百富移动推送

API 手册

V1.0

百富计算机计算技术（深圳）有限公司

版本记录

日期	版本	修订内容	作者
	V1.0	初始版本	Maly

目录

第一章 产品开通说明

百富移动推送产品帮助 App 快速集成移动推送的功能，在实现高效、精确、实时的移动推送的同时，极大地降低了开发成本。让开发者最有效地与用户保持连接，从而提高用户活跃度、提高应用的留存率。

产品开通

通过 BD 进行产品开通，开通以后将获取到下参数：

参数	说明
url	推送 API 地址
userId	用户 ID
accessKey	用户 API 密钥

创建应用

创建应用需要填如下参数：

- **应用类型**：Android 应用、IoS 应用、IoT 设备应用；
- **应用名称**：富收银
- **应用包名**：Android 应用包名、IoS bundle 、IoT 设备应用不用填

应用创建成功将生成下参数：

参数	说明

appId	应用 ID
appSecret	应用密钥

应用发布

应用创建成功后就可以直接使用了

第二章 后台 API 入门

基本 URL

可通过以下链接地址访问平台服务：

```
${URL}/api/v{apiVersion}/
```

当前版本为 1

身份验证

所有 API 调用都通过“**HTTP 基本身份验证方案**” 进行验证（**协议规范：**https://tools.ietf.org/html/rfc2617），相关参产品开通时由 BD 提供：

参数	参数描述

userId	用户 ID
accessKey	用户 API 密钥

客户在创建 API 请求时需要按照以下说明**创建授权标头（Authorization）**：

- 将用户 ID 和 API 密钥组合到一个字符串中，用分号分隔各个值。例如， 如果用户 ID 是 *starterkit*， 用户密钥是 *d703f52b-1200-4318-ae0d-0f6092b2e6ab*， 则串联的字符串将是：starterkit:d703f52b-1200-4318-ae0d-0f6092b2e6ab

- 使用 Base64(即 RFC2045-MIME)对串联的字符串进行编码

  c3RhcnRlcmtpdDpkNzAzZjUyYi0xMjAwLTQzMTgtYWUwZC0wZjYwOTJiMmU2YWI=

- 将授权标头值设置为 Basic， 后跟第 2 步中的编码字符串， 确保 Basic 和编码字符串之间有一个空格：

  Basic c3RhcnRlcmtpdDpkNzAzZjUyYi0xMjAwLTQzMTgtYWUwZC0wZjYwOTJiMmU2YWI=

请求响应

当您发出 REST API 请求时，平台将会返回 JSON 格式的响应。当 HTTP 响应码**非 200 OK** 时，JSON 响应报文如下：

```
{
    "errorCode": "错误码",
```

"**errorMessage**": "错误描述"

}

响应码 200 OK 时即请求成功

API 列表

AppStatus	查询应用状态
Push	向设备推送消息
DeviceStatus	查询设备状态

第三章 后台 API 接口

查询应用当前状态

资源 URL

GET ${URL}/api/v1/${appId}/status

请求参数

响应参数

返回值	描述

appId	
clientNums	在线客户端数量

错误描述

错误代码	HTTP 应答码	描述
40100001	401	API 凭证无效
40400001	404	应用不存在
50000001	500	未知服务器错误

推送消息到设备列表

资源 URL

POST ${URL}/api/v1/${appId}/push/${ clientId}

请求参数

参数	类型	是否必须	描述
title	String	否	消息标题
content	String	是	消息报文

响应参数

返回值	类型	是否必反	描述
messageId	String	是	唯一标识一笔消息

错误描述

错误代码	HTTP 应答码	描述
40100001	401	API 凭证无效
40400001	404	应用不存在
40400002	404	客户端不存在
50000001	500	未知服务器错误

查询设备在线状态

资源 URL

GET ${URL}/api/v1/${appId}/connects/${ clientId }

请求参数

响应参数

返回值	类型	是否必反	描述

clientId	String	是	
onLine	Boolean	是	设备是否在线，取值说明如下： true：在线 false：不在线
connectedAt	String	否	"2019-04-29 11:05:01",
ipaddress	String	否	127.0.0.1

错误描述

错误代码	HTTP 应答码	描述
40100001	401	API 凭证无效
40000001	400	应用不存在
50000001	500	未知服务器错误

第四章 终端 SDK 说明

一、Maven 库快速集成(本地同步)

将 SDK 压缩包在本地 maven 目录（your\path\to\.m2\repository）下解压

在 Project 根目录下 build.gradle 文件中配置 maven 库 URL 和 maven 本地库 mavenLocal()：

```
allprojects {
    repositories {
        google()
        jcenter()
        maven { url "https://jitpack.io" }
        mavenLocal()
    }}
```

在对应的 module 下的 build.gradle 文件中添加对应依赖

```
    dependencies {
    ......
    implementation 'cn.paxpay.push:pax-push:1.0.0'
}
```

二、AndroidManifest 配置

消息接收 MessageIntentService 配置

创建消息接收 Service，继承自 cn.paxpay.push.sevice.PaxMessageIntentService，并在

对应回调中添加业务处理逻辑,可参考以下代码：

```
    public class MyMessageIntentService extends PaxMessageIntentService {
    private final static String TAG = MyMessageIntentService.class.getSimpleName();

    @Override
```

```
    public void onDestroy() {
        super.onDestroy();
        Log.d(TAG,"onDestroy");
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG,"onCreate");
    }

    @Override
    public void onNotification(Context context, String title, String summary, Map<String, String>
extraMap) {
        Log.d(TAG, "收到一条推送通知：  " + title + ", summary: " + summary);
    }

    @Override
    public void onMessage(Context context, CPushMessage cPushMessage) {
        Log.d(TAG, "收到一条推送消息：  " + cPushMessage.getTitle() + ", content: " +
cPushMessage.getContent());
    }

    @Override
    public void onNotificationOpened(Context context, String title, String summary, String
extraMap) {
        Log.d(TAG, "onNotificationOpened title: " + title + ", summary: " + summary);
    }

    @Override
    public void onNotificationClickedWithNoAction(Context context, String title, String
summary, String extraMap) {
        Log.d(TAG, "onNotificationClickedWithNoAction title: " + title + ", summary: " +
summary);
    }

    @Override
    public void onNotificationRemoved(Context context, String messageId) {
        Log.d(TAG, "onNotificationRemoved messageId: " + messageId);
    }

    @Override
```

```
    public void onNotificationReceivedInApp(Context context, String title, String summary,
Map<String, String> extraMap, int openType, String openActivity, String openUrl) {
        Log.d(TAG, "onNotificationReceivedInApp title: " + title + ", summary: " + summary +
", openType:" + openType + ", openActivity:" + openActivity + ", openUrl:" + openUrl);
    }
}
```

将该 Service 添加到 AndroidManifest.xml 中

```
<service
    android:name=".MyMessageIntentService"
    android:exported="false">
    <intent-filter>
        <action android:name="cn.paxpay.push.NOTIFICATION_OPENED" />
    </intent-filter>
    <intent-filter>
        <action android:name="cn.paxpay.push.NOTIFICATION_REMOVED" />
    </intent-filter>
    <intent-filter>
        <action android:name="cn.paxpay.push.RECEIVE" />
    </intent-filter>
</service>
```

在应用中注册和启动移动推送

调用 register()初始化并注册云推送通道，并确保 Application 上下文中进行初始化工作。

请参照以下代码段进行初始化：

```
        public class MyApplication extends Application {
            private static final String TAG = MyApplication.class.getSimpleName();
            public static String clientId = "";

            @Override
            public void onCreate() {
                super.onCreate();

                Log.d(TAG, "MyApplication startService");

                // 设置接收消息 service
                PaxMqttService.setPushIntentServiceCls(MyMessageIntentService.class);
```

```
        // 设置通知图标
        PaxMqttService.setIconRes(R.mipmap.ic_launcher);
        // 设置认证主机地址
        PaxMqttService.setAuthHost("http://192.168.13.130:9001");
        // 注册推送服务
        PaxMqttService.register(getString(R.string.app_name), // appName
                "A8000001", // appId
                "123321", // appSecret
                this, // application context
                new CommonCallback() { // 注册回调
            @Override
            public void onSuccess(String response) {
                Log.d(TAG, "register onSuccess: " + response);
                clientId = response; // clientId
            }

            @Override
            public void onFailed(String errorCode, String errorMessage) {
                Log.d(TAG, "register onFailed: errorCode-" + errorCode + "
errorMessage-" + errorMessage);
            }
        });

        // Android 8.0 通知需要创建通知渠道
        createNotificationChannel(this);
    }

    private void createNotificationChannel(Context applicationContext) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationManager mNotificationManager = (NotificationManager)
applicationContext.getSystemService(Context.NOTIFICATION_SERVICE);
            if (mNotificationManager == null) {
                return;
            }
            // 通知渠道的 id
            String id = "1";
            // 用户可以看到的通知渠道的名字.
            CharSequence name = "notification channel";
            // 用户可以看到的通知渠道的描述
            String description = "notification description";
            int importance = NotificationManager.IMPORTANCE_HIGH;
```

```
                NotificationChannel mChannel = new NotificationChannel(id, name,
importance);
                // 配置通知渠道的属性
                mChannel.setDescription(description);
                // 设置通知出现时的闪灯（如果 android 设备支持的话）
                mChannel.enableLights(true);
                mChannel.setLightColor(Color.RED);
                // 设置通知出现时的震动（如果 android 设备支持的话）
                mChannel.enableVibration(true);
                mChannel.setVibrationPattern(new long[]{100, 200, 300, 400, 500, 400, 300,
200, 400});
                //最后在 notificationmanager 中创建该通知渠道
                mNotificationManager.createNotificationChannel(mChannel);
            }
        }
}
```