm": 1 // 是否需要用户确认
                        },
                        "intent": "xxx" //SDK版本需要大于等于1.0.9，然后在客户端的intent
                 },
                 "custom_content":"{\\"key\\":\\"value\\"}"
                }
            }
}
```

账号推送应答消息

```
{
    "seq": 0,
    "environment": "product",
    "ret_code": 0,
    "push_id": "3895624686"
}
```

iOS 单设备推送请求消息

```
{
    "audience_type": "token",
    "environment":"dev",
    "token_list": [ "05da87c0ae********fa9e08d884aada5bb2"],
    "message_type":"notify",
    "message":{
     "title": "推送标题",
    "content": "推送内容",
    "ios":{
```

```
        "aps": {
            "alert": {
                "subtitle": "推送副标题"
            },
            "badge_type": -2,
            "sound":"Tassel.wav",
            "category": "INVITE_CATEGORY"

        },
       "custom_content":"{\\"key\\":\\"value\\"}"
    }
 }
}
```

单设备推送应答消息

```
{
    "seq": 0,
    "push_id": "427184209",
    "ret_code": 0,
    "environment": "dev",
    "err_msg": "",
    "result": "[0]"
}
```

标签推送场景（tag_rules 方式）

场景一：广东和湖南，并且是20200408当天活跃过的男性用户

表达式：（xg_auto_province.guangdong 或 xg_auto_province.hunan）与 xg_auto_active.20200408 与 xg_user_define.male

```
{
  "audience_type": "tag",
  "tag_rules": [
        {
            "tag_items": [
                {
                    "tags": [
                        "guangdong",
                        "hunan"
                    ],
                    "is_not": false,    //是否对tags内标签计算的结果进行非运算，true-进行非运算，false-不进行非运算
                    "tags_operator": "OR",   //tags内标签对应的运算符
```

```
                    "items_operator": "OR", //tag_items内各元素的运算符，第一个元素的items_operator为无效数据，第二个元素的items_operator作为第一个和第二个元素之前的运算符，以此类推
                    "tag_type": "xg_auto_province" //tags内标签对应的标签类型
                },
                {
                    "tags": [
                        "20200408"
                    ],
                    "is_not": false,
                    "tags_operator": "OR",
                    "items_operator": "AND",
                    "tag_type": "xg_auto_active"
                },
                {
                    "tags": [
                        "male"
                    ],
                    "is_not": false,
                    "tags_operator": "OR",
                    "items_operator": "AND",
                    "tag_type": "xg_user_define"
                }
            ],
            "operator": "OR",
            "is_not": false
        }
    ]
}
```

场景二：近3天活跃，并且 App 版本不为1.0.2的华为用户

表达式：（xg_auto_active.20200406 或 xg_auto_active.20200407 或 xg_auto_active.20200408）与 （非 xg_auto_version.1.0.2） 与 xg_auto_devicebrand.huawei

```
{
  "audience_type": "tag",
  "tag_rules": [
        {
            "tag_items": [
                {
                    "tags": [
                        "20200406",
                        "20200407",
                        "20200408"
                    ],
                    "is_not": false,    //是否对tags内标签计算的结果进行非运算，true-进行非运算，false-不进行非运算
```

```
                    "tags_operator": "OR",  //tags内标签对应的运算符
                    "items_operator": "OR", //tag_items内各元素的运算符，第一个元素的items_operator为无效数据，第二个元素的items_operator作为第一个和第二个元素之前的运算符，以此类推
                    "tag_type": "xg_auto_active" //tags内标签对应的标签类型
                },
                {
                    "tags": [
                        "1.0.2"
                    ],
                    "is_not": true,
                    "tags_operator": "OR",
                    "items_operator": "AND",
                    "tag_type": "xg_auto_verison"
                },
                {
                    "tags": [
                        "huawei"
                    ],
                    "is_not": false,
                    "tags_operator": "OR",
                    "items_operator": "AND",
                    "tag_type": "xg_auto_devicebrand"
                }
            ],
            "operator": "OR",
            "is_not": false
        }
    ]
}
```

创建推送计划接口

最近更新时间：2024-01-17 14:24:51

接口说明

请求方式：POST

请求地址：

```
服务地址/v3/push/plan/add_plan_push
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

说明：

调用频率限制：200次/天。

请求参数

参数名	类型	是否必须	参数说明
planName	string	是	推送计划名称，限制60个英文
planDescribe	string	是	推送计划描述，限制300个中英文

响应参数

参数名	类型	是否必须	参数说明
retCode	string	是	返回状态码
errMsg	string	是	错误信息
result	object	是	详见下表 `PlanRecordData` 描述

PlanRecordData

参数名	类型	是否必须	参数说明
planId	string	是	由 TPNS 生成的推送计划 ID

planName	string	是	推送计划名称，限制23个英文
planDescribe	string	是	推送计划描述，限制300个中英文
planPushSource	string	是	-
createdAt	uint32	是	创建时间，时间戳格式
updateAt	uint32	是	更新时间，时间戳格式

示例说明

请求示例

```
{
            "planName":"TPNS_TEST123",
            "planDescribe":"plan_test"
}
```

返回示例

```
{
    "retCode": 0,
    "ErrMsg": "save success",
    "result": {
        "planId": "48",
        "planName": "TPNS_TEST123",
        "planDescribe": "plan_test",
        "planPushSource": "api",
        "createdAt": 1593314408,
        "updateAt": 1593314408
    }
}
```

号码包上传接口

最近更新时间：2024-01-17 14:24:51

接口说明

请求方式：POST。

```
服务地址/v3/push/package/upload
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：用户需要通过文件的方式，对批量账号上传号码包文件。然后对号码包中的文件进行推送。号码包推送接口主要包括号码包上传接口、以及号码包推送接口。

注意：

账号包文件名：长度限制为 [1, 100]。

账号包格式及大小： 支持 `zip\\txt\\csv` 文件；大小保持在100MB以内。

`zip` 压缩包中可包含：单个 `.txt` 或 `.csv` 文件（不能嵌套文件夹）。

. `txt` 文件要求：（1）编码为 UTF-8；（2）每行一个账号，账号长度限制为 [2, 100]。

. `csv` 文件要求：（1）只能有一列；（2）每行一个账号，账号长度限制为 [2, 100]。

请求参数

参数名	类型	是否必须	参数说明
file	form-data	是	账号包格式及大小： 支持 zip\\txt\\csv 文件；大小保持在100MB以内 zip 压缩包中可包含：单个 .txt 或 .csv 文件（不能嵌套文件夹） .txt 文件要求：（1）编码为 UTF-8；（2）每行一个账号，账号长度限制为 [2, 100] .csv 文件要求：（1）只能有一列；（2）每行一个账号，账号长度限制为 [2, 100]

响应参数

参数名	类型	是否必须	参数说明
retCode	Integer	是	错误码

errMsg	String	是	请求出错时的错误信息
uploadId	Integer	是	当上传文件成功时，将反馈一个正整数 uploadId ，代表上传文件 ID，提供给后续号码包接口进行推送

请求示例

Python3

Golang

C#

PHP

JAVA

C++

```
import base64
from pip._vendor import requests
from pip._vendor.urllib3 import encode_multipart_formdata

def upload(url, filePath, accessId, secret, data={}, header={}):
    openFile = open(filePath, 'rb')
    data['file'] = (openFile.name, openFile.read())
    encode_data = encode_multipart_formdata(data)
    data = encode_data[0]
    header['Content-Type'] = encode_data[1]
    authInfo = accessId + ":" + secret

    header['Authorization'] = "Basic " + str(base64.b64encode(bytes(authInfo,
encoding="utf8")),encoding="utf8")

    r = requests.post(url, headers=header, data=data)
    print(r.json())
```

```
func Upload(url string, filePath string, accessId string, secret string)(resp
string , err error) {
    fp, err := os.Open(filePath)
    if err != nil {
        return resp, err
    }
    defer fp.Close()
    body := &bytes.Buffer{}
    writer := multipart.NewWriter(body)
```

```
    defer writer.Close()
    part, err := writer.CreateFormFile("file", filepath.Base(fp.Name()))
    if err != nil {
        return resp, err
    }
    io.Copy(part, fp)
    writer.Close()
    httpReq, err := http.NewRequest(http.MethodPost, url, body)
    if err != nil {
        return resp, err
    }

    httpReq.Header.Add("Content-Type", writer.FormDataContentType())

    authInfo := base64.StdEncoding.EncodeToString([]byte(accessId + ":" +
secret))
    httpReq.Header.Add("Authorization", fmt.Sprintf("Basic %s", authInfo))
    cli := &http.Client{}
    httpResp, err := cli.Do(httpReq)
    if err != nil {
        return resp, err
    }
    defer httpResp.Body.Close()
    data, err := ioutil.ReadAll(httpResp.Body)
    if err != nil {
        return resp, err
    }
    if httpResp.StatusCode != http.StatusOK {
        return resp, fmt.Errorf("response error, status: %s, body: %s",
httpResp.Status, string(data))
    }

    resp = string(data)
    return resp, nil
}
```

```
        public string Upload(string url, string filePath, uint accessId, string
secretKey)
        {
            var tmp = accessId.ToString() + ":" + secretKey;
            var sign =
System.Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(tmp));
            MultipartFormDataContent form = new MultipartFormDataContent();
```

```
            var content = new StreamContent(new FileStream(filePath,
FileMode.Open));
            content.Headers.ContentDisposition = new
ContentDispositionHeaderValue("form-data")
            {
                Name = "file",
                FileName = Path.GetFileName(filePath)
            };
            form.Add(content);
            using(HttpClient client = new HttpClient())
            {
                client.DefaultRequestHeaders.Add("Authorization","Basic " +
sign);
                client.DefaultRequestHeaders.ExpectContinue = true;
                var response = (client.PostAsync(url, form)).Result;
                var result = response.Content.ReadAsStringAsync().Result;
                if (response.StatusCode != HttpStatusCode.OK) {
                    throw new Exception(result);
                }
                return result;
            }
        }
```

```
function upload($url, $file, $accessId, $secretkey) {
    $sign = base64_encode($accessId . ":" . $secretKey);
    $headers = array("Content-type: multipart/form-data", "Authorization: Basic
" . $sign);
    $cfile = new \\CURLFile($file,'multipart/form-data',basename($file));
    $options = array(
        CURLOPT_HTTPHEADER      => $headers,
        CURLOPT_HEADER          => 0,
        CURLOPT_SSL_VERIFYPEER  => false,
        CURLOPT_SSL_VERIFYHOST  => 0,
        CURLOPT_POST            => 1,
        CURLOPT_URL             => $url,
        CURLOPT_RETURNTRANSFER  => 1,
        CURLOPT_TIMEOUT         => 10000,
        CURLOPT_POSTFIELDS      => array("file" => $cfile)
    );
    $ch = curl_init();
    curl_setopt_array($ch, $options);
    $ret = curl_exec($ch);
    $error = curl_error($ch);
    $info = curl_getinfo($ch);
    curl_close($ch);
```

```
    if ($error != "") {
        throw new \\Exception($error);
    }
    $code = $info["http_code"];
    if ($code != 200) {
        throw new \\Exception("status: " . $code . ", message: " . $ret);
    }
    return $ret;

}
```

```
public JSONObject Upload(String url, String filePath, String accessId, String
secret) {
        OkHttpClient client = new OkHttpClient();
        File file  = new File(filePath);
        RequestBody requestBody = new MultipartBody.Builder()
                .setType(MultipartBody.FORM)
                .addFormDataPart("file", file.getName(),
                        RequestBody.create(MediaType.parse("multipart/form-
data"), file))
                .build();

        String authString = accessId+ ":" + secret;
        byte[] authEncBytes = Base64.encodeBase64(authString.getBytes());
        String authStringEnc = new String(authEncBytes);

        Request.Builder builder  = new Request.Builder()
                .url(url)
                .post(requestBody)
                .addHeader("Authorization", "Basic " + authStringEnc);



        Request request = builder.build();
        JSONObject jsonRet = null;
        Response response = null;
        try {
            response = client.newCall(request).execute();
            if(response.code() == 200){
                String retMsg = response.body().string();
                jsonRet = new JSONObject(retMsg);
            }
            else{
                jsonRet = new JSONObject();
```

```
                jsonRet.put("retCode", 10101);
                jsonRet.put("errMsg", "CallApiError,HttpStatus Code:" +
response.code());
            }
        } catch (IOException e) {
            jsonRet = new JSONObject();
            jsonRet.put("retCode", 10100);
            jsonRet.put("errMsg", e.getMessage());
        }finally {
            if (response != null) {
                response.close();
            }
        }
        return jsonRet;
    }
```

```
struct memory {
    char *data;
    size_t size;
    memory(): data(NULL), size(0) {}
    ~memory() {
        if (data) {
            free(data);
        }
    }
};

static size_t callback(void *data, size_t size, size_t nmemb, void *userp)
{
    size_t realsize = size * nmemb;
    struct memory *mem = (struct memory *)userp;

    char *ptr = (char *)realloc(mem->data, mem->size + realsize + 1);
    if(ptr == NULL)
        return 0;

    mem->data = ptr;
    memcpy(&(mem->data[mem->size]), data, realsize);
    mem->size += realsize;
    mem->data[mem->size] = 0;
    return realsize;
}
```

```
std::string upload(const std::string url, const std::string &file, uint32_t
accessId, const std::string &secretKey, std::string &err) {
    CURL *pcurl = curl_easy_init();
    if (pcurl == NULL) {
        err.assign("curl_easy_init error");
        return "";
    }
    struct curl_httppost *post = NULL;
    struct curl_httppost *last = NULL;
    std::string base;
    //for unix/linux
    size_t pos = file.find_last_of("/");
    if (pos >= 0) {
        base = file.substr(pos + 1);
    }
    //for windows
    pos = file.find_last_of("\\\\");
    if (pos >= 0) {
        return base = file.substr(pos + 1);
    }
    curl_formadd(&post, &last, CURLFORM_PTRNAME, "file", CURLFORM_FILE,
file.c_str(), CURLFORM_FILENAME, base.c_str(), CURLFORM_END);


    char buf[128];
    char dst[128];
    char authInfo[256];
    snprintf(buf, sizeof(buf), "%u:%s", accessId, secretKey.c_str());
    Base64encode(dst, buf, strlen(buf));
    snprintf(authInfo, sizeof(authInfo), "Authorization: Basic %s", dst);


    curl_slist *plist = NULL;
    plist = curl_slist_append(plist, "Content-Type: multipart/form-data");
    plist = curl_slist_append(plist, authInfo);


    curl_easy_setopt(pcurl, CURLOPT_CONNECTTIMEOUT, 10);
    curl_easy_setopt(pcurl, CURLOPT_TIMEOUT, 10);
    curl_easy_setopt(pcurl, CURLOPT_POST, true);


    curl_easy_setopt(pcurl, CURLOPT_HTTPHEADER, plist);
    curl_easy_setopt(pcurl, CURLOPT_URL, url.c_str());


    curl_easy_setopt(pcurl, CURLOPT_HTTPPOST, post);
```

```
    curl_easy_setopt(pcurl, CURLOPT_WRITEFUNCTION, callback);
    struct memory chunk;
    curl_easy_setopt(pcurl, CURLOPT_WRITEDATA, (void *)&chunk);
    curl_easy_setopt(pcurl, CURLOPT_NOSIGNAL, 1);


    CURLcode res = curl_easy_perform(pcurl);
    curl_slist_free_all(plist);


    // Check for errors
    if (res != CURLE_OK)
    {
        char buf[1024];
        snprintf(buf, sizeof(buf), "curl_easy_perform error: %s",
curl_easy_strerror(res));
        err.assign(buf);
        curl_easy_cleanup(pcurl);
        return "";
    }
    long status = 0;
    curl_easy_getinfo(pcurl, CURLINFO_RESPONSE_CODE, &status);
    curl_easy_cleanup(pcurl);
    if (status != 200) {
        char buf[1024];
        snprintf(buf, sizeof(buf), "%ld %s", status, chunk.data);
        err.assign(buf);
        return "";
    }


    std::string ret;
    ret.assign(chunk.data);
    return ret;
}
```

Token 包上传接口

最近更新时间：2024-01-17 14:24:51

接口说明

请求方式：POST。

```
服务地址/v3/push/package/upload
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：用户需要通过文件的方式，对批量设备 Token 上传 Token 包文件， 然后对 Token 包中的文件进行推送。

请求参数

参数名	类型	是否必须	参数说明
file	form-data	是	Token 包文件名：长度限制为 [1,100] Token 包格式及大小： 支持 zip\\txt\\csv 文件；大小保持在100MB以内 压缩包中可含有： 单个 .txt 或 .csv 文件 (不能嵌套文件夹) .txt 文件要求：（1）编码为 UTF-8；（2）每行一个 Token，有效 Token 长度为36位 .csv 文件要求：（1）只能有一列；（2）每行一个 Token，有效 Token 长度为36位

响应参数

参数名	类型	是否必须	参数说明
retCode	Integer	是	错误码
errMsg	String	是	请求出错时的错误信息
uploadId	Integer	是	当上传文件成功时，将返回一个正整数 uploadId ，代表上传文件 ID，提供给推送接口进行 Token 包推送

请求示例

Curl 示例

```
curl -X POST
https://api.tpns.tencent.com/v3/push/package/upload

-H 'Authorization: Basic  应用的认证信息'
-F 'file=@C:\\上传文件绝对路径'
```

Python 示例

```
import requests

url = "https://api.tpns.tencent.com/v3/push/package/upload"

files = {'file':open('文件名', 'rb')}
upload_data={"fileName":"文件绝对地址"}

headers = {
    'Authorization': "Basic 应用鉴权信息",
    }

response = requests.request("POST", url, data=upload_data, headers=headers, files=f
print(response.text.encode('utf-8'))
```

注意：

应用的认证信息，请参见 Basic Auth 认证 文档。

标签相关接口
标签绑定与解绑

最近更新时间：2024-01-17 14:24:51

接口说明

请求方式：POST。

```
服务地址/v3/device/tag
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：

Tag API 是所有 tag 接口的统称。Tag API 有多种设置、更新、删除接口，具体的接口见下文。

标签功能的使用场景可参考文档 标签功能使用说明。

参数说明

请求参数

参数名	类型	是否必需	参数说明
operator_type	Integer	是	操作类型： 1 - 增加单个 tag，对单个 token 而言 2 - 删除单个 tag，对单个 token 而言 3 - 增加多个 tag，对单个 token 而言 4 - 删除多个 tag，对单个 token 而言 5 - 删除所有标签，对单个 token 而言 6 - 标签覆盖接口（支持多个标签或自定义类标签覆盖），对单个 token 而言 （此接口要清除历史标签之后，才开始设置新的标签，所以针对单个相同 token 的调用，需要间隔一段时间（建议大于1s），否则可能造成更新失败） 7 - 添加单个 tag，对多个 token 而言 8 - 删除单个 tag，对多个 token 而言 9 - 批量添加标签（每次调用最多允许设置20对，每个对里面标签在前，token 在后） 10 - 批量删除标签（每次调用最多允许设置20对，每个对里面标签在前，token 在后）

token_list	Array	否	设备列表： operator_type = 1,2,3,4,5,6,7,8时，必填 operator_type = 1,2,3,4,5,6时如果该参数包含多个 token 只会设置第一个 token 格式 eg：["token1","token2"] 列表最大不能超过500个值 token 字符串长度不能超过36
tag_list	Array	否	标签列表： operator_type = 1,2,3,4,6,7,8时，必填，operator_type = 5时忽略 operator_type = 1,2,7,8 时，如果该参数包含多个 tag 时，如果只是对单个 tag 操作，则只会设置第一个 tag 格式 eg：["tag1","tag2"] 列表最大不能超过500个值 tag 字符串长度不能超过50
tag_token_list	Array	否	标签、设备对应列表： operator_type =9,10时，必填 格式 eg： [{"tag":"tag123", "token":"token123"}] 每个对里面标签在前，token 在后 列表最大不能超过500个值 tag 字符串长度不能超过50 token 字符串长度不能超过36

单个应用最多可以有10000个自定义 `Tag` ， 每个设备 `Token` 最多可绑定100个自定义 `Tag` ，如需提高该限制，请与我们 客服联系，每个自定义 `Tag` 可绑定的设备 token 数量无限制。

如果仅单次而非连续的标签设置接口调用，则接口调用方式无限制。

如果为连续标签设置接口调用，则需要注意如下：

如果需要对超过10个标签或超过10个 token 的连续标签设置接口调用时，建议使用批量接口，但为保证标签操作的正确，建议两次接口调用的时间间隔不低于1s。

如果调用非批量接口，则在明确拿到移动推送服务器的返回值时，再进行下次非批量标签接口调用，不建议使用多线程异步同时进行标签接口调用。

英文冒号":"是移动推送后台的关键字，用作标签的分类，如果设置标签时带了":"，则将第一个":"前面的字符串作为这个标签的类名，仅为了用于 operator_type 为6时按类进行标签覆盖的场景，如设备原来的标签为 level:1, level:2, male 这3个时，此种方式可直接用 level:3标签覆盖 level 类标签下的 level:1和 level:2标签，而无需先逐个解绑 level:1和 level:2后再绑定 level:4标签，详情见下面请求示例中 operator_type 为6的示例。

应答参数

字段名	类型	是否必填	注释
ret_code	Integer	是	错误码，详细参照错误码对照表

err_msg	String	否	请求出错时的错误信息
result	String	否	请求正确时： 若有额外数据要返回，则结果封装在该字段的 json 中 若无额外数据，则可能无此字段

示例说明

标签绑定解绑请求示例

增加单个 tag1，对单个 token1

```
{
    "operator_type": 1,
    "tag_list": [
        "tag1"
    ],
    "token_list": [
        "token1"
    ]
}
```

删除单个 tag1，对单个 token1

```
{
    "operator_type": 2,
    "tag_list": [
        "tag1"
    ],
    "token_list": [
        "token1"
    ]
}
```

增加多个 tag1、tag2，对单个 token1

```
{
    "operator_type": 3,
    "tag_list": [
        "tag1",
        "tag2"
    ],
    "token_list": [
        "token1"
    ]
}
```

删除多个 tag1、tag2，对单个 token1

```
{
    "operator_type": 4,
    "tag_list": [
        "tag1",
        "tag2"
    ],
    "token_list": [
        "token1"
    ]
}
```

删除所有标签，对单个 token1

```
{
    "operator_type": 5,
    "tag_list": [
        "tag1",
        "tag2"
    ],
    "token_list": [
        "token1"
    ]
}
```

标签覆盖自定义类标签，对单个 token1

```
{
    "operator_type": 6,
    "tag_list": [
        "test:2",
        "level"
    ],
    "token_list": [
        "token1"
    ]
}
```

说明：

若有其中一个或多个标签不带“：”号，则将 test:2 和 level 标签覆盖 token1 的全部自定义标签。

批量覆盖自定义类标签，对单个 token（此接口只覆盖到自定义类标签，而不是全部覆盖）

```
{
    "operator_type": 6,
    "tag_list": [
```

```
            "test:2",
            "level:2"
        ],
        "token_list": [
            "token1"
        ]
    }
```

说明：

若全部标签都带“:”，由于第一个“:”前面的字符串为标签的类，则仅会覆盖设备对应相同类的标签，如test:2覆盖test:*， level:2覆盖level:*，不会影响其它标签。

为多个 token1、token2 添加单个 tag1

```
{
    "operator_type": 7,
    "tag_list": [
        "tag1"
    ],
    "token_list": [
        "token1",
        "token2"
    ]
}
```

多个 token1、token2 删除单个 tag1

```
{
    "operator_type": 8,
    "tag_list": [
        "tag1"
    ],
    "token_list": [
        "token1",
        "token2"
    ]
}
```

批量设置标签

```
{
    "operator_type": 9,
    "tag_token_list": [
        {
            "tag": "tag1",
            "token": "token1"
        }
    ]
```

```
}
```

批量删除标签，为 tag1、tag2、tag3 删除标签

```
{
    "operator_type": 10,
    "tag_token_list": [
        {
            "tag": "tag1",
            "token": "token1"
        },
        {
            "tag": "tag2",
            "token": "token2"
        },
        {
            "tag": "tag3",
            "token": "token3"
        }
    ]
}
```

标签设置请求示例

```
POST /v3/device/tag HTTP/1.1
Host: api.tpns.tencent.com
Content-Type: application/json
Authorization: Basic YTViNWYwNzFmZjc3YTplYTUxMmViNzcwNGQ1ZmI1YTZhOTM3Y2FmYTcwZTc3MQ
Cache-Control: no-cache
Postman-Token: 4b82a159-afdd-4f5c-b459-de978d845d2f
{
    "operator_type": 1,
    "tag_list": [
        "tag1"
    ],
    "token_list": [
        "token1"
    ]
}
```

标签设置应答示例

```
{
    "seq": 0,
    "ret_code": 0
```

```
}
```

删除标签下所有设备

最近更新时间：2024-01-17 14:24:51

接口说明

```
服务地址/v3/device/tag/delete_all_device
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：删除某个标签下所有的设备接口。

参数说明

请求参数

参数名	类型	是否必需	参数说明
tag_list	Array	是	待删除标签列表：`"tag_list": ["test_tag_3_Ik0N0", "test_tag_2_Ik0N0"]`

应答参数

字段名	类型	是否必填	注释
seq	Integer	是	与请求包一致（如果请求包是非法 json 该字段为0）
ret_code	Integer	是	错误码，详细参照错误码对照表
err_msg	String	否	请求出错时的错误信息
result	String	否	请求正确时： 若有额外数据要返回，则结果封装在该字段的 json 中 若无额外数据，则可能无此字段

示例说明

请求示例

```
{
```

```
    "tag_list": ["test_tag_3_Ik0N0", "test_tag_2_Ik0N0"]
}
```

应答示例

```
{
    "ret_code": 0,
    "err_msg": "",
    "seq": 0,
    "result": ""
}
```

账号相关接口
账号绑定与解绑

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。

```
服务地址/v3/device/account/batchoperate
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：异步接口。本接口只负责任务下发，当前不支持实时操作。

参数说明

请求参数

参数名	类型	是否必需	参数说明
operator_type	Integer	是	操作类型参数，取值范围为[1,5]；值对应功能说明如下： 1：Token 追加设置 account（该类型已废弃）。 2：Token 覆盖设置 account。 3：Token 删除绑定的多个 account。 4：Token 删除绑定的所有 account。 5： account 删除绑定的所有 Token。
account_list	Array	否	账号标识集合，当 operator_type = 5 时有效，且必填每个元素包含 account，以及 account_type 字段，加上 account_type 就可以设置类型。 示例如下： [{"account":"926@126.com"},{"account":"1527000000"}, {"account":"2849249569","account_type":9}]
token_list	Array	否	设备标识集合， operator_type = 4 时有效，且必填
token_accounts	Array	否	当 operator_type = 1、2 、3 时有效且必须每次调用最多允许设置20个 Token。每个 Token_account 由1个 Token 和1个 account_list 组成。 具体示例如下： [{"token":"token1","account_list":[{"account":"926@126.com"},

			{"account":"1527000000"}], {"token":"token2","account_list":[{"account":"926@163.com", {"account":"1527000001"}]}]
account_type	Integer	否	账号类型，绑定时可选择账号类型，默认类型为account_type:0，取值可参考 账号类型取值表

说明：

因“Token 追加设置 account”使用率非常低，且容易被开发者误解，从2020年10月26日开始，追加账号接口停止使用。如您此前有使用该接口，该接口功能将变更为“Token 覆盖设置 account”功能。

一个账号类型只能绑定一个账号，绑定多个账号，后面的账号会覆盖前面绑定的账号。

响应参数

参数名	类型	参数说明
ret_code	Integer	错误码，详细参照 错误码对照表。
err_msg	String	错误信息
result	Array	每一个 Token 或账号对应的操作状态码["0","1008006"]

示例说明

请求示例

Token 追加设置 Account（该类型已废弃）：

```
{
  "operator_type": 1,
  "token_accounts": [
      {
          "token": "token1",
          "account_list": [
              {
                  "account": "926@126.com"
              },
              {
                  "account": "1527000000"
              }
          ]
      },
      {
          "token": "token2",
```

```
            "account_list": [
                {
                    "account": "926@163.com"
                },
                {
                    "account": "1527000001"
                }
            ]
        }
    ]
}
```

Token 覆盖绑定 Account：

```
{
    "operator_type": 2,
    "token_accounts": [
        {
            "token": "token1",
            "account_list": [
                {
                    "account": "926@126.com"
                },
                {
                    "account": "1527000000"
                }
            ]
        },
        {
            "token": "token2",
            "account_list": [
                {
                    "account": "926@163.com"
                },
                {
                    "account": "1527000001"
                }
            ]
        }
    ]
}
```

Token 删除绑定 Account：

```
{
    "operator_type": 3,
    "token_accounts": [
```

```
        {
            "token": "token1",
            "account_list": [
                {
                    "account": "926@126.com"
                },
                {
                    "account": "1527000000"
                }
            ]
        },
        {
            "token": "token2",
            "account_list": [
                {
                    "account": "926@163.com"
                },
                {
                    "account": "1527000001"
                }
            ]
        }
    ]
}
```

Token 删除所有绑定 Account：

```
{
  "operator_type": 4,
  "token_list": [
      "token1",
      "token2",
      "token3"
  ]
}
```

Account 删除所有绑定 Token：

```
{
  "operator_type": 5,
  "account_list": [
      {
          "account": "926@126.com"
      },
      {
          "account": "1527000000"
      }
```

```
    ]
}
```

返回示例

Token 删除绑定 Account：

```
{
    "ret_code": 0,
    "err_msg": "NO_ERROR",
    "result": [
        "0"
    ]
}
```

账号设备绑定查询

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。

```
服务地址/v3/device/account/query
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：查询 Account 与 Token 的绑定关系。

参数说明

请求参数

参数名	类型	是否必需	参数说明
operator_type	Integer	是	操作类型： 1：根据 Account 批量查询对应 Token 2：根据 Token 查询 Account
account_list	Array	否	当 operator_type = 1 时有效且必填，待查询 Account 列表每个元素含一组 Account 。 具体示例如下：`[{"account":"account1"},{"account":"account2"}]`
token_list	Array	否	当 operator_type = 2 时有效且必填待查询 Token 的列表

响应参数

参数名	类型	参数说明
retCode	Integer	错误码，详细参照 错误码对照表
errMsg	String	错误信息
account_tokens	Array	Account 到 Token 的映射关系数组。示例如下：`[{"account":"account1","token_list":["token1","token2"]}{"account":"account2","token_list":["token2","token3"]}]`

token_accounts	Array	Token 到 Account 的映射关系数组。示例如下：`[{"token":"token1","account_list":[{"account":"926@126.com"},{"account":"1527000000",}]},{"token":"token2","account_list":[{"account":"926@163.com"},{"account":"1527000001"}]}]`

示例说明

请求示例

批量查询 Account 绑定的 Token 关系

```
{
    "operator_type": 1,
    "account_list": [
        {
            "account": "account1"
        },
        {
            "account": "account2"
        }
    ]
}
```

批量查询 Token 绑定的 Account 关系

```
{
    "operator_type": 2,
    "token_list": [
        "token1",
        "token2"
    ]
}
```

返回示例

批量查询 Account 绑定的 Token

```
{
    "retCode": 0,
    "errMsg": "ok",
    "result": [
        "0",
        "0"
    ],
```

```
    "account_tokens": [
        {
            "account": "account1",
            "token_list": [
                "token1",
                "token2"
            ]
        },
        {
            "account": "account2",
            "token_list": [
                "token2",
                "token3"
            ]
        }
    ]
}
```

批量查询 Token 绑定的 Account

```
{
    "retCode": 0,
    "errMsg": "ok",
    "result": [
        "0",
        "0"
    ],
    "token_accounts": [
        {
            "token": "token1",
            "account_list": [
                {
                    "account": "926@126.com"
                },
                {
                    "account": "1527000000"
                }
            ]
        },
        {
            "token": "token2",
            "account_list": [
                {
                    "account": "926@163.com"
                },
                {
                    "account": "1527000001"
```

```
                    }
                ]
            }
        ]
}
```

统计相关接口
推送统计按天聚合统计

最近更新时间：2024-11-19 14:11:08

接口说明

请求方式：POST。
调用频率限制：200次/小时。

```
服务地址/v3/statistics/get_push_channel_stat_overview
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。
接口功能： 查询某个时间段内每天分推送通道的推送转化汇总数据。

参数说明

请求参数

参数名称	必选	类型	描述
startDate	是	String	查询开始日期 格式：YYYYMMDD 限制：只能查询最近6个月的数据
endDate	是	String	查询截止日期，格式：YYYYMMDD

应答参数

参数名称	类型	描述
retCode	Integer	返回状态码
errMsg	String	错误信息
pushDateChannelStat	Array	返回结果：按天统计的结果数组。 数组单元素代表每天分通道的数据统计结果，由日期 date 和分通道统计数据数组 channelDatas 组成。channelDatas 单元素结构如下。

channelDatas

参数名称	类型	说明
channel	String	推送通道名称 xg：移动推送自建通道 hw：华为通道 xm：小米通道 mz：魅族通道 oppo：OPPO 通道 vivo：vivo 通道 apns：APNs 通道 fcm：FCM 通道 rog：ROG 通道 apns: 苹果通道 iospk: 苹果 pushkit 通道，仅国际站支持 honor：荣耀通道
pushState	Object	推送漏斗统计数据， 数据结构如下见 pushState。

pushState（Android）

参数名称	类型	说明
pushActiveUv	Integer	计划发送 根据推送选定的符合目标人群筛选条件且开启了通知栏状态的90天内有联网的有效设备数。
pushOnlineUv	Integer	实际发送 在计划发送设备数中，实际已经成功下发到厂商通道的或者通过移动推送自建通道对进程在线终端下发成功的有效设备数。
arrivalUv	Integer	抵达设备数（包含移动推送自建通道及厂商通道抵达回执，其中华为、魅族通道抵达回执需要手动添加配置，详情可参考 厂商通道抵达回执获取指南）
verifySvcUv	Integer	抵达设备数（仅移动推送自建通道、ROG 通道、FCM 通道有效。其他厂商通道由移动推送实际发送 pushOnlineUv 指标补齐） **注意：**此字段后续会下线，抵达数据建议参考 arrivalUv 字段。
callbackVerifySvcUv	Integer	厂商通道抵达回执（华为、魅族通道抵达回执需要手动添加配置，详情可参考 厂商通道抵达回执获取指南） **注意：**此字段后续会下线，抵达数据建议参考 arrivalUv 字段。
clickUv	Integer	点击
cleanupUv	Integer	清除

说明：

数组中“all” 通道对应汇总统计数据。
汇总数据中 verifySvcUv（抵达设备），verifyUv（展示），clickUv（点击），cleanupUv（清除）指标只汇总计算了移动推送自建通道数据、ROG 通道数据、FCM 通道数据。
汇总数据中 pushActiveUv（计划发送）, pushOnlineUv（实际发送）汇总计算了移动推送自建通道 + 厂商通道的数据。
汇总数据中 callbackVerifySvcUv（厂商通道抵达回执）汇总计算了 厂商通道 callbackVerifySvcUv（厂商通道抵达回执）+ 移动推送自建通道 verifySvcUv（抵达设备）+ ROG 通道 verifySvcUv（抵达设备）+ FCM 通道 verifySvcUv（抵达设备）。

pushState（iOS）

参数名称	类型	说明
pushActiveUv	Integer	计划发送
pushOnlineUv	Integer	APNs 成功接收
verifySvcUv	Integer	抵达
clickUv	Integer	点击

示例说明

请求示例

```
{
    "startDate": "20200216",
    "endDate": "20200216"
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "NO_ERROR",
    "pushDateChannelStat": [
        {
            "date": "20200216",
            "channelDatas": [
                {
```

```
            "channel": "xm",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "hw",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "oppo",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "xg",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 800,
                "verifySvcUv": 800,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 1000,
                "verifyUv": 800,
```

```
                        "clickUv": 300,
                        "cleanupUv": 500
                    }
                },
                {
                    "channel": "all",
                    "pushState": {
                        "pushActiveUv": 4000,
                        "pushOnlineUv": 3800,
                        "verifySvcUv": 3800,
                        "callbackVerifySvcUv": 2400,
                        "arrivalUv": 3800,
                        "verifyUv": 3800,
                        "clickUv": 300,
                        "cleanupUv": 500
                    }
                }
            ]
        }
    ]
}
```

设备统计查询

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。
调用频率限制：200次/小时。

```
服务地址/v3/statistics/get_device_stat_overview
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。
接口功能：查询应用某个时间段内每天的“日新增设备数”、“日活跃用户数（DAU）”和“历史累计设备数”。

参数说明

请求参数

参数名称	必选	类型	描述
startDate	是	String	查询开始日期 格式：YYYYMMDD 限制：只能查询最近3个月内的数据
endDate	是	String	查询截止日期，格式：YYYYMMDD

应答参数

参数名称	类型	描述
retCode	Integer	返回状态码
errMsg	String	错误信息
getDeviceStatOverviewData	Array	返回结果，getDeviceStatOverviewData 结构变量见下表

getDeviceStatOverviewData

参数名称	类型	说明
date	Integer	数据日期

accuUv	Integer	累积设备量
newUv	Integer	日新增设备量
activeUv	Integer	日在线设备峰值

示例说明

请求示例

```
{
    "startDate": "20190724",
    "endDate": "20190726"
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "",
    "getDeviceStatOverviewData": [
        {
            "date": 20190724,
            "accuUv": 0,
            "newUv": 0,
            "activeUv": 0
        },
        {
            "date": 20190725,
            "accuUv": 0,
            "newUv": 5,
            "activeUv": 0
        },
        {
            "date": 20190726,
            "accuUv": 5,
            "newUv": 0,
            "activeUv": 2
        }
    ]
}
```

特定时间段所有推送信息查询

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST

调用频率限制：200次/小时。

```
服务地址/v3/statistics/get_push_record
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：查询特定时间段内所有任务的基本信息和设置。

参数说明

请求参数

参数名称	必选	类型	描述
startDate	是	String	查询起始日期， 格式：YYYY-MM-DD 查询限制：当前日期1个月内
endDate	是	String	查询截止日期，格式：YYYY-MM-DD
msgType	否	String	消息类型： notify：通知 message：静默消息
pushType	否	String	推送类型： all：全推 tag：标签推 token：设备列表/设备单推 account：账号列表/账号单推
offset	否	Integer	分页查询起始偏移
limit	否	Integer	分页查询每页消息数 （最大限制为200）

应答参数

参数名称	类型	描述
retCode	Integer	返回状态码
errMsg	String	错误信息
pushRecordData	Array	返回结果，pushRecordData 结构变量见下表
count	Integer	符合条件记录数

pushRecordData

参数名称	类型	说明	取值说明
date	String	推送时间	格式：YYYY-MM-DD hh:mm:ss
pushId	String	消息 ID	-
title	String	推送标题	-
content	String	推送内容	-
status	String	推送状态	PUSH_INIT //任务已创建 PUSH_WAIT// 等待任务被调度 PUSH_STARTED// 推送中 PUSH_FINISHED// 推送完成 PUSH_FAILED//推送失败 PUSH_CANCELED// 用户取消推送 PUSH_DELETED// 推送被删除 PUSH_REVOKED//推送已被撤回 PUSH_COLLAPSED//推送已被覆盖 PUSH_DELETED_PUSH_MSG//推送被终止
pushType	String	推送目标	all //全推 tag //标签推送 token_list //设备列表 account_list //账号列表 package_account_push //号码包推送
messageType	String	推送类型	notify //通知 message //消息
environment	String	推送环境	product //生产环境 dev //开发环境
expireTime	uint32	过期时间	单位 s
xgMediaResources	String	富媒体信息	-

multiPkg	Boolean	是否多包名推送	true //开启多包名推送 false //关闭多包名推送
targetList	Array(String)	推送账号或推送设备列表	pushType 为 token_list 或 account_list 时有效
collapseID	Integer	消息覆盖id	pushType 为 all、tag、package_account_push 时有效

tagSet	Object	标签设置	pushType 为 tag 时有效
			数据结构：
			`{`
			`"op":"OR", //标签间逻辑操作`
			`"tagWithType":[`
			`{ "tagTypeName":"xg_user_define", //标签类型`
			`"tagValue":"test68" //标签值}`
			`]`
			`}`

uploadId	Intege	号码包 ID	pushType 为 package_account_push 时有效
pushConfig	Object	推送配置信息	"Android"： Android 推送相关配置信息，具体参考下述代码 "iOS"：iOS 推送相关配置信息， 具体参考下述代码

配置信息

Android 推送配置信息

```
"android": {
        "ring": 1, //响铃
        "vibrate": 1,//震动
        "lights": 1,//呼吸灯
        "clearable": 1, //是否可清除
        "action": {
            "action_type": 3,// 动作类型，1，打开activity或app本身；2，打开浏览器；3，打开
            "intent": "" //SDK版本需要大于等于1.0.9，然后在客户端的intent配置data标签，并设
        },
      "custom_content":"{}"
    }
```

iOS 推送配置信息

```
"ios":{
        "aps": {
            "alert": {
                "subtitle": "my subtitle"
            },
            "badge_type": 5, //App显示的角标数(可选) -2 自增, -1 不变,
            "category": "INVITE_CATEGORY",
            "sound":"default", //缺省代表默认音效
            "mutable-content":1
        },
```

示例说明

请求示例

```
{
    "limit": 50,
    "startDate": "2019-07-01",
    "endDate": "2019-08-01",
    "msgType": "notify",
    "pushType": "all",
    "offset": 0
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "NO_ERROR",
    "count": 126,
    "pushRecordData": [
        {
            "date": "2019-11-18 11:26:54",
            "pushId": "12",
            "title": "测试标题",
            "content": "测试日志",
            "status": "PUSH_FINISHED",
            "pushType": "all",
            "targetList": null,
            "tagSet": null,
            "uploadId": 0,
```

```
            "groupId": "",
            "expireTime": 43200,
            "messageType": "notify",
            "xgMediaResources": "",
            "environment": "product",
            "pushConfig": {
                "android": {
                    "n_id": 0,
                    "builder_id": 0,
                    "ring": 1,
                    "ring_raw": "",
                    "vibrate": 1,
                    "lights": 1,
                    "clearable": 1,
                    "icon_type": 0,
                    "icon_res": "",
                    "style_id": 0,
                    "small_icon": "",
                    "action": {
                        "action_type": 3,
                        "activity": "",
                        "aty_attr": null,
                        "browser": null,
                        "intent": ""
                    },
                    "custom_content": ""
                },
                "ios": null,
                "iot": null
            },
            "multiPkg": true,
            "source": "api"
        }
    ]
}
```

单个任务推送信息查询

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。
调用频率限制：200次/小时。

```
服务地址/v3/statistics/get_push_record
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。
接口功能：根据任务的 pushid 查询该条任务的基本信息和设置。

参数说明

请求参数

参数名称	必选	类型	描述
pushId	是	String	推送任务 ID

应答参数

参数名称	类型	描述
retCode	Integer	返回状态码
errMsg	String	错误信息
pushRecordData	Array	返回结果，pushRecordData 结构变量见下表

pushRecordData

参数名称	类型	说明	取值说明
date	String	推送时间	格式：YYYY-MM-DD hh:mm:ss
pushId	String	消息ID	-
title	String	推送标题	-

content	String	推送内容	-
status	String	推送状态	PUSH_INIT //任务已创建 PUSH_WAIT// 等待任务被调度 PUSH_STARTED// 推送中 PUSH_FINISHED// 推送完成 PUSH_FAILED//推送失败 PUSH_CANCELED// 用户取消推送 PUSH_DELETED// 推送被删除 PUSH_REVOKED//推送已被撤回 PUSH_COLLAPSED//推送已被覆盖 PUSH_DELETED_PUSH_MSG//推送被终止
pushType	String	推送目标	all //全推 tag //标签推送 token_list //设备列表 account_list //账号列表 package_account_push //号码包推送
messageType	String	推送类型	notify //通知 message //消息
environment	String	推送环境	product //生产环境 dev //开发环境
expireTime	Integer	过期时间	单位 s
xgMediaResources	String	富媒体信息	-
multiPkg	Boolean	是否多包名推送	true //开启多包名推送 false //关闭多包名推送
targetList	Array(String)	推送账号或推送设备列表	pushType 为 token_list 或 account_list 时有效
collapseID	Integer	消息覆盖 ID	pushType 为 all、tag、package_account_push 时有效

tagSet	Object	标签设置	pushType 为 tag 时有效
			数据结构：
			`{`
			`"op":"OR", //标签间逻辑操作`
			`"tagWithType":[`
			`{ "tagTypeName":"xg_user_define", //标签类型`
			`"tagValue":"test68" //标签值}`

			] }
uploadId	Integer	号码包 ID	pushType 为 package_account_push 时有效
pushConfig	Object	推送配置信息	"Android"：Android 推送相关配置信息，具体参考下述代码 "iOS"：iOS 推送相关配置信息， 具体参考下述代码

配置信息

Android 推送配置信息

```
"android": {
        "ring": 1, //响铃
        "vibrate": 1,//震动
        "lights": 1,//呼吸灯
        "clearable": 1, //是否可清除
        "action": {
            "action_type": 3,// 动作类型，1，打开activity或app本身；2，打开浏览器；3，打开
            "intent": "" //SDK版本需要大于等于1.0.9，然后在客户端的intent配置data标签，并设
        },
      "custom_content":"{}"
    }
    ```
iOS 推送配置信息

```json
"ios":{
        "aps": {
            "alert": {
                "subtitle": "my subtitle"
            },
            "badge_type": 5, //App显示的角标数(可选) -2 自增，-1 不变，
            "category": "INVITE_CATEGORY",
            "sound":"default",//缺省代表默认音效
            "mutable-content":1
        },
          }
```

示例说明

请求示例

```
{
    "pushId": "133703"
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "NO_ERROR",
    "pushRecordData": [
        {
            "date": "2019-07-25 20:06:28",
            "pushId": 133703,
            "title": "1",
            "content": "2",
            "status": "PUSH_FINISHED",
            "pushType": "tag",
            "targetList": null,
            "tagSet": {
                "op": "OR",
                "tagWithType": [
                    {
                        "tagTypeName": "xg_user_define",
                        "tagValue": "test68"
                    }
                ]
            },
            "uploadId": 0,
            "expireTime": 86400,
            "messageType": "notify",
            "xgMediaResources": "",
            "environment": "product",
            "collapseID": 0,
            "pushConfig": {
                "android": {
                    "ring": 1,
                    "vibrate": 0,
                    "lights": 1,
                    "clearable": 1,
                    "action": {
                        "action_type": 1
                    },
                    "custom_content": "{}"
                },
```

```
                "ios": null
            },
            "multiPkg": false
        }
    ]
}
```

单条任务推送统计信息查询

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。
调用频率限制：200次/小时。

```
服务地址/v3/statistics/get_push_task_stat_channel
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。
接口功能：查询每个推送任务的详细统计，包含所有通道信息及汇总结果。pushStatDataAll 里的通道类型会变化，根据 iOS/Android 和推送通道的不同而不同。

参数说明

请求参数

参数名称	必选	类型	描述
pushId	是	String	推送任务 ID，限制查询当前日期起1个月内的推送任务

应答参数

参数名称	类型	描述
retCode	Integer	返回状态码
errMsg	String	错误信息
pushStatDataAll	Array	pushStatDataAll 结构变量见下表

pushStatDataAll

参数名称	类型	描述
channel	String	推送通道名称 xg：TPNS 通道 hw：华为通道 xm：小米通道 mz：魅族通道

		oppo：OPPO 通道 vivo：vivo 通道 apns：APNs 通道 fcm：FCM 通道 rog：ROG 通道 apns: 苹果通道 iospk: 苹果 pushkit 通道，仅国际站支持 honor：荣耀通道 all：通道汇总
pushState	Object	pushState 结构变量见下表

PushState（Android）

参数名称	类型	说明
pushActiveUv	Integer	计划发送数 根据推送选定的符合目标人群筛选条件且开启了通知栏状态的90天内有联网的有效设备数。
pushOnlineUv	Integer	实际发送数 在计划发送设备数中，实际已经成功下发到厂商通道的或者通过 TPNS 通道对进程在线终端下发成功的有效设备数。
arrivalUv	Integer	抵达设备（包含 TPNS 通道及厂商通道抵达回执，其中华为、魅族通道抵达回执需要手动添加配置，详情可参考 厂商通道抵达回执获取指南）
verifySvcUv	Integer	抵达设备（仅 TPNS 通道、ROG 通道、FCM 通道有效。其他厂商通道由 TPNS 实际发送 pushOnlineUv 指标补齐） **注意：**此字段后续会下线，抵达数据建议参考 arrivalUv 字段。
callbackVerifySvcUv	Integer	厂商通道抵达回执（华为、魅族通道抵达回执需要手动添加配置，详情可参考 厂商通道抵达回执获取指南） **注意：**此字段后续会下线，抵达数据建议参考 arrivalUv 字段。
verifyUv	Integer	展示（已废弃，后续会下线此字段）
clickUv	Integer	点击
cleanupUv	Integer	清除

说明：

数组中“all” 通道对应汇总统计数据。

汇总数据中verifySvcUv（抵达设备），verifyUv（展示），clickUv（点击），cleanupUv（清除）指标只汇总计算了 TPNS 通道数据、ROG 通道数据、FCM 通道数据。

汇总数据中 pushActiveUv（计划发送）， pushOnlineUv（实际发送）汇总计算了 TPNS 通道 + 厂商通道的数据。
汇总数据中 callbackVerifySvcUv（厂商通道抵达回执）汇总计算了 厂商通道 callbackVerifySvcUv（厂商通道抵达回执）+ TPNS 通道 verifySvcUv（抵达设备）+ ROG 通道 verifySvcUv（抵达设备）+ FCM 通道 verifySvcUv（抵达设备）。

pushState（iOS 与 macOS）

参数名称	类型	说明
pushActiveUv	Integer	计划发送
pushOnlineUv	Integer	APNs 成功接收
verifySvcUv	Integer	抵达
clickUv	Integer	点击

示例说明

请求示例

```
{
   "pushId": "130248"
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "NO_ERROR",
    "pushStatDataAll": [
        {
            "channel": "xm",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
```

```
        {
            "channel": "mz",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "vivo",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "hw",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "xg",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 800,
                "verifySvcUv": 800,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 1000,
```

```
                "verifyUv": 800,
                "clickUv": 300,
                "cleanupUv": 500
            }
        },
        {
            "channel": "oppo",
            "pushState": {
                "pushActiveUv": 1000,
                "pushOnlineUv": 1000,
                "verifySvcUv": 1000,
                "callbackVerifySvcUv": 800,
                "arrivalUv": 1000,
                "verifyUv": 1000,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "fcm",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "rog",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "all",
            "pushState": {
```

```
                "pushActiveUv": 6000,
                "pushOnlineUv": 5800,
                "verifySvcUv": 5800,
                "callbackVerifySvcUv": 4000,
                "arrivalUv": 5800,
                "verifyUv": 5800,
                "clickUv": 300,
                "cleanupUv": 500
            }
        }
    ]
}
```

多条任务统计数据聚合查询

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。
调用频率限制：200次/小时。

```
服务地址/v3/statistics/get_push_group_stat_channel
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。
接口功能：可根据 GroupID 或 planId 来查询**最近7天内相同 GroupID 或 planId** 的所有已推送任务的分推送渠道的聚合统计数据。
说明：
`GroupID` 将逐渐废弃，推荐使用 `planId` 查询聚合统计数据。

参数说明

请求参数

参数名称	必选	类型	描述
planId	是	String	多条任务聚合统计的字段，对应推送参数中的 “plan_id”
groupId	是	String	多条任务聚合统计的字段，对应推送参数中的 “group_Id” **注意**：该参数将逐渐废弃，推荐使用 planId 查询聚合统计数据
startDate	是	String	查询开始日期 格式：YYYYMMDD 限制：只能聚合统计最近7天的推送任务
endDate	是	String	查询截止日期，格式：YYYYMMDD

说明：
若请求参数中同时存在 planId 和 groupId，则默认按 planId 查询。

应答参数

参数名称	类型	描述

retCode	Integer	返回状态码
errMsg	String	错误信息
pushStatDataAll	Array	pushStatDataAll 结构变量见下表

pushStatDataAll

参数名称	类型	描述
channel	String	推送通道名称 xg：TPNS 通道 hw：华为通道 xm：小米通道 mz：魅族通道 oppo：OPPO 通道 vivo：vivo 通道 apns：APNs 通道 fcm：FCM 通道 rog：ROG 通道 iospk: 苹果 pushkit 通道，仅国际站支持 apns: 苹果通道 honor：荣耀通道 all：通道汇总
pushState	Object	pushState 结构变量见下表

pushState（Android）

参数名称	类型	说明
pushActiveUv	Integer	计划发送数 根据推送选定的符合目标人群筛选条件且开启了通知栏状态的90天内有联网的有效设备数。
pushOnlineUv	Integer	实际发送数 在计划发送设备数中，实际已经成功下发到厂商通道的或者通过 TPNS 通道对进程在线终端下发成功的有效设备数。
arrivalUv	Integer	抵达设备数（包含 TPNS 通道及厂商通道抵达回执，其中华为、魅族通道抵达回执需要手动添加配置，详情可参见 厂商通道抵达回执获取指南）
verifySvcUv	Integer	抵达设备数（仅 TPNS 通道、ROG 通道、FCM 通道有效。其他厂商通道由 TPNS 实际发送 pushOnlineUv 指标补齐）。 **注意**：此字段后续将下线，抵达数据建议参考 arrivalUv 字段。

callbackVerifySvcUv	Integer	厂商通道抵达回执（华为、魅族通道抵达回执需要手动添加配置，详情可参见 厂商通道抵达回执获取指南）。 **注意**：此字段后续将下线，抵达数据建议参考 arrivalUv 字段。
verifyUv	Integer	展示（已废弃，后续会下线此字段）
clickUv	Integer	点击
cleanupUv	Integer	清除

说明：

数组中“all”通道对应汇总统计数据。

汇总数据中 verifySvcUv（抵达设备），verifyUv（展示），clickUv（点击），cleanupUv（清除）指标只汇总计算了 TPNS 通道数据、ROG 通道数据、FCM 通道数据。

汇总数据中 pushActiveUv（计划发送）, pushOnlineUv（实际发送）汇总计算了 TPNS 通道 + 厂商通道的数据。

汇总数据中 callbackVerifySvcUv（厂商通道抵达回执）汇总计算了 厂商通道 callbackVerifySvcUv（厂商通道抵达回执）+ TPNS 通道 verifySvcUv（抵达设备）+ ROG 通道 verifySvcUv（抵达设备）+ FCM 通道 verifySvcUv（抵达设备）。

pushState（iOS）

参数名称	类型	说明
pushActiveUv	Integer	计划发送
pushOnlineUv	Integer	APNs 成功接收
verifySvcUv	Integer	抵达
clickUv	Integer	点击

示例说明

请求示例

```
{
    "groupid": "pt:testGroup",
    "startDate": "20190912",
    "endDate": "20190912"
}
```

应答示例


```
{
    "retCode": 0,
    "errMsg": "NO_ERROR",
    "pushStatDataAll": [
        {
            "channel": "fcm",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "rog",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "hw",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "xm",
```

```
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "oppo",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "vivo",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
                "cleanupUv": 0
            }
        },
        {
            "channel": "mz",
            "pushState": {
                "pushActiveUv": 0,
                "pushOnlineUv": 0,
                "verifySvcUv": 0,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 0,
                "clickUv": 0,
```

```
                "cleanupUv": 0
            }
        },
        {
            "channel": "xg",
            "pushState": {
                "pushActiveUv": 4641,
                "pushOnlineUv": 4641,
                "verifySvcUv": 4641,
                "callbackVerifySvcUv": 0,
                "arrivalUv": 0,
                "verifyUv": 4639,
                "clickUv": 3818,
                "cleanupUv": 4200
            }
        },
        {
            "channel": "all",
            "pushState": {
                "pushActiveUv": 4641,
                "pushOnlineUv": 4641,
                "verifySvcUv": 4641,
                "callbackVerifySvcUv": 4641,
                "arrivalUv": 4641,
                "verifyUv": 4639,
                "clickUv": 3818,
                "cleanupUv": 4200
            }
        }
    ]
}
```

根据 Token 查询当天所有推送任务

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。
调用频率限制：200次/小时。

```
服务地址v3/toolbox/getPushListByToken
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。
接口功能：查询应用某个 Token 一天的推送数据。

参数说明

请求参数

参数名称	必选	类型	描述
token	是	String	设备 Token

应答参数

参数名称	类型	描述
retCode	Integer	返回状态码
errMsg	String	错误信息
pushTaskList	pushTask	推送参数

pushTask

参数名称	类型	说明
pushId	Integer	推送任务 id
pushTime	Integer	推送时间戳
pushTargetType	Array	推送类型

示例说明

请求示例

```
{
   "token":"05a305f6b71abb3a6b8c759fd1bc56b4bb44"
}
```

应答示例

```
{
   "retCode": 0,
    "errMsg": "NO_ERROR",
    "pushTaskList": [
        {
            "pushId": 589840563,
            "pushTime": 1651662600,
            "pushTargetType": "TAG_PUSH"
        },
        {
            "pushId": 590166744,
            "pushTime": 1651741604,
            "pushTargetType": "TAG_PUSH"
        },
        {
            "pushId": 590178525,
            "pushTime": 1651742807,
            "pushTargetType": "TAG_PUSH"
        },
        {
            "pushId": 590179538,
            "pushTime": 1651742914,
            "pushTargetType": "TAG_PUSH"
        },
        {
            "pushId": 590179672,
            "pushTime": 1651742928,
            "pushTargetType": "TAG_PUSH"
        },
        {
            "pushId": 590235722,
            "pushTime": 1651750200,
            "pushTargetType": "TAG_PUSH"
        }
        ]
```

```
}
```

用户属性相关接口

最近更新时间：2024-01-17 14:26:44

接口说明

请求方式：POST。

```
服务地址/v3/device/set_custom_attribute
```

接口服务地址与服务接入点一一对应，请选择与您的应用服务接入点对应的 服务地址。

接口功能：用于 token 级别的个性化属性配置，包括增加、删除、更新、查询功能。

参数说明

请求参数

参数名称	是否必填	类型	描述
cmd	是	Integer	操作类型： 1：新增属性 2：更新属性 3：删除属性 4：删除所有属性 5：查询属性
token	是	String	TPNS 为设备分配的唯一 ID 获取 Token交互建议（Android） 获取 Token 交互建议（iOS）
attributeInfo	当 cmd=1，2，3时必填	Map	属性详情，参考下方 attributeMap 描述
attributeMap	当 cmd=1，2，3时必填	Map	属性详情： key 为 属性名，长度限制为50字节 **注意**：需要已经在【控制台】>【配置管理】>【用户属性管理】中创建属性，否则会被过滤掉，并返回 invalidAttribute。 value 为属性值，长度限制为50字节

返回参数

参数名称	是否必定返回	类型	描述
retCode	是	Integer	错误码，详细参照 错误码对照表 。
errMsg	是	String	请求出错时的错误信息。
attributeInfo	cmd = 5	Map	属性详情。
invalidAttribute	属性无效时	Array	无效属性详情。

示例说明

新增属性

请求示例

为单个 token 增加3个属性。

```
{
    "cmd": 1,
    "token": "04cac74a714f61bf089987a986363d88****",
    "attributeInfo": {
         "attributeMap": {
            "age": "100",
            "name": "Ming",
            "high": "2.66"
        }
    }
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "success",
    "invalidAttribute": [
        "high"    // 控制台上没有对应的 key 值
    ]
}
```

更新属性

请求示例

更新属性“name”对应的值“workman”。

```
{
    "cmd": 2,
    "token": "04cac74a714f61bf089987a986363d88****",
    "attributeInfo": {
         "attributeMap": {
            "name": "workman"
        }
    }
}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "success"
}
```

删除属性

请求示例

删除属性“name”对应的值“workman”。

```
{
    "cmd": 3,
    "token": "04cac74a714f61bf089987a986363d88****",
    "attributeInfo": {
         "attributeMap": {
            "name": "workman"
        }
    }
}
```

应答示例

```
{
```

```
    "retCode": 0,
    "errMsg": "success"
}
```

删除所有属性

请求示例

删除该 token 下的所有属性。

```
{
    "cmd": 4,
    "token": "04cac74a714f61bf089987a986363d88****"

}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "success"
}
```

查询属性

请求示例

查询该 token 下的属性详情。

```
{
    "cmd": 5,
    "token": "04cac74a714f61bf089987a986363d88****"

}
```

应答示例

```
{
    "retCode": 0,
    "errMsg": "success",
    "attributeInfo": {
        "attributeMap": {
            "nickname": "workman"
        }
    }
```

```
}
```

服务端错误码

最近更新时间：2024-01-17 14:26:44

移动推送Rest API 的通用错误主要如下：

错误码	错误英文描述	错误中文描述	反馈接口
1008001	paramter parse error	参数解析错误	通用接口
1008002	missing parameter	必填参数缺失	通用接口
1008003	auth failure	认证鉴权失败	通用接口
1008004	invoke service error	调用内部服务失败	通用接口
1008006	invalid token	Token 无效，请检查	标签操作，账号操作相关接口
1008007	invalid parameter	参数校验失败	通用接口
1008011	upload file error	文件上传失败	证书上传接口，号码包上传接口
1008012	upload empty file	上传文件为空	证书上传接口，号码包上传接口
1008013	parse certificate error	证书解析失败	证书上传接口
1008015	push id not exist	推送任务 ID 不存在	统计查询接口
1008016	date param format error	日期时间参数格式不对	统计查询接口
1008019	content may be evil	被内容安全服务判定不和谐	推送接口
1008020	cert bundleId check failed	证书包名校验失败	证书上传接口
1008021	not correct p12 file	p12 证书格式内容校验失败	证书上传接口
1008022	p12 password is not correct	p12 证书密码不对	证书上传接口
1008023	ios push cert was expired for the environment: PRODUCT	推送证书校验错误，请检查证书是否在有效期内	推送接口

1008025	create app , platform exist	创建应用失败，产品下已存在该平台的应用	应用创建接口
1008026	batch op，partly error	批量操作，部分失败	账号绑定，查询接口
1008027	batch op，total error	批量操作，全部失败	账号绑定，查询接口
1008028	request too fast	超出限频： 调用统计接口超频率 全量推送、标签推送和号码包推送超限额，参见 推送接口 的注意事项	统计接口，推送接口
1008029	token is not valid	Token 校验非法	单推接口，账号绑定接口，标签绑定接口
1008030	app not paid	App 未付费	通用错误
1008031	app was expired	App 资源已销毁	通用错误
10110008	no token，no account	查询的 Token，账号不存在	账号绑定，查询接口
10010005	internal error	推送目标不存在	推送接口
10010012	illegal push time	非法的推送时间	推送接口
10010018	Push repeat	重复推送	推送接口，全量、tag 推送
1008035	accessId is not correct or domain not correct	请确认应用 accessid 是否正确，或应用服务接入点与填写的域名地址不匹配，参见 服务地址	通用错误

服务端 SDK

最近更新时间：2024-01-17 14:26:44

除了 REST API 之外，移动推送 TPNS 还提供主流的服务端接口 SDK。

官方维护

官方维护的版本在**腾讯工蜂-tpns**上以开源的形式发布。
官方插件地址包含：源码、推送示例代码，请开发者参阅。

项目	地址
Java	官方地址
Python	官方地址
PHP	官方地址
C#	官方地址
C++	官方地址
Golang	官方地址

API（Java）

最近更新时间：2024-01-17 14:26:44

SDK 说明

本 SDK 提供移动推送服务端接口的 Java 封装，与移动推送后台通信。使用时引用 XingeApp 包即可，本 SDK 封装的主要是 V3 推送相关接口。

集成方式

Maven 依赖引用方式：

```
<dependency>
                <groupId>io.github.tpnsPush</groupId>
                <artifactId>xinge</artifactId>
                <version>1.2.4.11</version>
</dependency>
```

注意：

groupId从1.2.4.11版本起有变更。

使用方法

XingeApp 接口说明

该类提供与移动推送后台交互的接口。由 XingeApp.Builder 进行构建，对应参数如下

参数名	类型	必需	默认值	参数描述
appId	String	是	无	推送目标 AccessID，可在 产品管理 页面获取。
secretKey	String	是	无	推送密钥，可在 产品管理 > **应用配置**页面获取。
proxy	Proxy	否	Proxy.NO_PROXY	如果需要设置代理可以设定该参数。
connectTimeOut	Integer	否	10s	链接超时时间设置。
readTimeOut	Integer	否	10s	请求超时时间设置。

domainUrl	String	否	https://openapi.xg.qq.com/	请求接口服务域名地址，默认为请求信鸽平台的接口地址。使用时需要根据您产品的服务接入点选择 请求服务地址。

示例

```
XingeApp xingeApp = new XingeApp.Builder()
        .appId(appid)
        .secretKey(secretKey)
        .domainUrl(“https://api.tpns.tencent.com/”)
        .build();

PushAppRequest pushAppRequest = new PushAppRequest();
//完善PushAppRequest  消息
...
JSONObject ret =  xingeApp.pushApp(pushAppRequest );
```

pushAppRequest 接口说明

该类提供封装好的推送消息体，各参数说明及使用方法可参考 推送接口说明。

示例

说明：

`XingeAppSimple` 中包含推送、绑定、解绑等接口的示例。

Android 单设备推送示例

```
public JSONObject pushTokenAndroid() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.token);
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("title");
        message.setContent("content");
        pushAppRequest.setMessage(message);
        MessageAndroid messageAndroid = new MessageAndroid();
        message.setAndroid(messageAndroid);
        ArrayList<String> tokenList = new ArrayList();
        tokenList.add("04cac74a714f61bf089********63d880993");
        pushAppRequest.setToken_list(tokenList);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

Android 单账号推送示例

```
public JSONObject pushAccountAndroid() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.account);
        pushAppRequest.setPlatform(Platform.android);
        pushAppRequest.setMessage_type(MessageType.notify);
        pushAppRequest.setAccount_push_type(1);
        Message message = new Message();
        message.setTitle("title");
        message.setContent("content");
        MessageAndroid messageAndroid = new MessageAndroid();
        message.setAndroid(messageAndroid);
        pushAppRequest.setMessage(message);
        ArrayList<String> accountList = new ArrayList();
        accountList.add("123");
        pushAppRequest.setAccount_list(accountList);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

Android 标签推送示例

```
public JSONObject pushTagAndroid() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.tag);
        pushAppRequest.setPlatform(Platform.android);
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("title");
        message.setContent("content");
        MessageAndroid messageAndroid = new MessageAndroid();
        message.setAndroid(messageAndroid);
        pushAppRequest.setMessage(message);
        ArrayList<String> tagList = new ArrayList();
        tagList.add("tag");
        TagListObject tagListObject = new TagListObject();
        tagListObject.setTags(tagList);
        tagListObject.setOp(OpType.OR);
        pushAppRequest.setTag_list(tagListObject);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

Android 全部设备推送示例

```
public JSONObject pushAllAndroid() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.all);
        pushAppRequest.setPlatform(Platform.android);
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("title");
        message.setContent("content");
        MessageAndroid messageAndroid = new MessageAndroid();
        message.setAndroid(messageAndroid);
        pushAppRequest.setMessage(message);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

iOS 单设备推送示例

```
public JSONObject pushTokenIos(){
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.token);
        pushAppRequest.setEnvironment(Environment.valueOf("dev"));
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("title");
        message.setContent("content");
        MessageIOS messageIOS = new MessageIOS();
        Alert alert = new Alert();
        Aps aps = new Aps();
        aps.setAlert(alert);
        messageIOS.setAps(aps);
        message.setIos(messageIOS);
        pushAppRequest.setMessage(message);
        ArrayList<String> tokenList = new ArrayList<String>();
        tokenList.add("0250df875c93c55********536b54fc1c49f");
        pushAppRequest.setToken_list(tokenList);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

iOS 单账号推送示例

```
 public JSONObject pushAccountIos() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.account);
        pushAppRequest.setEnvironment(Environment.valueOf("dev"));
```

```
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("账号推送");
        message.setContent("content");
        MessageIOS messageIOS = new MessageIOS();
        Alert alert = new Alert();
        Aps aps = new Aps();
        aps.setAlert(alert);
        messageIOS.setAps(aps);
        message.setIos(messageIOS);
        pushAppRequest.setMessage(message);
        ArrayList<String> accountList = new ArrayList();
        accountList.add("1122");
        pushAppRequest.setAccount_list(accountList);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

iOS 标签推送示例

```
public JSONObject pushTagIos() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.tag);
        pushAppRequest.setEnvironment(Environment.valueOf("dev"));
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("标签推送");
        message.setContent("content");
        MessageIOS messageIOS = new MessageIOS();
        Alert alert = new Alert();
        Aps aps = new Aps();
        aps.setAlert(alert);
        messageIOS.setAps(aps);
        message.setIos(messageIOS);
        pushAppRequest.setMessage(message);
        ArrayList<String> tagList = new ArrayList();
        tagList.add("1122");
        TagListObject tagListObject = new TagListObject();
        tagListObject.setTags(tagList);
        tagListObject.setOp(OpType.OR);
        pushAppRequest.setMessage(message);
        pushAppRequest.setTag_list(tagListObject);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

iOS 全部设备推送示例

```
public JSONObject pushAllIos() {
        PushAppRequest pushAppRequest = new PushAppRequest();
        pushAppRequest.setAudience_type(AudienceType.all);
        pushAppRequest.setEnvironment(Environment.valueOf("dev"));
        pushAppRequest.setMessage_type(MessageType.notify);
        Message message = new Message();
        message.setTitle("全量推送");
        message.setContent("content");
        MessageIOS messageIOS = new MessageIOS();
        Alert alert = new Alert();
        Aps aps = new Aps();
        aps.setAlert(alert);
        messageIOS.setAps(aps);
        message.setIos(messageIOS);
        pushAppRequest.setMessage(message);
        return this.xingeApp.pushApp(pushAppRequest);
    }
```

推送应答示例

```
{"result":"{}","environment":"","push_id":"13282451386901258 24","err_msg":"NO_ERROR
```

服务端返回码

ret_code 含义可参考 服务端错误码。

常见问题

接口返回错误码10101或403是什么原因，如何解决？

请检查应用 AccessID 与 SecretKey 是否匹配，domainUrl 与产品 服务接入点 是否匹配。

接口返回错误码1008007，参数校验失败如何解决？

请参考 推送示例，检查参数填写是否缺失或字段类型填写有误。

是否有其它开发语言的 SDK ？

更多服务端 SDK 可前往 SDK 下载 页面获取。

推送接口返回 `Peer certificate cannot be authenticated with given CA certificates` ，如何解决？

此问题原因是ca证书过期，进入证书目录，通过 openssl 命令进行查看到期时间：

```
# openssl x509 -in signed.crt -noout -dates
```

`signed.crt` 修改为您自己服务端上的证书名称。