
vivo 接入文档

1. 目录

vivo 接入文档.....	1
1 接入简介.....	3
2 接口业务背景.....	3
3 服务端接入.....	3
3.1 交互协议.....	3
3.2 接口说明.....	3
3.2.1 请求参数.....	3
3.2.2 返回字段.....	4
3.3 签名算法说明.....	4
3.3.1 sign 加密密钥 secret 获取.....	4
3.3.3 签名计算方法.....	4
3.3.3 请求参数格式.....	7
3.3.4 返回数据格式.....	7
3.4 返回码.....	7
4 客户端接入.....	8
4.1 客户端接入描述.....	8

4.2 备注	9
--------------	---

4.2 常见问题	9
----------------	---

VIVO接入文档

1 接入简介

本文档提供接入 vivo 使用说明

2 接口业务背景

接入“商店更新”能力，可实现应用在更新时跳转至应用商店自动更新安装，避免了用户在应用自更新时需要多次确认的繁琐，从而优化应用版本更新时的体验，提升版本更新效率和覆盖度。

3 服务端接入

3.1 交互协议

- 双方服务器之间的接口交互：
- 使用 http 协议，post 方式
 - 接口请求参数格式为 json
 - 接口返回数据格式为 json

3.2 接口说明

3.2.1 请求参数

接口地址：	https://developer-api.vivo.com.cn/gateway/developer/app/publish/v1/get-online-version		
请求方式	HTTP		
Header 请求头设置	content-type: application/json		
参数名称	类型	是否必填	字段说明
account	String	是	开发者帐号
timestamp	Long	是	时间戳（毫秒）示例：1567945333425
method	String	否	请求接口名，默认值（get-online-version）
sign	String	是	用 HMAC-SHA256 算法计算的签名
packageName	String	是	应用包名
androidVersion	String	否	安卓版本。示例：7.0.1
model	String	否	机型。示例：vivo X27

注：为确保用户更新体验，请尽量上传安卓版本和机型参数，以防出现部分机型用户无法更新的状况。

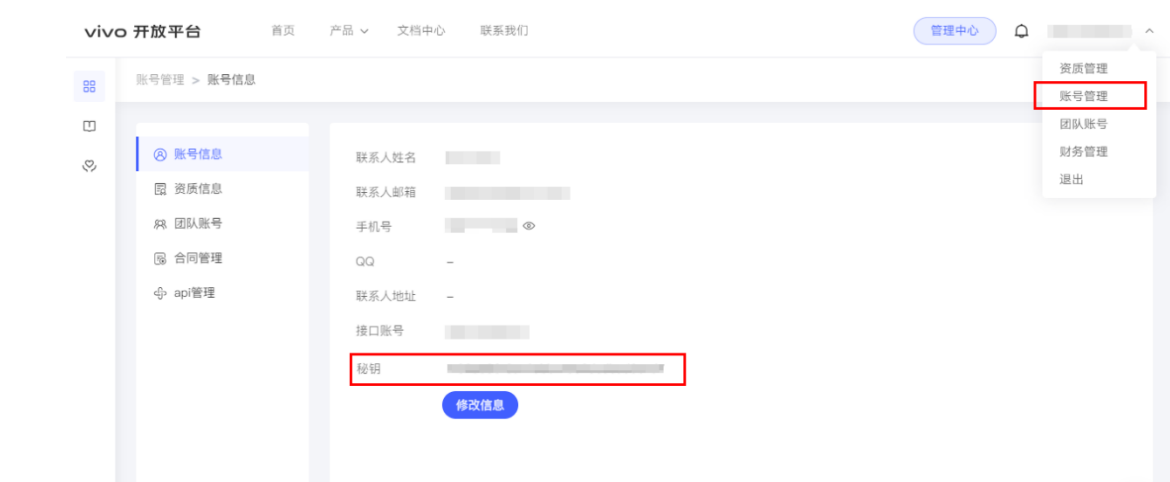
3.2.2 返回字段

参数名称	类型	是否必填	字段说明
code	int	是	返回码，详细说明见 返回码
msg	String	是	结果描述
data	Object	否	返回结果对象
packageName	String	否	包名
versionName	String	否	版本名称，示例：3.2.1
versionCode	Long	否	版本号，示例：100

3.3 签名算法说明

3.3.1 sign 加密密钥 secret 获取

vivo 开放平台给每个开发者帐号分配一个独立的 secret 密钥，用来做参数加密验签。开发者可登录开放平台，进入账号信息页面即可获取 secret，如下图：



3.3.3 签名计算方法

1.由公共参数（account、timestamp）和业务参数（bizParam）按照 scii 码排序后，根据字段顺序进行拼串，字段之间通过“&”相连接。

2.通过上面得到的参数字符串后将参数转换成 byte[]数组。登录开放平台后在个人信息中获取到平台的分配的 secret，然后将 secret 也转换成 byte[]数组。进行 HmacSHA256 加密，得到签名 sign。

3. (java) 签名加密代码示例：

```
/**
 * 获取加密验签
 */
public static void getSign() {
    String secret = "xxxxxxxxxxxxxxxxxxxxxxxx";

    Map<String, Object> parmas = Maps.newHashMap();
    parmas.put("account", "test");
    parmas.put("timestamp", System.currentTimeMillis());
    parmas.put("bizParam", getBizParam());
    String sign = HmacSHA256(getUrlParamsFromMap(parmas), secret);
    System.out.println("得到加密后的验签: " + sign);
}

public static Map<String, String> getBizParam() {
    Map<String, String> bizParams = Maps.newHashMap();
    bizParams.put("packageName", "com.xxx.xxx");
    bizParams.put("androidVersion", "4.0.1");
    bizParams.put("model", "vivo x20");
    return bizParams;
}

/**
 * 根据传入的map, 把map里的key value转换为接口的请求参数, 并给参数按ascii码排序
 *
 * @param paramsMap 传入的map
 * @return 按ascii码排序的参数键值对拼接结果
 */
public static String getUrlParamsFromMap(Map<String, Object> paramsMap) {
    List<String> keysList = new ArrayList<>(paramsMap.keySet());
    Collections.sort(keysList);
    StringBuilder sb = new StringBuilder();
    for (String key : keysList) {
        if ("sign".equals(key)) {
            continue;
        }
        Object object = paramsMap.get(key);
        if (object == null) {
            continue;
        }
        sb.append(key).append("=");
        if ("bizParam".equals(key)) {
            sb.append(JSON.toJSONString(object)).append("&");
        }
    }
}
```

```

        } else {
            sb.append(object).append("&");
        }
    }
    if (sb.length() > 0) {
        sb.deleteCharAt(sb.length() - 1);
    }
    return sb.toString();
}

/**
 * HMAC_SHA256 验签加密
 */
public static String HmacSHA256(String data, String key) {
    try {
        byte[] secretByte = key.getBytes(Charset.forName("UTF-8"));
        SecretKeySpec signingKey = new SecretKeySpec(secretByte, "HmacSHA256");
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(signingKey);
        byte[] dataByte = data.getBytes(Charset.forName("UTF-8"));
        byte[] by = mac.doFinal(dataByte);
        return byteArr2HexStr(by);
    } catch (Exception e) {
        log.error("HmacSHA256异常, data={},", data, e);
    }
    return null;
}

private static String byteArr2HexStr(byte[] arrB) {
    int iLen = arrB.length;
    // 每个byte用两个字符才能表示, 所以字符串的长度是数组长度的两倍
    StringBuilder sb = new StringBuilder(iLen * 2);
    for (int i = 0; i < iLen; i++) {
        int intTmp = arrB[i];
        // 把负数转换为正数
        while (intTmp < 0) {
            intTmp = intTmp + 256;
        }
        // 小于0F的数需要在前面补0
        if (intTmp < 16) {
            sb.append("0");
        }
        sb.append(Integer.toString(intTmp, 16));
    }
}

```

```
        return sb.toString();
    }
```

3.3.3 请求参数格式

```
{
  "account": "xxxxxxxxxxxx",
  "timestamp": 156794533425,
  "sign": "2b29665baed6bde9b82e44300c8738d1ae59833a5f4ea029cce52e46f95b8954",
  "bizParam": {
    "packageName": "com.xx.xx",
    "model": "vivo X27",
    "androidVersion": "7.0.1"
  }
}
```

3.3.4 返回数据格式

成功:

```
{
  "code": 0,
  "msg": "success",
  "data": {
    "packageName": "com.xxx.xxx",
    "versionName": "3.2.1",
    "versionCode": "100"
  }
}
```

失败:

```
{
  "code": 10001,
  "msg": "签名校验失败"
}
```

3.4 返回码

返回码 Code	描述
0	成功
404	接口不存在

405	不允许的 HTTP 请求方法
440	缺少参数
441	请求参数错误
500	服务器错误
10001	签名校验失败
10002	业务请求参数不能为空
10003	请求查询异常，请稍后再试
10004	没有接口访问权限
10005	timestamp 时间戳失效
10006	请求频次过高，请稍后再试
11001	根据包名查询不到应用信息
11002	应用已下架
11003	应用没有在当前机型上架
11004	应用没有在当前安卓系统版本上架
11005	开发者帐号不存在

4 客户端接入

4.1 客户端接入描述

在判断商店（包名为 `com.bbkk.appstore`）已安装且版本号大于等于 3100（8.1 版本）时调用以下方法即可实现调起详情页自动更新。具体调用代码如下

```
/**
 * 跳转到商店详情页自动更新，请按照商店版本要求自行控制调用此方法的前置条件
 * @param context 上下文
 */
public static void jumpToAppStoreDetailUpdate(Context context) {
    Intent intent = new Intent();
    String packageName = context.getPackageName();
    // th_update_delay=1代表使用默认接口方案（预约闲时更新），如果移除该参数代表使用备选接口方案（立即更新）
    String url = "vivomarket://details?id=" + packageName
        + "&th_name=self_update&th_update_delay=1";
    Uri uri = Uri.parse(url);
    intent.setData(uri);
    intent.setPackage("com.bbkk.appstore");
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
        Intent.FLAG_ACTIVITY_CLEAR_TOP);
    context.startActivity(intent);
}
```

```
}
```

4.2 备注

- 1) 仅支持客户端集成，不支持 h5 使用 deeplink 方式调起详情页自动更新。
- 2) 请务必判断商店版本为 8.1 或以上才调起商店更新(versionCode:3100) 。
- 3) 调用前请确保您的最新更新包已在 vivo 应用商店上架并审核通过。
- 4) 对于个别同一个应用对应多个包名的 cp，需要自行判断仅在 vivo 应用商店上架的包名对应的包才能采用本方式更新。
- 5) 默认接口方案：当应用正在前台使用时，应用商店将在闲时为用户自动安装应用，不会因直接安装导致应用闪退，打断用户行为。该功能从 9.12.60.0 版本开始支持。

4.2 常见问题

1、调起详情页后显示网络异常，无法加载出详情页。

- (1) 确认网络正常。
- (2) 确认调用上面代码的调用方包名已上架 vivo 应用商店并审核通过。

2、能正常展示详情页，但没有自动开始更新，需要手动点。

- (1) 确认商店版本为 8.1 或以上版本。
- (2) 如果下载按钮为暂停、继续、重试，则不自动继续下载是正常的。
- (3) 确认已在站内信或邮件中获取审核通过通知，且以上代码确实是在已审核通过的应用里调用的。
- (4) 确认系统 Android 版本为 Android 4.3 及以上