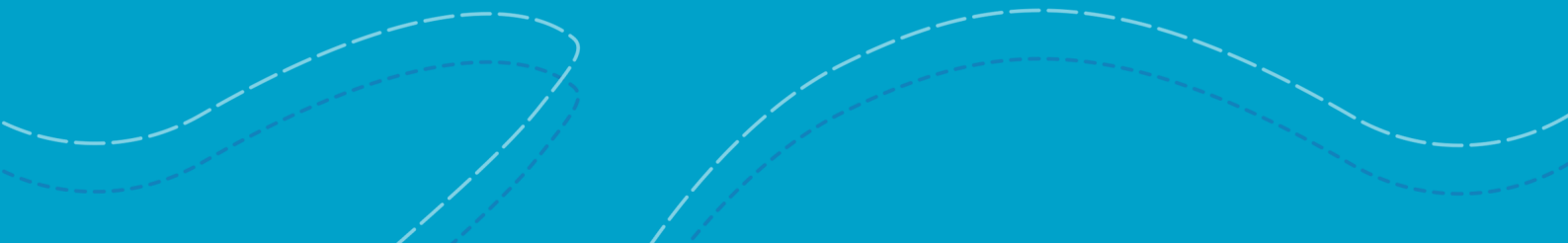
移动推送

OpenAPI文档



OpenAPI文档

简介

欢迎使用阿里云移动推送服务，用户可以使用本文档介绍的API对移动推送服务进行相关操作。

使用前必读：移动推送名词解释&约束

API更新历史

- 最新API版本号：2015-08-27

更新时间	更新说明
2015-12-08	提供消息/通知推送接口
2015-12-29	增加.NET,PHP版SDK与使用示例
2016-01-07	接口参数优化
2016-01-21	增加设备状态查询接口
2016-03-28	增加Tag相关操作接口
2016-04-18	推送接口添加按alias推送

API概览

API	说明
PushMessageToAndroid	发送消息给android
PushNoticeToAndroid	发送通知给android
PushMessageToiOS	发送消息给iOS
PushNoticeToiOS	发送通知给iOS
Push	推送高级接口（**通知/消息** 均可发送）
GetDeviceInfos	查询设备状态

公共参数获取

- AccessKeyId和AccessKeySecret：在阿里云官网控制台获取
- AppKey：在移动推送控制台的APP列表页，点击应用证书获取

SDK获取

- aliyun-java-sdk-push
- aliyun-php-sdk-push
- aliyun-python-sdk-push
- aliyun-nodejs-sdk-push
- aliyun-net-sdk-push

Java SDK 可直接使用以下Maven引用

```
<ol><li> <dependency></li><li> <groupId> com.aliyun </groupId></li><li> <artifactId> aliyun-java-sdk-push
</artifactId></li><li> <version> 2.1.0 </version></li><li> </dependency></li></ol>
```

注：maven官方库更新需要几天时间才能同步，如果maven方式引用失败，请先用上面的git地址下载源码本地编译最新版本jar包。

Demo

- Java Demo
- PHP Demo
- Python Demo
- NodeJS Demo
- .Net Demo

API调用方式

请求结构

服务地址

移动推送API的服务接入地址为：cloudpush.aliyuncs.com

通信协议

支持通过HTTP或HTTPS通道进行请求通信，为了获得更高的安全性，推荐您使用HTTPS通道发送请求。

请求方法

支持HTTP GET方法发送请求，这种方式下请求参数需要包含在请求的URL中。

请求参数

每个请求都需要指定要执行的操作，即Action参数（例如Push），以及每个操作都需要包含的公共请求参数和指定操作所特有的请求参数。

字符编码

请求及返回结果都使用UTF-8字符集进行编码。

公共参数

公共请求参数

公共请求参数是指每个接口都需要使用到的请求参数。

名称	类型	是否必须	描述
Format	String	否	返回值的类型，支持JSON与XML，默认为XML。
RegionId	String	是	当前请设置为cn-hangzhou
Version	String	是	API版本号，为日期形式YYYY-MM-DD，本版本对应为2015-08-27。
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID。
Signature	String	是	签名结果串，关于签名的计算方法，请参见签名机制。
SignatureMethod	String	是	签名方式，目前支持HMAC-SHA1。
Timestamp	String	是	请求的时间戳。日期格

			式按照ISO8601标准表示，并需要使用UTC时间，格式为YYYY-MM-DDThh:mm:ssZ，例如2016-02-25T12:00:00Z（为UTC时间2016年2月25日12点0分0秒）。
SignatureVersion	String	是	签名算法版本，目前版本是1.0。
SignatureNonce	String	是	唯一随机数，用于防止网络重放攻击，用户在不同请求间要使用不同的随机数值。

示例

```
https://cloudpush.aliyuncs.com/
?Format=XML
&RegionId=cn-hangzhou
&Version=2015-08-27
&AccessKeyId=testid
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&Timestamp=2016-02-25T12:00:00Z
&<接口相关参数>
...
```

公共返回参数

用户发送的每次接口调用请求，无论成功与否，系统都会返回一个唯一识别码RequestId给用户。

XML示例

```
<?xml version="1.0" encoding="UTF-8"?>
<!-结果的根结点-->
<接口名称+Response>
    <!-返回请求标签-->
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!-返回结果数据-->
</接口名称+Response>
```

JSON示例

```
{
   "RequestId": "4C467B38-3910-447D-87BC-AC049166F216"
   /* 返回结果数据 */
}
```

返回结果

调用API服务后返回数据采用统一格式，返回的HTTP状态码为2xx，代表调用成功；返回4xx或5xx的HTTP状态码代表调用失败。

调用成功返回的数据格式主要有XML和JSON两种，外部系统可以在请求时传入参数来制定返回的数据格式，默认为XML格式。

本文档中的返回示例为了便于用户查看，做了格式化处理，实际返回结果是没有进行换行、缩进等处理的。

成功结果

XML示例

```
<?xml version="1.0" encoding="UTF-8"?>
<!-结果的根结点-->
<接口名称+Response>
   <!-返回请求标签-->
   <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
   <!-返回结果数据-->
</接口名称+Response>
```

JSON示例

```
{
   "RequestId": "4C467B38-3910-447D-87BC-AC049166F216"
   /* 返回结果数据 */
}
```

错误结果

调用接口出错后，将不会返回结果数据。调用方可根据每个接口对应的错误码以及错误代码表来定位错误原因。

当调用出错时，HTTP请求返回一个4xx或5xx的HTTP状态码，返回的消息体中是具体的错误代码及错误信息。另外还包含一个全局唯一的请求ID（RequestId）和一个您该次请求访问的站点ID（HostId）。在调用方找不到错误原因时，可以联系阿里云客服，并提供该HostId和RequestId，以便我们尽快帮您解决问题。

XML示例

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <RequestId>8906582E-6722-409A-A6C4-0E7863B733A5</RequestId>
  <HostId>cloudpush.aliyuncs.com</HostId>
  <Code>InvalidAccessKeyId.NotFound</Code>
  <Message>The Access Key ID provided does not exist in our records.</Message>
</Error>
```

JSON示例

```
{
   "RequestId": "8906582E-6722-409A-A6C4-0E7863B733A5",
   "HostId": "cloudpush.aliyuncs.com",
   "Code": "InvalidAccessKeyId.NotFound",
   "Message": "The Access Key ID provided does not exist in our records."
}
```

签名机制

移动推送服务会对每个访问的请求进行身份验证，所以无论使用HTTP还是HTTPS协议提交请求，都需要在请求中包含签名（Signature）信息。服务通过使用Access Key ID和Access Key Secret进行对称加密的方法来验证请求的发送者身份。

Access Key ID和Access Key Secret由阿里云官方颁发给访问者（可以通过阿里云官方网站申请和管理），其中Access Key ID用于标识访问者的身份；Access Key Secret是用于加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密，只有阿里云和用户知道。

用户在访问时，按照下面的方法对请求进行签名处理：

1. 使用请求参数构造规范化的请求字符串（Canonicalized Query String）
   a) 按照参数名称的字典顺序对请求中所有的请求参数（包括文档中描述的"公共请求参数"和给定了的请求接口的自定义参数，但不能包括"公共请求参数"中提到Signature参数本身）进行排序。
   注意：此排序严格大小写敏感排序。
   注：当使用GET方法提交请求时，这些参数就是请求URI中的参数部分（即URI中"?"之后由"&"连接的部分）。
   b) 对每个请求参数的名称和值进行编码。名称和值要使用UTF-8字符集进行URL编码，URL编码的编码规则是：

   i. 对于字符 A-Z、a-z、0-9以及字符"-"、"_"、"."、"~"不编码；
   ii. 对于其他字符编码成"%XY"的格式，其中XY是字符对应ASCII码的16进制表示。比如英文的双引号（"）对应的编码就是%22

iii. 对于扩展的UTF-8字符，编码成"%XY%ZA..."的格式；
iv. 需要说明的是英文空格（ ）要被编码是%20，而不是加号（+）。

注：一般支持URL编码的库（比如Java中的java.net.URLEncoder）都是按照"application/x-www-form-urlencoded"的MIME类型的规则进行编码的。实现时可以直接使用这类方式进行编码，把编码后的字符串中加号（+）替换成%20、星号（*）替换成%2A、%7E替换回波浪号（~），即可得到上述规则描述的编码字符串。
c) 对编码后的参数名称和值使用英文等号（=）进行连接。
d) 再把英文等号连接得到的字符串按参数名称的字典顺序依次使用&符号连接，即得到规范化请求字符串。

2. 使用上一步构造的规范化字符串按照下面的规则构造用于计算签名的字符串：

   StringToSign= HTTPMethod + "&" + percentEncode("/") + "&" + percentEncode(CanonicalizedQueryString)

   其中HTTPMethod是提交请求用的HTTP方法，比如GET。
   percentEncode("/")是按照1.b中描述的URL编码规则对字符"/"进行编码得到的值，即"%2F"。
   percentEncode(CanonicalizedQueryString)是对第1步中构造的规范化请求字符串按1.b中描述的URL编码规则编码后得到的字符串。
3. 按照RFC2104的定义，使用上面的用于签名的字符串计算签名HMAC值。注意：计算签名时使用的Key就是用户持有的Access Key Secret并加上一个"&"字符(ASCII:38)，使用的哈希算法是SHA1。
4. 按照Base64编码规则把上面的HMAC值编码成字符串，即得到签名值（Signature）。
5. 将得到的签名值作为Signature参数添加到请求参数中，即完成对请求签名的过程。
6. 注意：得到的签名值在作为最后的请求参数值提交给服务器的时候，要和其他参数一样，按照RFC3986的规则进行URL编码）。

以GetDeviceInfos为例，签名前的请求URL为：

```
http://cloudpush.aliyuncs.com/?Format=XML&AccessKeyId=testid&Action=GetDeviceInfos&SignatureMethod=HMAC-SHA1&RegionId=cn-hangzhou&Devices=e2ba19de97604f55b165576736477b74%2C92a1da34bdfd4c9692714917ce22d53d&SignatureNonce=c4f5f0de-b3ff-4528-8a89-fa478bda8d80&SignatureVersion=1.0&Version=2015-08-27&AppKey=23267207&Timestamp=2016-03-29T03%3A59%3A24Z
```

那么StringToSign就是：

```
GET&%2F&AccessKeyId%3Dtestid%26Action%3DGetDeviceInfos%26AppKey%3D23267207%26Devices%3De2ba19de97604f55b165576736477b74%252C92a1da34bdfd4c9692714917ce22d53d%26Format%3DXML%26RegionId%3Dcn-hangzhou%26SignatureMethod%3DHMAC-SHA1%26SignatureNonce%3Dc4f5f0de-b3ff-4528-8a89-fa478bda8d80%26SignatureVersion%3D1.0%26Timestamp%3D2016-03-29T03%253A59%253A24Z%26Version%3D2015-08-27
```

假如使用的Access Key Id是"testid"，Access Key Secret是"testsecret"，用于计算HMAC的Key就是"testsecret&"，则计算得到的签名值是：

```
Q4jj5vC+NRtz294V+oIW7gfaJ6U=
```

签名后的请求URL为（注意增加了Signature参数）：

```
http://cloudpush.aliyuncs.com/?Format=XML&AccessKeyId=testid&Action=GetDeviceInfos&SignatureMethod=HMAC-SHA1&RegionId=cn-hangzhou&Devices=e2ba19de97604f55b165576736477b74%2C92a1da34bdfd4c9692714917ce22d53d&SignatureNonce=c4f5f0de-b3ff-4528-8a89-fa478bda8d80&SignatureVersion=1.0&Version=2015-08-27&AppKey=23267207&Signature=Q4jj5vC%2BNRtz294V%2BoIW7gfaJ6U%3D&Timestamp=2016-03-29T03%3A59%3A24Z
```

API列表

PushMessageToAndroid

描述

推送消息给Android设备。

注：该接口默认只发送给在线设备，要发送离线保存消息请用推送高级接口。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：PushMessageToAndroid
AppKey	Long	是	AppKey信息
Target	String	是	推送目标： - device:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - account:推

			送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - alias:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - tag:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - all:推送给全部设备
TargetValue	String	是	根据Target来设定，多个值使用逗号分隔，最多支持100个。 - Target=device，值如deviceid111,deviceid1111 - Target=account，值如account111,account222 - Target=alias，值如alias111,alias222 - Target=tag，支持单Tag和多

			Tag，格式请参考标签格式 - Target=all，值为all
Message	String	是	发送的消息内容

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述
ResponseId	String	请求返回ID（如需排查问题可告诉工作人员该ID）

错误码

对于所有接口的通用性错误，请参考错误代码表。

示例

请求示例

```
<ol><li> http : //cloudpush.aliyuncs.com/?Action=PushMessageToAndroid</li><li> & AppKey = 23267207</li><li> & Target = all &</li><li> & TargetValue = all</li><li> & Message = hello</li><li> &<公共请求参数></li></ol>
```

返回示例

XML格式

```
<ol><li> <? xml version = "1.0" encoding = "UTF-8" ?></li><li> <PushMessageToAndroidResponse></li><li> <ResponseId> 129376032 </ResponseId></li><li> </PushMessageToAndroidResponse></li></ol>
```

JSON格式

```
<ol><li> {</li><li> "ResponseId" : "129376288"</li><li> }</li></ol>
```

PushNoticeToAndroid

描述

推送通知给Android。

注：该接口默认只发送给在线设备，要发送离线保存消息请用推送高级接口。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：PushNoticeToAndroid
AppKey	Long	是	AppKey信息
Target	String	是	推送目标： - device:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - account:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - alias:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - tag:推送给设备(注：推送

			目标以TargetValue为准，该场景不区分设备类型) - all:推送给全部设备
TargetValue	String	是	根据Target来设定，多个值使用逗号分隔，最多支持100个。 - Target=device，值如deviceid111,deviceid1111 - Target=account，值如account111,account222 - Target=alias，值如alias111,alias222 - Target=tag，支持单Tag和多Tag，格式请参考标签格式 - Target=all，值为all
Title	String	是	发送的通知标题,最长20个字符，中文算1个字符
Summary	String	是	发送的通知内容
AndroidExtParameters	String	否	自定义的KV结构，供开发者扩展使用，针对Android设备

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述
ResponseId	String	请求返回ID（如需排查问题可告诉工作人员该ID）

错误码

对于所有接口的通用性错误，请参考**错误代码表**。

示例

请求示例

```
<ol><li> http : //cloudpush.aliyuncs.com/?Action=PushNoticeToAndroid</li><li> & AppKey = 23267207</li><li> & Target = all &</li><li> & TargetValue = all</li><li> & Title = hello</li><li> & Summary = hello</li><li> &<公共请求参数></li></ol>
```

返回示例

XML格式

```
<ol><li> <? xml version = '1.0' encoding = 'UTF-8' ?></li><li> <PushNoticeToAndroidResponse></li><li> <ResponseId> 129376928 </ResponseId></li><li> </PushNoticeToAndroidResponse></li></ol>
```

JSON格式

```
<ol><li> {</li><li> "ResponseId" : "129377184"</li><li> }</li></ol>
```

PushMessageToiOS

描述

推送消息给iOS。

注：该接口默认只发送给在线设备，要发送离线保存消息请用推送高级接口。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：PushMessageToiOS
AppKey	Long	是	AppKey信息
Target	String	是	推送目标： - device:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - account:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - alias:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - tag:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - all:推送给全部设备
TargetValue	String	是	根据Target来设定

			，多个值使用逗号分隔，最多支持100个。 - Target=device，值如deviceid111,deviceid1111 - Target=account，值如account111,account222 - Target=alias，值如alias111,alias222 - Target=tag，支持单Tag和多Tag，格式请参考标签格式 - Target=all，值为all
Message	String	是	发送的消息内容
Summary	String	是	发送的消息概要

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述
ResponseId	String	请求返回ID（如需排查问题可告诉工作人员该ID）

错误码

对于所有接口的通用性错误，请参考错误代码表。

示例

请求示例

 http : //cloudpush.aliyuncs.com/?Action=PushMessageToiOS & AppKey = 23267207 & Target = all & & TargetValue = all & Message = hello & Summary = hello &<公共请求参数>

返回示例

XML格式

 <? xml version = '1.0' encoding = 'UTF-8' ?> <PushMessageToiOSResponse> <ResponseId> 129377568 </ResponseId> </PushMessageToiOSResponse>

JSON格式

 { "ResponseId" : "129376288" }

PushNoticeToiOS

描述

推送通知给iOS。

注：该接口默认只发送给在线设备，要发送离线保存消息请用推送高级接口。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：PushNoticeToiOS
AppKey	Long	是	AppKey信息
Target	String	是	推送目标： - device:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型)

			- account:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - alias:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - tag:推送给设备(注：推送目标以TargetValue为准，该场景不区分设备类型) - all:推送给全部设备
TargetValue	String	是	根据Target来设定，多个值使用逗号分隔，最多支持100个。 - Target=device，值如deviceid111,deviceid1111 - Target=account，值如account111,account222 - Target=alias，值如alias111,alias222 - Target=tag，支持单

			Tag和多Tag，格式请参考标签格式 - Target=all，值为all
Env	String	是	iOS的通知是通过APNS中心来发送的，需要填写对应的环境信息，DEV表示开发环境，PRODUCT表示生产环境
Summary	String	是	发送的通知内容
Ext	String	是	用于自定义设置系统参数:iOSBadge和iOSMusic，设置方式:{"sound":"default","badge":"42"}
iOSExtParameters	String	否	自定义的KV结构，开发者扩展用，针对iOS设备

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述
ResponseId	String	请求返回ID（如需排查问题可告诉工作人员该ID）

错误码

对于所有接口的通用性错误，请参考错误代码表。

示例

请求示例

```
<ol><li> http : //cloudpush.aliyuncs.com/?Action=PushNoticeToiOS</li><li> & AppKey = 23267207</li><li> &
```

Target = all & & TargetValue = all & Env = DEV & Summary = hello & Ext =% 7B % 22aaa % 22 % 3A % 22bbb % 22 % 7D &<公共请求参数>

返回示例

XML格式

 <? xml version = '1.0' encoding = 'UTF-8' ?> <PushNoticeToiOSResponse> <ResponseId> 129377952 </ResponseId> </PushNoticeToiOSResponse>

JSON格式

 { "ResponseId" : "129377184" }

Push

描述

推送高级接口。

请求参数

- 通用参数请参考公共请求参数

基础参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：Push
AppKey	Long	是	AppKey信息

推送目标(destination)

名称	类型	是否必须	描述
Target	String	是	推送目标 - device：推送给设备(注：推送目标以TargetValue为准，该场

			景不区分设备类型) - account：推送给指定帐号(注：推送目标以TargetValue为准，该场景不区分设备类型) - alias：推送给指定别名(注：推送目标以TargetValue为准，该场景不区分设备类型) - tag：推送给指定Tag(注：推送目标以TargetValue为准，该场景不区分设备类型) - all：推送给全部设备
TargetValue	String	是	根据Target来设定，多个值使用逗号分隔，最多支持100个。 - Target=device，值如deviceid111,deviceid1111 - Target=account，值如account111,account222 - Target=alias，值如

			alias111,alias222 - Target=tag，支持单Tag和多Tag，格式请参考**标签格式** - Target=all，值为all
DeviceType	Integer	是	设备类型，取值范围为： - 0：iOS设备 - 1：Andriod设备 - 3：全部类型设备

推送配置(config)

名称	类型	是否必须	描述
Type	Integer	否	- 0：表示消息，**默认值** - 1：表示通知
Title	String	是	Android推送时通知的标题/消息的标题,最长20个字符，中文算1个字符
Summary	String	否	iOS通知的内容
Body	String	是	Android推送时通知的内容/消息的内容；iOS消息内容

Title/Summary/Body 展开说明如下：

属性\推送类型	消息 - iOS	消息 - Android	通知 - iOS	通知 - Android
Title	N/A	CPushMessage.title字段	N/A	通知标题，通知回调方法（onNotificationOpened）

Body	消息体，对应onMessageReceived回调的参数NSNotification的data字段	消息体，CPushMessage.content字段	N/A	通知内容，通知回调方法（onNotificationOpened）
Summary	N/A	N/A	通知内容	N/A

专用配置：iOS通知（不用于消息！）

名称	类型	是否必须	描述
iOSMusic	String	否	iOS通知声音
iOSBadge	String	否	iOS应用图标右上角角标
iOSExtParameters	String	否	通知的扩展属性
ApnsEnv	String	是	iOS的通知是通过APNs中心来发送的，需要填写对应的环境信息。 - DEV：表示开发环境 - PRODUCT：表示生产环境

专用配置：iOS消息（不用于通知！）

名称	类型	是否必须	描述
Remind	Boolean	是	推送时设备不在线（既与移动推送的服务端的长连接通道不通），则这条推送会做为通知，通过苹果的APNs通道送达一次。

专用配置：Android通知（不用于消息！）

名称	类型	是否必须	描述
AndroidMusic	String	否	Android通知声音
AndroidOpenType	String	是	点击通知后动作 - 1：打开应用 - 2：打开应用Activity

			- 3：打开URL
AndroidActivity	String	否	设定通知打开的activity，仅当AndroidOpenType=2有效
AndroidOpenUrl	String	否	Android收到推送后打开对应的url,仅当AndroidOpenType=3有效
AndroidExtParameters	String	否	设定通知的扩展属性

推送控制(push control)

名称	类型	是否必须	描述
PushTime	String	否	用于定时发送。不设置缺省是立即发送。时间格式按照ISO8601标准表示，并需要使用UTC时间，格式为YYYY-MM-DDThh:mm:ssZ。
StoreOffline	Boolean	是	离线消息是否保存,若保存, 在推送时候，用户即使不在线，下一次上线时会收到，与ExpireTime参数配合使用。
ExpireTime	String	否	离线消息的过期时间，过期则不会再被发送。离线消息最长保存72小时，过期时间时长不会超过发送时间加72小时。时间格式按照ISO8601标准表示，并需要使用UTC时间，格式为YYYY-MM-DDThh:mm:ssZ。

推送跟踪(trace)

名称	类型	是否必须	描述
BatchNumber	String	否	批次编号,用于批量数据统计，比如把一批推送设置为一个批次号

响应参数

名称	类型	描述
ResponseId	String	请求返回ID（如需排查问题可告诉工作人员该ID）

错误码

对于所有接口的通用性错误，请参考**错误代码表**。

示例

请求示例

```
<ol><li> http : //cloudpush.aliyuncs.com/?Action=Push</li><li> & AppKey = 23267207</li><li> & Target = device &
</li><li> & TargetValue = e2ba19de97604f55b165576736477b74 % 2C92a1da34bdfd4c9692714917ce22d53d
</li><li> & Title = hello</li><li> & Body = hello</li><li> & Type = 1</li><li> & AndroidOpenType = 1</li><li> &
DeviceType = 3</li><li> & Remind = false</li><li> & StoreOffline = false</li><li> &<公共请求参数></li></ol>
```

返回示例

XML格式

```
<ol><li> <? xml version = '1.0' encoding = 'UTF-8' ?></li><li> <PushResponse></li><li> <ResponseId> 129376928
</ResponseId></li><li> </PushResponse></li></ol>
```

JSON格式

```
<ol><li> {</li><li> "ResponseId" : "129377184"</li><li> }</li></ol>
```

Java示例代码

```
<ol><li> PushRequest pushRequest = new PushRequest ();</li><li></li><li> // 推送目标</li><li> pushRequest .
setAppKey ( appKey );</li><li> pushRequest . setTarget ( "all" ); //推送目标: device:推送给设备; account:推送给指定帐号
,tag:推送给自定义标签; all: 推送给全部</li><li> pushRequest . setTargetValue ( "all" ); //根据Target来设定，如
Target=device, 则对应的值为 设备id1,设备id2. 多个值使用逗号分隔.(帐号与设备有一次最多100个的限制)</li><li>
pushRequest . setDeviceType ( 3 ); // 设备类型deviceType 取值范围为:0~3. iOS设备: 0; Android设备: 1; 全部: 3, 这是默认值.
</li><li></li><li></li><li> // 推送配置</li><li> pushRequest . setType ( 1 ); // 0:表示消息(默认为0), 1:表示通知</li><li>
pushRequest . setTitle ( "Hello" ); // 消息的标题</li><li> pushRequest . setBody ( "PushRequest body" ); // 消息的内容
```

 pushRequest . setSummary ("PushRequest summary"); // 通知的摘要 // 推送配置: iOS pushRequest . setiOSBadge ("5"); // iOS应用图标右上角角标 pushRequest . setiOSMusic ("default"); // iOS通知声音 pushRequest . setiOSExtParameters ("{\"k1\":\"ios\",\"k2\":\"v2\"}"); //自定义的kv结构,开发者扩展用 针对iOS设备 pushRequest . setApnsEnv ("DEV"); pushRequest . setRemind (true); // 当APP不在线时候，是否通过通知提醒 // 推送配置: Android pushRequest . setAndroidOpenType ("3"); // 点击通知后动作,1:打开应用 2: 打开应用Activity 3:打开 url pushRequest . setAndroidOpenUrl ("http://www.baidu.com"); // Android收到推送后打开对应的url,仅仅当androidOpenType=3有效 pushRequest . setAndroidExtParameters ("{\"k1\":\"android\",\"k2\":\"v2\"}"); // 设定android类型设备通知的扩展属性 // 推送控制 final Date pushDate = new Date (System . currentTimeMillis () + 3600 * 1000); // 一小时后发送, 也可以设置成你指定固定时间 final String pushTime = ParameterHelper . getISO8601Time (pushDate); pushRequest . setPushTime (pushTime); // 延后推送。可选，如果不设置表示立即推送 pushRequest . setStoreOffline (true); // 离线消息是否保存,若保存, 在推送时候，用户即使不在线，下一次上线则会收到 final String expireTime = ParameterHelper . getISO8601Time (new Date (System . currentTimeMillis () + 12 * 3600 * 1000)); // 12小时后消息失效, 不会再发送 pushRequest . setExpireTime (expireTime); pushRequest . setBatchNumber ("100010"); // 批次编号,用于活动效果统计. 设置成业务可以记录的字符串 PushResponse pushResponse = client . getAcsResponse (pushRequest); System . out . printf ("RequestId: %s, ResponseId: %s, message: %s\n" , pushResponse . getRequestId (), pushResponse . getResponseId (), pushResponse . getMessage ());

 com . aliyuncs . exceptions . ClientException : Push Failed : Push Fail ! 排查步骤 1 、检查 SDK 的版本号：目前的版本是 aliyun - java - sdk - core 2.3 . 2 ； aliyun - java - sdk - push 2.1 . 0 2 、 title 是不是大于 20 个字符了？ 3 、是不是有必填参数没有填写？ 高级接口因为既可以发送 iOS 也可以发送 Android ，所以 iOS 和 Android 必填参数都需要填写，参数的完整性校验之后才会按照参数设置走 iOS 或者 Android 的发送流程。

GetDeviceInfos

描述

查询设备状态信息。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：GetDeviceInfos
AppKey	Long	是	AppKey信息
Devices	String	是	查询的设备，多个设备用逗号分隔，最多支持1000个

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述

DeviceInfos	复杂对象	包含设备信息DeviceInfo的列表

错误码

对于所有接口的通用性错误，请参考错误代码表。

示例

请求示例

```
http://cloudpush.aliyuncs.com/?Action=GetDeviceInfos
&AppKey=23267207
&Devices=e2ba19de97604f55b165576736477b74%2C92a1da34bdfd4c9692714917ce22d53d
&<公共请求参数>
```

返回示例

XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<GetDeviceInfosResponse>
    <DeviceInfos>
        <DeviceInfo>
            <DeviceId>e2ba19de97604f55b165576736477b74</DeviceId>
            <IsOnline>false</IsOnline>
        </DeviceInfo>
        <DeviceInfo>
            <DeviceId>92a1da34bdfd4c9692714917ce22d53d</DeviceId>
            <IsOnline>false</IsOnline>
        </DeviceInfo>
    </DeviceInfos>
</GetDeviceInfosResponse>
```

JSON格式

```
{
  "DeviceInfos": {
    "DeviceInfo": [
      {
        "DeviceId": "e2ba19de97604f55b165576736477b74",
        "IsOnline": false
      },
      {
        "DeviceId": "92a1da34bdfd4c9692714917ce22d53d",
        "IsOnline": true
```

```
    }
   ]
  }
}
```

BindTag

描述

绑定Tag。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：BindTag
AppKey	Long	是	AppKey信息
ClientKey	String	是	设备或account，多个key用逗号分隔，最多支持1000个
KeyType	Integer	是	ClientKey的类型，设备(1)，account(2)
TagName	String	是	绑定的Tag，多个Tag用逗号分隔，系统总共支持128个Tag，此接口一次最多能绑定10个Tag

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述
RequestId	String	全局唯一的请求ID

错误码

对于所有接口的通用性错误，请参考错误代码表。

示例

请求示例

```
http://cloudpush.aliyuncs.com/?Action=BindTag
&AppKey=23267207
&KeyType=1
&ClientKey=e2ba19de97604f55b165576736477b74%2C92a1da34bdfd4c9692714917ce22d53d
&TagName=test_tag1%2Ctest_tag2
&<公共请求参数>
```

返回示例

XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<BindTagResponse>
   <RequestId>82FD0A09-5BB8-40FB-8221-9A11FE92D620</RequestId>
</BindTagResponse>
```

JSON格式

```
{
  "RequestId": "159E4422-6624-4750-8943-DFD98D34858C"
}
```

UnbindTag

描述

解绑Tag。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：UnbindTag
AppKey	Long	是	AppKey信息
ClientKey	String	是	设备或account，多个

			key用逗号分隔，最多支持1000个
KeyType	Integer	是	ClientKey的类型，设备(1)，account(2)
TagName	String	是	绑定的Tag，多个Tag用逗号分隔，系统总共支持128个Tag，此接口一次最多能绑定10个Tag

- 其它通用参数，请参考**公共请求参数**。

响应参数

名称	类型	描述
RequestId	String	全局唯一的请求ID

错误码

对于所有接口的通用性错误，请参考**错误代码表**。

示例

请求示例

```
http://cloudpush.aliyuncs.com/?Action=UnbindTag
&AppKey=23267207
&KeyType=1
&ClientKey=e2ba19de97604f55b165576736477b74%2C92a1da34bdfd4c9692714917ce22d53d
&TagName=test_tag1%2Ctest_tag2
&<公共请求参数>
```

返回示例

XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<UnbindTagResponse>
   <RequestId>82FD0A09-5BB8-40FB-8221-9A11FE92D620</RequestId>
</UnbindTagResponse>
```

JSON格式

```
{
  "RequestId": "159E4422-6624-4750-8943-DFD98D34858C"
}
```

ListTags

描述

查询App全部的Tag列表。

请求参数

名称	类型	是否必须	描述
Action	String	是	操作接口名称，取值：ListTags
AppKey	Long	是	AppKey信息

- 其它通用参数，请参考公共请求参数。

响应参数

名称	类型	描述
RequestId	String	全局唯一的请求ID
TagInfos	复杂对象	包含Tag信息TagInfo的列表

错误码

对于所有接口的通用性错误，请参考错误代码表。

示例

请求示例

```
http://cloudpush.aliyuncs.com/?Action=ListTags
&AppKey=23267207
&<公共请求参数>
```

返回示例

XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<ListTagsResponse>
    <RequestId>9998B3CC-ED9E-4CB3-A8FB-DCC61296BFBC</RequestId>
    <TagInfos>
        <TagInfo>
            <TagName>test_tag2</TagName>
        </TagInfo>
        <TagInfo>
            <TagName>test_tag1</TagName>
        </TagInfo>
    </TagInfos>
</ListTagsResponse>
```

JSON格式

```
{
  "RequestId": "6EEF262B-EA7D-41DC-89B9-20F3D1E28194",
  "TagInfos": {
    "TagInfo": [
      {
        "TagName": "test_tag2"
      },
      {
        "TagName": "test_tag1"
      }
    ]
  }
}
```

QueryTags

描述

查询某个设备的Tag列表。

请求参数

名称	类型	是否必须	描述

Action	String	是	操作接口名称，取值：QueryTags
AppKey	Long	是	AppKey信息
ClientKey	String	是	设备或account，**每次只能查询1个clientKey**
KeyType	Integer	是	ClientKey的类型，设备(1)，account(2)

- 其它通用参数，请参考**公共请求参数**。

响应参数

名称	类型	描述
RequestId	String	全局唯一的请求ID
TagInfos	复杂对象	包含Tag信息TagInfo的列表

错误码

对于所有接口的通用性错误，请参考**错误代码表**。

示例

请求示例

```
http://cloudpush.aliyuncs.com/?Action=QueryTags
&AppKey=23267207
&KeyType=1
&ClientKey=e2ba19de97604f55b165576736477b74%2C92a1da34bdfd4c9692714917ce22d53d
&<公共请求参数>
```

返回示例

XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<QueryTagsResponse>
  <RequestId>1197FEB8-3644-4355-A96D-E332F45470EA</RequestId>
  <TagInfos>
    <TagInfo>
      <TagName>test_tag2</TagName>
    </TagInfo>
```

```
      <TagInfo>
        <TagName>test_tag1</TagName>
      </TagInfo>
    </TagInfos>
</QueryTagsResponse>
```

JSON格式

```
{
  "RequestId": "D68AE5C6-8AAF-46C9-B627-3FDACD1A4168",
  "TagInfos": {
    "TagInfo": [
      {
        "TagName": "test_tag2"
      },
      {
        "TagName": "test_tag1"
      }
    ]
  }
}
```

SDK手册

Java

获得Access Key

登录阿里云官网控制台获取：https://ak-console.aliyun.com/#/accesskey

获得SDK

使用Maven依赖：

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-push</artifactId>
    <version>2.1.0</version>
</dependency>
```

示例代码

```
IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou", accessKeyId, accessKeySecret);
DefaultAcsClient client = new DefaultAcsClient(profile);
PushNoticeToAndroidRequest androidRequest  = new PushNoticeToAndroidRequest();
androidRequest.setAppKey(appKey);
androidRequest.setTarget("all");
androidRequest.setTargetValue("all");
androidRequest.setTitle("Hello,push");
androidRequest.setSummary("PushNoticeToAndroid from api");
androidRequest.setAndroidExtParameters("{\"key\":\"value123\"}");
PushNoticeToAndroidResponse pushNoticeToAndroidResponse = client.getAcsResponse(androidRequest);
System.out.println(pushNoticeToAndroidResponse.getResponseId());
```

附录

错误代码表

客户端错误

错误代码	描述	HTTP 状态码	语义
MissingParameter	The input parameter "<parameter name>" that is mandatory for processing this request is not supplied	400	缺少参数
InvalidParameter	The specified value of parameter "<parameter name>" is not valid.	400	参数取值无效
UnsupportedOperation	The specified action is not supported.	400	无效的接口
NoSuchVersion	The specified version does not exist.	400	无效的版本
Throttling	Request was denied due to request throttling.	400	操作被流量控制系统拒绝

InvalidAccessKeyId.NotFound	The Access Key ID provided does not exist in our records.	400	无效的Access Key
Forbidden	User not authorized to operate on the specified resource.	403	操作被禁止
Forbidden.RiskControl	This operation is forbidden by Aliyun Risk Control system.	403	操作被风险控制系统禁止
SignatureDoesNotMatch	The signature we calculated does not match the one you provided. Please refer to the API reference about authentication for details.	403	无效的签名
Forbidden.UserVerification	Your user account is not verified by Aliyun.	403	无实名验证

服务器端错误

错误代码	描述	HTTP 状态码	语义
InternalError	The request processing has failed due to some unknown error, exception or failure.	500	服务器无法完成对请求的处理
ServiceUnavailable	The request has failed due to a temporary failure of the server.	503	服务器当前无法处理请求

标签格式

移动推送支持**单Tag**和**多Tag**推送：

- 单Tag推送时，TargetValue的值是普通字符串，直接存放该Tag的名字即可。
- 多Tag推送时，TargetValue需要使用标签表达式，如下所述。

标签表达式

当推送目标为标签（Target=tag）时，可以选择在标签字段（TargetValue）中使用标签表达式，以实现自定

义的多标签条件推送。

语法

标签表达式通过使用条件操作符和标签的嵌套与组合，来表达多标签之间的复杂条件关系，其描述基于JSON格式。

操作符和标签类型对象在JSON结构中使用了不同的关键字进行标识，如下表所示：

关键字	类型	含义
and	操作符	表达"且"关系，可作用于一个或多个子表达式或标签
or	操作符	表达"或"关系，可作用于一个或多个子表达式或标签
not	操作符	表达"非"关系，只能作用于一个标签
tag	标签	具体的标签节点，存储业务标签名称

示例

```
{
  "and": [
    {
      "tag": "男性"
    },
    {
      "not": {
        "tag": "90后"
      }
    },
    {
      "or": [
        {
          "not": {
            "tag": "国外"
          }
        },
        {
          "tag": "活跃"
        }
      ]
    }
  ]
}
```

上述表达式将筛选出标签符合"**男性、非90后、活跃或非国外**"的用户。

限制

- and和or操作符的最大嵌套层数：2
- and和or操作符的最大操作数：5