uniapp 安卓友盟消息推送原生插件

目　录

使用方法

介绍

安卓友盟+消息推送原生插件，支持自定义响铃，震动，免打扰时间段，厂商离线推送等
具体功能介绍和设备采集信息请参考官方文档
官方文档：https://developer.umeng.com/docs/67966/cate/67966

联系作者

关注微信公众号可联系作者

插件地址

https://ext.dcloud.net.cn/plugin?id=14421

用法

在需要使用插件的页面加载以下代码

```
const module = uni.requireNativePlugin("leven-umeng-upush-PushModule");
```

申请权限

无

页面内容

```
<template>
  <view>
    <uni-card title="友盟消息推送原生插件">
      <view style="margin-bottom: 20px;">
        <button type="primary" @click="init">初始化</button>
        <button type="primary" @click="registerPush">注册推送</button>
        <button type="primary" @click="getDeviceToken">获取deviceToken</button>
        <button type="primary" @click="setNotificationOnForeground">设置app前台是否
        <button type="primary" @click="setDisplayNotificationNumber">设置显示通知的
        <button type="primary" @click="setNotificationPlay">设置通知提醒</button>
        <button type="primary" @click="setNoDisturbMode">设置免打扰时间段</button>
        <button type="primary" @click="setMuteDurationSeconds">设置冷却时间</button>
        <button type="primary" @click="onNotificationMessage">监听通知消息</button>
        <button type="primary" @click="addTags">添加标签</button>
        <button type="primary" @click="deleteTags">删除标签</button>
        <button type="primary" @click="getTags">获取服务端所有标签</button>
        <button type="primary" @click="addAlias">添加别名</button>
        <button type="primary" @click="setAlias">别名绑定</button>
        <button type="primary" @click="deleteAlias">移除别名</button>
        <button type="primary" @click="setBadgeNum">设置角标数字</button>
        <button type="primary" @click="changeBadgeNum">角标数字递增减</button>
        <button type="primary" @click="notificationStatus">获取弹出通知权限状态</butt
        <button type="primary" @click="openNotificationSettings">打开通知权限设置界面
        <button type="primary" @click="getThirdToken">获取厂商Token</button>
        <button type="primary" @click="setInAppMsgDebugMode">设置应用内消息模式</but
        <button type="primary" @click="setCardMessage">设置插屏消息</button>
        <button type="primary" @click="setPlainTextMessage">设置纯文本插屏</button>
        <button type="primary" @click="setEnable">开启消息推送</button>
        <button type="primary" @click="setDisable">关闭消息推送</button>
        <button type="primary" @click="registerThirdPush">厂商通道注册</button>
      </view>
    </uni-card>
  </view>
  </view>
</template>

<script>
  const module = uni.requireNativePlugin("leven-umeng-upush-PushModule");
  export default {
    data() {
      return {}
    },
    methods: {
      // 初始化
      init() {
        module.init({
```

```
        appKey: "64ed58015488fe7b3affc3f9",
        messageSecret: "607e6d90a6f047d6a8eeea59c32d0bc0",
        channel: "android",
        deviceType: 1,
        logEnabled: true
      }, res => {
        console.log(res)
      })
    },
    // 一键初始化
    registerPush() {
      module.registerPush(res => {
        console.log(res)
      })
    },
    // 初始化
    getDeviceToken() {
      module.getDeviceToken(res => {
        console.log(res)
      })
    },
    // 设置app前台是否显示通知
    setNotificationOnForeground() {
      module.setNotificationOnForeground({
        isNotification: true
      }, res => {
        console.log(res)
      })
    },
    // 设置显示通知的数量
    setDisplayNotificationNumber() {
      module.setDisplayNotificationNumber({
        number: 5
      }, res => {
        console.log(res)
      })
    },
    // 设置通知提醒
    setNotificationPlay() {
      module.setNotificationPlay({
        sound: 0,
        lights: 1,
        vibrate: 2
      }, res => {
        console.log(res)
      })
    },
    // 设置通知提醒
    setNoDisturbMode() {
      module.setNoDisturbMode({
```

```
      startHour: 11,
      startMinute: 30,
      endHour: 20,
      endMinute: 0
    }, res => {
      console.log(res)
    })
  },
  // 设置冷却时间
  setMuteDurationSeconds() {
    module.setMuteDurationSeconds({
      seconds: 10,
    }, res => {
      console.log(res)
    })
  },
  // 监听通知消息
  onNotificationMessage() {
    module.onNotificationMessage(res => {
      console.log(res)
    })
  },
  // 添加标签
  addTags() {
    module.addTags({
      tags: ['标签1', '标签2', '标签3']
    }, res => {
      console.log(res)
    })
  },
  // 删除标签
  deleteTags() {
    module.deleteTags({
      tags: ['标签1', '标签2']
    }, res => {
      console.log(res)
    })
  },
  // 获取服务端所有标签
  getTags() {
    module.getTags(res => {
      console.log(res)
    })
  },
  // 添加别名
  addAlias() {
    module.addAlias({
      alias: "Android1",
      aliasType: "Android"
    }, res => {
```

```
      console.log(res)
    })
  },
  // 别名绑定
  setAlias() {
    module.setAlias({
      alias: "Android1",
      aliasType: "Android"
    }, res => {
      console.log(res)
    })
  },
  // 删除别名
  deleteAlias() {
    module.deleteAlias({
      alias: "Android1",
      aliasType: "Android"
    }, res => {
      console.log(res)
    })
  },
  // 设置角标数字
  setBadgeNum() {
    module.setBadgeNum({
      number: 5
    }, res => {
      console.log(res)
    })
  },
  // 角标数字递增减
  changeBadgeNum() {
    module.changeBadgeNum({
      number: -1
    }, res => {
      console.log(res)
    })
  },
  // 获取弹出通知权限状态
  notificationStatus() {
    module.notificationStatus(res => {
      console.log(res)
    })
  },
  // 打开通知权限设置界面
  openNotificationSettings() {
    module.openNotificationSettings(res => {
      console.log(res)
    })
  },
  // 获取厂商Token
```

使用方法

```
    getThirdToken() {
      module.getThirdToken(res => {
        console.log(res)
      })
    },
    // 设置应用内消息模式
    setInAppMsgDebugMode() {
      module.setInAppMsgDebugMode({
        mode: 2
      }, res => {
        console.log(res)
      })
    },
    // 设置插屏消息
    setCardMessage() {
      module.setCardMessage({
        label: "这是插屏消息"
      }, res => {
        console.log(res)
      })
    },
    // 设置纯文本插屏
    setPlainTextMessage() {
      module.setPlainTextMessage({
        titleTextSize: 16,
        contentTextSize: 14,
        buttonTextSize: 14
      }, res => {
        console.log(res)
      })
    },
    // 开启推送
    setEnable() {
      module.setEnable(res => {
        console.log(res)
      })
    },
    // 开启推送
    setDisable() {
      module.setDisable(res => {
        console.log(res)
      })
    },
    // 开启推送
    registerThirdPush() {
      module.registerThirdPush({
        xiaomi: {
          appid: "abc",
          appkey: "def"
        },
```

```
            meizu: {
              appid: "abc",
              appkey: "def"
            },
            oppo: {
              appid: "abc",
              appkey: "def"
            }
          }, res => {
            console.log(res)
          })
        },
      }
    }
</script>

<style>

</style>
```

更新日志

2023-09-02

首次发布

错误码

错误码	描述
-1	默认失败
-100	appKey为空
-101	未初始化
-102	注册失败
-103	Message Secret为空
-104	未注册消息推送

初始化

方法名

`init`

用法

- 用法如下：

```
module.init({
  appKey: "64ed58015488fe7b3affc3f9",
  messageSecret: "607e6d90a6f047d6a8eeea59c32d0bc0",
  channel: "android",
  deviceType: 1,
  logEnabled: true
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
appKey	String	是	无	友盟申请的appKey
messageSecret	String	是	无	友盟申请的messageSecret
channel	String	否	空	渠道
deviceType	Integer	否	1	设备类型，1：手机，2：盒子
logEnabled	Boolean	否	false	是否开启日志，如果开启会有Logcat打印

回调

- 示例

初始化

```
{
    "data": {},
    "message": "初始化成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

注册推送

方法名

`registerPush`

用法

- 用法如下：

```
module.registerPush(res => {
  console.log(res)
})
```

- 参数说明

  无

回调

- 示例

```
{
    "data": {
        "deviceToken": "AoK10qhpkf_Dnjyt5EcUoB76qeRvUgbJUbMfhxqhLKPQ"
    },
    "message": "注册成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
data.deviceToken	String	注册成功后的设备的唯一标识
code	Integer	返回类型，0.成功，其他：失败

获取deviceToken

方法名

`getDeviceToken`

用法

- 用法如下：

```
module.getDeviceToken(res => {
  console.log(res)
})
```

- 参数说明

  无

回调

- 示例

```
{
    "data": {
        "deviceToken": "AoK10qhpkf_Dnjyt5EcUoB76qeRvUgbJUbMfhxqhLKPQ"
    },
    "message": "成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
data.deviceToken	String	注册成功后的设备的唯一标识
code	Integer	返回类型，0.成功，其他：失败

设置app前台是否显示通知

方法名

`setNotificationOnForeground`

用法

- 用法如下：

```
module.setNotificationOnForeground({
  isNotification: true
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
isNotification	Boolean	否	true	是否显示通知

> 默认情况下，应用在前台是显示通知的

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象

设置app前台是否显示通知

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：失败

设置显示通知的数量

方法名

`setDisplayNotificationNumber`

用法

- 用法如下：

```
module.setDisplayNotificationNumber({
  number: 5
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
number	Integer	否	0	数量

> 参数number可以设置为0~10，当参数为0时，表示不限制显示个数

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象

设置显示通知的数量

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：失败

设置通知提醒

方法名

`setNotificationPlay`

用法

- 用法如下：

```
module.setNotificationPlay({
  sound: 0,
  lights: 1,
  vibrate: 2
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
sound	Integer	否		响铃声音控制方式，0：服务端控制，1.客户端控制，2.禁止响铃
lights	Integer	否		呼吸灯控制方式，0：服务端控制，1.客户端控制，2.禁止点亮
vibrate	Integer	否		振动控制方式，0：服务端控制，1.客户端控制，2.禁止震动

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

设置免打扰时间段

方法名

`setNoDisturbMode`

用法

- 用法如下：

```
module.setNoDisturbMode({
  startHour: 11,
  startMinute: 30,
  endHour: 20,
  endMinute: 0
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
startHour	Integer	否	23	开始小时
startMinute	Integer	否	0	开始分钟
endHour	Integer	否	7	结束小时
endMinute	Integer	否	0	结束分钟

> 本例为：11:30-20:00为免打扰时间段，如果全部设置为0则取消免打扰

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

设置冷却时间

方法名

`setMuteDurationSeconds`

用法

- 用法如下：

```
module.setMuteDurationSeconds({
  seconds: 10,
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
seconds	Integer	是	无	冷却时间，单位：秒

> 默认情况下，同一台设备在1分钟内收到同一个应用的多条通知时，不会重复提醒

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

设置冷却时间

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

监听通知消息

方法名

`onNotificationMessage`

> 如果有自定义参数将通过该方法返回，自定义参数可参考下方说明，该方法为监听方法，每次推送到客户端时都会有数据返回

用法

- 用法如下：

```
module.onNotificationMessage(res => {
  console.log(res)
})
```

- 参数说明

  无

回调

- 示例

```
{
    "data": {
        "list": {
            "key2": "value2",
            "key1": "value1"
        }
    },
    "message": "收到通知",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

监听通知消息

参数名	参数类型	参数描述
data	Object	数据对象
data.list	Object	自定义参数返回数据
code	Integer	返回类型，0.成功，其他：失败

友盟自定义参数

开发者在【友盟+】后台推送通知和自定义消息时，可以添加自定义参数，如下图：

添加标签

方法名

`addTags`

用法

- 用法如下：

```
module.addTags({
  tags: ['标签1', '标签2', '标签3']
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
tags	Array	是	无	标签集合

> tags名称请不要加入URL Encode等变换处理，请使用原生字符串。 目前每个用户tag限制在1024个， 每个tag 最大128字符。tag需使用半角字符，大小写敏感，tag中请不要使用逗号（,）双竖线（||）。

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

添加标签

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

删除标签

方法名

`deleteTags`

用法

- 用法如下：

```
module.deleteTags({
  tags: ['标签1', '标签2']
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
tags	Array	是	无	标签集合

> tags名称请不要加入URL Encode等变换处理，请使用原生字符串。 目前每个用户tag限制在1024个， 每个tag 最大128字符。tag需使用半角字符，大小写敏感，tag中请不要使用逗号（,）双竖线（||）。

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

删除标签

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

获取服务端所有标签

方法名

用法

- 用法如下：

```
```

- 参数说明
  无

回调

- 示例

```
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
data.list	Array	标签集合
code	Integer	返回类型，0.成功，其他：失败

添加别名

方法名

`addAlias`

> 别名增加，将某一类型的别名ID绑定至某设备，老的绑定设备信息还在，别名ID和device_token是一对多的映射关系
> 默认单个alias下同时生效的deviceToken数最多10个，pro可调整，需要绑定大量设备（>1k)的场景用tag更合适

用法

- 用法如下：

```
module.addAlias({
  alias: "Android1",
  aliasType: "Android"
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
alias	String	是	无	别名，最大长度128个字节
aliasType	String	是	无	别名类型，最大长度64个字节

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

添加别名

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

别名绑定

方法名

`setAlias`

> 别名绑定，将某一类型的别名ID绑定至某设备，老的绑定设备信息被覆盖，别名ID和deviceToken是一对一的映射关系

用法

- 用法如下：

```
module.setAlias({
  alias: "Android1",
  aliasType: "Android"
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
alias	String	是	无	别名，最大长度128个字节
aliasType	String	是	无	别名类型，最大长度64个字节

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

移除别名

方法名

`deleteAlias`

用法

- 用法如下：

```
module.deleteAlias({
  alias: "Android1",
  aliasType: "Android"
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
alias	String	是	无	别名，最大长度128个字节
aliasType	String	是	无	别名类型，最大长度64个字节

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

移除别名

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

设置角标数字

方法名

`setBadgeNum`

用法

- 用法如下：

```
module.setBadgeNum({
  number: 5
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
number	Integer	是	无	数值

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

角标数字递增减

方法名

`changeBadgeNum`

用法

- 用法如下：

```
module.changeBadgeNum({
  number: -1
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
number	Integer	是	无	增或减的值

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

获取弹出通知权限状态

方法名

`notificationStatus`

用法

- 用法如下：

```
module.notificationStatus(res => {
  console.log(res)
})
```

- 参数说明
  无

回调

- 示例

```
{
    "data": {
        "status": true
    },
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
data.status	Boolean	状态，true：已获取，false：未获取
code	Integer	返回类型，0.成功，其他：失败

打开通知权限设置界面

方法名

`openNotificationSettings`

用法

- 用法如下：

```
module.openNotificationSettings(res => {
  console.log(res)
})
```

- 参数说明
  无

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

获取厂商Token

方法名

`getThirdToken`

> 如果注册过厂商通道则会返回数据，否则不会有任何返回

用法

- 用法如下：

```
module.getThirdToken(res => {
  console.log(res)
})
```

- 参数说明
  无

回调

- 示例

```
{
    "data": {
        "type":"xiaomi",
        "token":"123456"
    },
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
data.type	String	厂商类型

参数名	参数类型	参数描述
data.token	String	厂商token
code	Integer	返回类型，0.成功，其他：失败

设置应用内消息模式

方法名

`setInAppMsgDebugMode`

用法

- 用法如下：

```
module.setInAppMsgDebugMode({
  mode: 2
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
mode	Integer	否	1	消息模式

> mode的值
> 1.线上模式（请求服务器的最小间隔是30分钟）
> 2.测试模式（请求服务器的最小间隔是1秒）

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

设置插屏消息

方法名

`setCardMessage`

用法

- 用法如下：

```
module.setCardMessage({
  label: "这是插屏消息"
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
label	String	否	通知	标签

> label是插屏消息的标签，用来标识该消息。 客户端需先调用setCardMessage，把label发送到服务器，之后U-Push后台【展示位置】才会出现可选label。 以label为单位，生产模式请求服务器的最小间隔是30分钟，测试模式的最小间隔是1秒。

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示

设置插屏消息

参数名	参数类型	参数描述
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

设置纯文本插屏

方法名

`setPlainTextMessage`

用法

- 用法如下：

```
module.setPlainTextMessage({
  titleTextSize: 16,
  contentTextSize: 14,
  buttonTextSize: 14
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
titleTextSize	Integer	否	18	标题字号大小
contentTextSize	Integer	否	16	内容字号大小
buttonTextSize	Integer	否	16	按钮字号大小

> 单位为sp

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

设置纯文本插屏

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

开启消息推送

方法名

`setEnable`

用法

- 用法如下：

```
module.setEnable(res => {
  console.log(res)
})
```

- 参数说明
  无

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

关闭消息推送

方法名

`setDisable`

用法

- 用法如下：

```
module.setDisable(res => {
  console.log(res)
})
```

- 参数说明
  无

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

厂商通道注册

方法名

`registerThirdPush`

用法

- 用法如下：

```
module.registerThirdPush({
  xiaomi: {
    appid: "abc",
    appkey: "def"
  },
  meizu: {
    appid: "abc",
    appkey: "def"
  },
  oppo: {
    appid: "abc",
    appkey: "def"
  }
}, res => {
  console.log(res)
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
xiaomi	Object	否	无	小米厂商通道注册
xiaomi.appid	String	是	无	小米的appId
xiaomi.appkey	String	是	无	小米的appkey
meizu	Object	否	无	魅族厂商通道注册
meizu.appid	String	是	无	魅族的appId
meizu.appkey	String	是	无	魅族的appkey

参数名	参数类型	是否必填	默认值	参数描述
oppo	Object	否	无	oppo厂商通道注册
oppo.appid	String	是	无	oppo的appId
oppo.appkey	String	是	无	oppo的appkey

> 其他厂商的注册（华为，vivo，荣耀）需要在插件中注册，请参考下方说明

回调

- 示例

```
{
    "data": {},
    "message": "操作成功",
    "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

其他厂商注册

华为、vivo和荣耀需要在插件的参数中设置，如下图所示：

填写完成后打包后会自动集成，其他的厂商需要在上面的api中设置，目前支持：华为、荣耀、小米、vivo、oppo、魅族的集成