信鸽服务端 PHP SDK V1.1.4 用户手册

1 整体介绍.....2
2 快捷方式.....2
2.1 Android 平台推送消息给单个设备.....2
2.2 Android 平台推送消息给单个账号.....2
2.3 Android 平台推送消息给所有设备.....3
2.4 Android 平台推送消息给标签选中设备.....3
2.5 IOS 平台推送消息给单个设备.....3
2.6 IOS 平台推送消息给单个账号.....3
2.7 IOS 平台推送消息给所有设备.....4
2.8 IOS 平台推送消息给标签选中设备.....4
3 高级接口.....4
3.1 数据结构介绍.....4
3.1.1 TimeInterval.....4
3.1.2 ClickAction.....5
3.1.3 Style.....6
3.1.4 Message.....7
3.1.5 MessageIOS.....8
3.1.6 TagTokenPair.....9
3.1.7 XingeApp.....9
3.2 XingeApp 高级接口.....10
3.2.1 PushSingleDevice 推送消息给单个设备.....10
3.2.2 PushSingleAccount 推送消息给单个账号.....10
3.2.3 PushAccountList 推送消息给多个账号.....11
3.2.4 PushAllDevices 推送消息给单个 app 的所有设备.....11
3.2.5 PushTags 推送消息给 tags 指定的设备.....12
3.2.6 QueryPushStatus 查询群发消息发送状态.....13
3.2.7 QueryDeviceCount 查询应用覆盖的设备数.....13
3.2.8 QueryTags 查询应用的 tags.....14
3.2.9 CancelTimingPush 取消尚未推送的定时消息.....14
3.2.10 BatchSetTag 批量为 token 设置标签.....14
3.2.11 BatchDelTag 批量为 token 删除标签.....15
3.2.12 QueryTokenTags 查询 token 的 tags.....15
3.2.13 QueryTagTokenNum 查询 tag 下 token 的数目.....16
4 附录.....17
4.1 通用返回码描述.....17

1 整体介绍

本 SDK 提供信鸽服务端接口的 PHP 封装，与信鸽后台通信。使用时包含 XingeApp.php 即可。

2 快捷方式

本章节描述如何使用**一行代码**完成推送操作。具体代码可参考 simple_demo.php。

基本概念：

名称	类型	描述
accessId	uint	推送目标应用的 ID
secretKey	string	目标应用的密钥
title	string	消息标题，**Android 专属**
content	string	消息内容
token	string	接收消息的设备 Token
account	string	接收消息的账号
tag	string	接收消息的设备标签
environment	int	可选值为 XingeApp::IOSENV_PROD 或者 XingeApp::IOSENV_DEV，**IOS 专属**

2.1 Android 平台推送消息给单个设备

```
function PushTokenAndroid($accessId,$secretKey,$title,$content,$token)
```

示例：

```
XingeApp::PushTokenAndroid(000, "myKey", "标题", "大家好!", "3dc4gcd98sdc");
```

返回值：

```
{"ret_code":0}                                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.2 Android 平台推送消息给单个账号

```
function PushAccountAndroid($accessId,$secretKey,$title,$content,$account)
```

示例：

```
XingeApp::PushAccountAndroid(000, "myKey", "标题", "大家好!", "nickName");
```

返回值：

```
{"ret_code":0}                                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.3 Android 平台推送消息给所有设备

```
function PushAllAndroid($accessId,$secretKey,$title,$content)
```

示例：

```
XingeApp::PushAllAndroid(000, "myKey", "标题", "大家好!");
```

返回值：

```
//返回 push_id，即推送的任务 ID
{"ret_code":0, "result":{"push_id":11121}}                    //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.4 Android 平台推送消息给标签选中设备

```
function PushTagAndroid($accessId,$secretKey,$title,$content,$tag)
```

示例：

```
XingeApp::PushTagAndroid(000, "myKey", "标题", "大家好!", "beijing");
```

返回值：

```
//返回 push_id，即推送的任务 ID
{"ret_code":0, "result":{"push_id":11121}}                    //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.5 IOS 平台推送消息给单个设备

```
function PushTokenIos($accessId,$secretKey,$content,$token,$environment)
```

示例：

```
XingeApp::PushTokenIos(000, "myKey", "你好!", "3dc4gcd98sdc", XingeApp::IOSENV_PROD);
```

返回值：

```
{"ret_code":0}                                                //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.6 IOS 平台推送消息给单个账号

```
function PushAccountIos($accessId,$secretKey,$content,$account,$environment)
```

示例：

```
XingeApp::PushAccountIos(000, "myKey", "你好", "nickName", XingeApp::IOSENV_PROD);
```

返回值：

```
{"ret_code":0}                                                //成功
```

```
{"ret_code":-1, "err_msg":"error description"}                    //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.7 IOS 平台推送消息给所有设备

```
function PushAllIos($accessId,$secretKey,$content,$environment)
```

示例：

```
XingeApp::PushAllIos(000, "myKey", "大家好!", XingeApp::IOSENV_PROD);
```

返回值：

```
//返回 push_id，即推送的任务 ID
{"ret_code":0, "result":{"push_id":11121}}                        //成功
{"ret_code":-1, "err_msg":"error description"}                    //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

2.8 IOS 平台推送消息给标签选中设备

```
function PushTagIos($accessId,$secretKey,$content,$tag,$environment)
```

示例：

```
XingeApp::PushTagIos(000, "myKey", "大家好!", "beijing", XingeApp::IOSENV_PROD);
```

返回值：

```
//返回 push_id，即推送的任务 ID
{"ret_code":0, "result":{"push_id":11121}}                        //成功
{"ret_code":-1, "err_msg":"error description"}                    //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3 高级接口

本章节描述如何使用带有全部参数的推送接口。具体代码可参考 demo.php。

3.1 数据结构介绍

3.1.1 TimeInterval

表示一个允许推送的时间闭区间，从 startHour : startMin 到 endHour : endMin

字段名	类型	说明
startHour	int	起始小时，取值范围 [0,23]
startMin	int	起始分钟，取值范围 [0,59]
endHour	int	截止小时，取值范围 [0,23]
endMin	int	截止分钟，取值范围 [0,59]

Example:

```
$acceptTime = new TimeInterval(0, 0, 23, 59);
```

3.1.2 ClickAction

通知消息被点击时触发的事件

字段名	类型	说明
actionType	int	TYPE_ACTIVITY 打开 activity 或 app 本身 TYPE_URL 打开指定的 url TYPE_INTENT 打开指定 Intent TYPE_PACKAGE 拉起指定的应用 默认为 TYPE_ACTIVITY 使用 setActionType($value)设置
url	string	要打开的 url， actionType 为 TYPE_URL 时必填。 使用 setUrl($value)设置
confirmOnUrl	int	打开 url 时是否需要用户确认，1 需要，0 不需要，默认为 0。 actionType 为 TYPE_URL 时生效 使用 setComfirmOnUrl($value)设置
activity	string	打开指定的 activity，不填则运行 app。 actionType 为 TYPE_ACTIVITY 时生效 使用 setActivity($value)设置
atyAttrIntentFlag	int	activity 属性，只针对 action_type=TYPE_ACTIVITY 的情况。 创建通知时，intent 的属性，如： intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED) 使用 setAtyAttrIntentFlag($value)设置
atyAttrPendingIntentFlag	int	activity 属性，只针对 action_type=TYPE_ACTIVITY 的情况。 PendingIntent 的属性，如： PendingIntent.FLAG_UPDATE_CURRENT 使用 setAtyAttrPendingIntentFlag($value)设置
intent	string	打开指定 intent。 actionTYpe 为 TYPE_INTENT 时生效。 请在 Android 上使用 intent.toUri(Intent.URI_INTENT_SCHEME)方法来得到序列化后的 intent 字符串填在此处，自定义 intent 参数也会包含在其中 使用 setIntent($value)设置
packageDownloadUrl	string	拉起应用的下载链接（若客户端没有找到此应用会自动去下载） actionTYpe 为 TYPE_PACKAGE 时生效。 使用 setPackageDownloadUrl($value)设置
confirmOnPackageDownloadUrl	int	打开 packageDownloadUrl 时是否需要用户确认，1 需要，0 不需要。 actionTYpe 为 TYPE_PACKAGE 时生效。 使用 setConfirmOnPackageDownloadUrl($value)设置

packageName	string	app 应用拉起别的应用的包名 actionTYpe 为 TYPE_PACKAGE 时生效。 使用 setPackageName($value)设置

Example:

```
$action = new ClickAction();
$action->setActionType(ClickAction::TYPE_URL);
$action->setUrl("http://xg.qq.com");
```

3.1.3 Style

定义通知消息如何展现

字段名	类型	说明
builderId	int	本地通知样式，必填。含义参见终端 SDK 文档 构造函数中指定
ring	int	是否响铃，0 否，1 是。选填，默认 0 构造函数中指定
vibrate	int	是否振动，0 否，1 是。选填，默认 0 构造函数中指定
clearable	int	通知栏是否可清除，0 否，1 是。选填，默认 1 构造函数中指定
nId	int	若大于 0，则会覆盖先前弹出的相同 id 通知；若为 0，展示本条通知且不影响其他通知；若为-1，将清除先前弹出的所有通知，仅展示本条通知。选填，默认为 0 构造函数中指定
lights	int	是否呼吸灯，0 否，1 是，选填，默认 1 构造函数中指定
iconType	int	指定通知栏图标是使用应用内图标还是使用自己上传的图标。 0 是应用内图标，1 是上传图标，选填。默认 0 构造函数中指定
iconRes	string	应用内图标文件名（xg.png）或者下载图标的 url 地址，选填 使用 setIconRes($value)设置
ringRaw	string	指定应用内的声音（ring.mp3），选填 使用 setRingRaw($value)设置
styleId	int	Web 端设置是否覆盖编号的通知样式，0 否，1 是，选填。默认 1 构造函数中指定
smallIcon	string	"xg"指定状态栏的小图片(xg.png),选填 使用 setSmallIcon($value)设置

Example:

```
#依次为$builderId[,$ring][,$vibrate][,$clearable][,$nId][,$lights][,$iconType][,$styleId]
$style = new Style(0,0,0,1,0,1,0,1);
$style->setIconRes("xg.png");
```

3.1.4 Message

定义推送消息(Android 平台)

字段名	类型	说明
title	string	标题，标题+内容不得超过 800 英文字符或 400 非英文字符 使用 `setTitle($value)`设置
content	string	内容，标题+内容不得超过 800 英文字符或 400 非英文字符 使用 `setContent($value)`设置
expireTime	int	消息离线存储多久，单位为秒，最长存储时间 3 天。选填，默认为 3 天，不允许为 0 使用 `setExpireTime($value)`设置
sendTime	string	消息定时推送的时间，格式为 year-mon-day hour:min:sec，若小于服务器当前时间则立即推送。选填，默认为空字符串，代表立即推送 使用 `setSendTime($value)`设置
acceptTime	array	元素为 TimeInterval 类型，表示允许推送的时间段，选填 使用 `addAcceptTime($value)`来添加元素
type	int	消息类型。 `TYPE_NOTIFICATION`:通知； `TYPE_MESSAGE`:透传消息。必填 注意：`TYPE_MESSAGE` 类型消息 默认在终端是不展示的 使用 `setType($value)`设置
multiPkg	int	0 表示按注册时提供的包名分发消息；1 表示按 access id 分发消息，所有以该 access id 成功注册推送的 app 均可收到消息。选填，默认为 0 使用 `setMultiPkg($value)`设置
style	Style	通知样式，透传消息可不填 使用 `setStyle($value)`设置
action	ClickAction	通知被点击的动作，默认为打开 app。透传消息可不填 使用 `setAction($value)`设置
custom	array	选填。用户自定义的 key-value，key 和 value 都必须为 string 注意：用于用户透传消息，此项内容将记入消息字符数。 使用 `setCustom($value)`设置
loopInterval	int	循环任务的执行间隔，选填。以天为单位，取值 1 到 14。第一条循环任务与最后一条循环任务间隔不超过 14 天。 此参数仅对 PushAllDevices 和 PushTags 接口有效。 使用 `setLoopInterval($value)`设置
loopTimes	int	循环任务的执行次数，选填。取值 1 到 15。第一条循环任务与最后一条循环任务间隔不超过 14 天。 此参数仅对 PushAllDevices 和 PushTags 接口有效。 使用 `setLoopTimes($value)`设置

Example:

```php
$mess = new Message();
$mess->setType(Message::TYPE_NOTIFICATION);
$mess->setTitle("title");
$mess->setContent("中午");
$mess->setExpireTime(86400);

#含义：样式编号0，响铃，震动，不可从通知栏清除，不影响先前通知
$mess->setStyle(new Style(0,1,1,0,0));

$action = new ClickAction();
$action->setActionType(ClickAction::TYPE_URL);
$action->setUrl("http://xg.qq.com");
$action->setComfirmOnUrl(1);
$mess->setAction($action);

$custom = array('key1'=>'value1', 'key2'=>'value2');
$mess->setCustom($custom);

$acceptTime1 = new TimeInterval(12, 0, 13, 59);
$acceptTime2 = new TimeInterval(19, 0, 20, 59);
$mess->addAcceptTime($acceptTime1);
$mess->addAcceptTime($acceptTime2);
```

3.1.5 MessageIOS

定义 iOS 平台推送消息

字段名	类型	说明
expireTime	int	消息离线存储多久，单位为秒，最长存储时间 3 天。选填，默认为 3 天，不允许为 0 使用 `setExpireTime($value)`设置
sendTime	string	消息定时推送时间，格式为 year-mon-day hour:min:sec，若小于服务器当前时间则立即推送。选填，默认为空字符串，代表立即推送 使用 `setSendTime($value)`设置
acceptTime	array	元素为 TimeInterval，表示允许推送的时间段，选填 使用 `addAcceptTime($value)`添加元素
alert	string/array	定义详见 APNS payload 中的 alert 字段 使用 `setAlert($value)`设置
badge	int	设置角标数值。定义详见 APNS payload 中的 badge 字段 使用 `setBadge($value)`设置
sound	string	设置通知声音。定义详见 APNS payload 中的 sound 字段 使用 `setSound($value)`设置
custom	array	选填。用户自定义的 key-value。注意每一个 key 和 value 的字符都计入消息字符数

		使用 `setCustom($value)`设置
category	string	iOS 8新增。定义详见 APNS payload 中的 category 字段 使用 `setCategory($value)`设置
loopInterval	int	循环任务的执行间隔，选填。以天为单位，取值 1 到 14。第一条循环任务与最后一条循环任务间隔不超过 14 天。 此参数仅对 PushAllDevices 和 PushTags 接口有效。 使用 `setLoopInterval($value)`设置
loopTimes	int	循环任务的执行次数，选填。取值 1 到 15。第一条循环任务与最后一条循环任务间隔不超过 14 天。 此参数仅对 PushAllDevices 和 PushTags 接口有效。 使用 `setLoopTimes($value)`设置

Example:

```
$mess = new MessageIOS();
$mess->setExpireTime(86400);
$mess->setAlert("ios test");
$mess->setBadge(1);
$mess->setSound("beep.wav");
$custom = array('key1'=>'value1', 'key2'=>'value2');
$mess->setCustom($custom);
$acceptTime = new TimeInterval(0, 0, 23, 59);
$mess->addAcceptTime($acceptTime);
```

3.1.6 TagTokenPair

定义一个标签-token 对

字段名	类型	说明
tag	string	标签，长度最多 50 字节，不可包含空格
token	string	设备 token。注意长度，Android 是 40 或 64 字节，iOS 是 64 字节

Example:

```
$pair = new TagTokenPair("tag1", "token00000000000000000000000000000001");
```

3.1.7 XingeApp

该类提供与信鸽后台交互的接口。构造函数有两个参数，均为必选。

参数名	类型	必需	默认值	参数描述
accessId	int	是	无	推送目标应用 id
secretKey	string	是	无	推送密钥

Example:

```
$push = new XingeApp(000, "myKey");
```

3.2 XingeApp 高级接口

3.2.1 PushSingleDevice 推送消息给单个设备

```
function PushSingleDevice($deviceToken,$message,$environment=0)
```

参数说明：

参数名	类型	必需	默认值	参数描述
deviceToken	string	是	无	推送目标设备 token
message	Message	是	无	消息体，IOS 推送使用 MessageIOS 类型
environment	int	**iOS 专用**	0	向 iOS 设备推送时必填，IOSENV_PROD 表示推送生产环境；IOSENV_DEV 表示推送开发环境。 推送 Android 平台不填或填 0

Example：

```
$push = new XingeApp(000, 'myKey');
$mess = new Message(); //$mess = new MessageIOS();
//完善 Message 消息
...
$ret = $push->PushSingleDevice('token', $mess);
```

Return value：

```
{"ret_code":0}                                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.2 PushSingleAccount 推送消息给单个账号

```
function PushSingleAccount($deviceType,$account,$message,$environment=0)
```

备注：设备的账号由终端 SDK 在调用推送注册接口时设置，详情参考终端 SDK 文档。

参数说明：

参数名	类型	必需	默认值	参数描述
deviceType	int	否	0	请填 0
account	string	是	无	推送目标账号
message	Message	是	无	消息体，IOS 推送使用 MessageIOS 类型
environment	int	**iOS 专用**	0	向 iOS 设备推送时必填，IOSENV_PROD 表示推送生产环境；IOSENV_DEV 表示推送开发环境。 推送 Android 平台不填或填 0

Example：

```
$push = new XingeApp(000, 'myKey');
$mess = new Message(); //$mess = new MessageIOS();
//完善 Message 消息
...
$ret = $push->PushSingleAccount(0, 'nickName', $mess);
```

Return value:

```
{"ret_code":0}                                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.3 PushAccountList 推送消息给多个账号

```
function PushAccountList($deviceType,$accountList,$message,$environment=0)
```

备注：设备的账号由终端 SDK 在调用推送注册接口时设置，详情参考终端 SDK 文档。

参数说明：

参数名	类型	必需	默认值	参数描述
deviceType	int	否	0	请填 0
accountList	Array	是	无	推送目标账号，账号数量有限制，如 100 个
message	Message	是	无	消息体，IOS 推送使用 MessageIOS 类型 批量发送 message 不使用 sendTime
environment	int	**iOS 专用**	0	向 iOS 设备推送时必填，IOSENV_PROD 表示推送生产环境；IOSENV_DEV 表示推送开发环境。 推送 Android 平台不填或填 0

Example:

```
$push = new XingeApp(000, 'myKey');
$mess = new Message(); //$mess = new MessageIOS();
//完善 Message 消息
...
$ret = $push->PushAccountList(0, array('joel','nickName'), $mess);
```

Return value:

```
//result 为一个整数数组，按顺序记录每一个 account 的推送结果
{"ret_code":0, "result":[0,2]}                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.4 PushAllDevices 推送消息给单个 app 的所有设备

```
function PushAllDevices($deviceType,$message,$environment=0)
```

参数说明：

参数名	类型	必需	默认值	参数描述
deviceType	int	否	0	请填 0
message	Message	是	无	消息体，IOS 推送使用 MessageIOS 类型
environment	int	**iOS 专用**	0	向 iOS 设备推送时必填，IOSENV_PROD 表示推送生产环境；IOSENV_DEV 表示推送开发环境。 推送 Android 平台不填或填 0

Example：

```
$push = new XingeApp(000, 'myKey');
$mess = new Message(); //$mess = new MessageIOS();
//完善 Message 消息
...
$ret = $push->PushAllDevices(0, $mess);
```

Return value：

```
//返回 push_id，即推送的任务 ID
{"ret_code":0, "result":{"push_id":11121}}                    //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.5 PushTags 推送消息给 tags 指定的设备

```
function PushTags($deviceType,$tagList,$tagsOp,$message,$environment=0)
```

参数说明：

参数名	类型	必需	默认值	参数描述
deviceType	int	否	0	请填 0
tagList	array	是	无	指定推送目标的 tag 列表，每个 tag 是一个 string
tagsOp	string	是	无	多个 tag 的运算关系，取值为 AND 或 OR
message	Message	是	无	消息体，IOS 推送使用 MessageIOS 类型
environment	int	**iOS 专用**	0	向 iOS 设备推送时必填，IOSENV_PROD 表示推送生产环境；IOSENV_DEV 表示推送开发环境。 推送 Android 平台不填或填 0

Example：

```
$push = new XingeApp(000, 'myKey');
$mess = new Message(); //$mess = new MessageIOS();
//完善 Message 消息
...
$ret = $push->PushTags(0, array('beijing', 'man'), 'AND', $mess);
```

Return value:

```
//返回 push_id，即推送的任务 ID
{"ret_code":0, "result":{"push_id":11121}}                    //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.6 QueryPushStatus 查询群发消息发送状态

```
function QueryPushStatus($pushIdList)
```

参数说明：

参数名	类型	必需	默认值	参数描述
pushIdList	array	是	无	推送任务 id 列表，每个 id 为一个 string

Example:

```
$push = new XingeApp(000, 'myKey');
$ret = $push->QueryPushStatus(array('11121'));
```

Return value:

```
//返回由
//{
//   "push_id":11121,    //任务 ID
//   "status":1,         //状态 0->待推送; 1、6->推送中; 2->推送完成; 3->推送失败
//   "start_time":"2015-02-02 12:12:12",
//}
//对象组成的数组
{"ret_code":0, "result": {"list": [{"push_id":11121, "status":1, "start_time":"2015-02-02
12:12:12"}] } }                                               //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.7 QueryDeviceCount 查询应用覆盖的设备数

```
function QueryDeviceCount()
```

Example:

```
$push = new XingeApp(000, 'myKey');
$ret = $push->QueryDeviceCount();
```

Return value:

```
{"ret_code":0, "result":{"device_num":10000000}}              //成功
{"ret_code":-1, "err_msg":"error description"}                //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.8 QueryTags 查询应用的 tags

```
function QueryTags($start=0,$limit=100)
```

参数说明：

参数名	类型	必需	默认值	参数描述
start	int	否	0	开始位置
limit	int	否	100	限制结果数量

Example：

```
$push = new XingeApp(000, 'myKey');
$ret = $push->QueryTags(0, 100);
```

Return value：

```
//其中 total 为标签总数，tags 为要查询区间的所有 tag
{"ret_code":0, "result":{"total":3, "tags":["tag1","tag2","tag3"]}}  //成功
{"ret_code":-1, "err_msg":"error description"}                       //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.9 CancelTimingPush 取消尚未推送的定时消息

```
function CancelTimingPush($pushId)
```

参数说明：

参数名	类型	必需	默认值	参数描述
pushId	string	是	无	推送任务 id

Example：

```
$push = new XingeApp(000, 'myKey');
$ret = $push->CancelTimingPush('1234');
```

Return value：

```
{"ret_code":0}                                                       //成功
{"ret_code":-1, "err_msg":"error description"}                       //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.10 BatchSetTag 批量为 token 设置标签

```
function BatchSetTag($tagTokenPairs)
```

参数说明：

参数名	类型	必需	默认值	参数描述
tagTokenPairs	array	是	无	元素必须是 TagTokenPair。后台将对每个 pair 里的 token 设置相应

				的标签，每次调用最多允许输入 20 个 pair。

Example:

```
$push = new XingeApp(000, 'myKey');
$pairs = array();
array_push($pairs,new TagTokenPair("tag1","token0000000000000000000000000001"));
array_push($pairs,new TagTokenPair("tag2","token0000000000000000000000000002"));
$ret = $push->BatchSetTag($pairs);
```

Return value:

```
{"ret_code":0}                                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.11 BatchDelTag 批量为 token 删除标签

```
function BatchDelTag($tagTokenPairs)
```

参数说明：

参数名	类型	必需	默认值	参数描述
tagTokenPairs	array	是	无	元素必须是 TagTokenPair。后台将对每个 pair 里的 token 删除相应的标签，每次调用最多允许输入 20 个 pair。

Example:

```
$push = new XingeApp(000, 'myKey');
$pairs = array();
array_push($pairs,new TagTokenPair("tag1","token0000000000000000000000000001"));
array_push($pairs,new TagTokenPair("tag2","token0000000000000000000000000002"));
$ret = $push->BatchDelTag($pairs);
```

Return value:

```
{"ret_code":0}                                          //成功
{"ret_code":-1, "err_msg":"error description"}          //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.12 QueryTokenTags 查询 token 的 tags

```
function QueryTokenTags($deviceToken)
```

参数说明：

参数名	类型	必需	默认值	参数描述
deviceToken	string	是		

Example:

```
$push = new XingeApp(000, 'myKey');
$ret = $push->QueryTokenTags("token0000000000000000000000000002");
```

Return value:

```
{"ret_code":0, "result":{"tags":["tag1","tag2","tag3"]}}          //成功
{"ret_code":-1, "err_msg":"error description"}                    //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

3.2.13 QueryTagTokenNum 查询 tag 下 token 的数目

```
function QueryTagTokenNum($tag)
```

参数说明：

参数名	类型	必需	默认值	参数描述
tag	string	是		

Example:

```
$push = new XingeApp(000, 'myKey');
$ret = $push->QueryTagTokenNum("beijing");
```

Return value:

```
{"ret_code":0, "result":{"device_num":100000}}                    //成功
{"ret_code":-1, "err_msg":"error description"}                    //失败
```

注：ret_code 为 0 表示成功，其他为失败，具体请查看附录。

4 附录

4.1 通用返回码描述

`ret_code` 含义如下

值	含义
0	调用成功
-1	参数错误
-2	请求时间戳不在有效期内
-3	sign 校验无效，检查 access id 和 secret key（注意不是 access key）
2	参数错误，请对照文档检查请求参数
20	鉴权错误
40	推送的 token 没有在信鸽中注册，或者推送的帐号没有绑定 token
48	推送的账号没有在信鸽中注册
73	消息字符数超限
76	请求过于频繁，请稍后再试
100	APNS 证书错误。请重新提交正确的证书
其他	内部错误