火舞联运平台SDK

`火舞联运平台` 游戏接入SDK文档 v1.2.2

应用接入

登记

- `游戏名称` : 用于前台展示的 `应用名称` , 请控制在 `10字以内`
- `图标` : 用于前台展示推广的 `ICON图` , 尺寸 `512 * 512` , 格式 `png`
- `应用地址` : 最终 `访问地址` (通常 `域名` 即可)
- `回调地址` : 用于接收 `事件回调通知` 的URL地址

> 目前 `联运前台` 系统还**未开放**服务, 以上登记信息暂由 `客服/接口人` 处理, 请多多谅解~

生成

> 应用**登记创建成功**后, 系统会生成用于开发的配置信息, 请**务必妥善保存**

- `appid` : 被分配的应用ID
- `secret` : 被分配的应用密钥

> 暂不支持 `重置修改` (`联运前台` 系统上线后**开放**), 若需修改可直接联系 `客服/接口人`

交互规范

- 请求方式: `POST`

  > 少量接口（如 `用户授权` ）因业务性质，使用 `GET` 方式

- 返回格式: 标准 JSON

- 成功、正常

```
{
    status: 1,
    data: "返回数据"
}
```

> `data` {Object} 取值参考各个接口的 `返回结果` 部分

- 失败、错误

```
{
    status: 0,
    code: 100,
    data: "发生未知错误"
}
```

> `code` {number} 取值参考 `错误码` 部分
>
> `data` {string} 针对错误的辅助描述

注意，考虑篇幅以下所有接口的 `返回数据` 均代表 `成功` 格式的 `data` 字段

错误码

全局类

- `100` : 未知错误
- `101` : 应用未注册
- `102` : 应用被冻结
- `103` : token无效
- `104` : token生成失败
- `105` : token更新失败
- `106` : 请求方式错误

帐号类

- `200` : 帐号或密码错误

- `201` : 帐号不存在
- `202` : 帐号已冻结
- `203` : 帐号未授权
- `204` : 帐号未登录

授权类

- `400` : 参数相关错误
- `401` : token相关错误
- `402` : 用户相关错误
- `403` : 应用相关错误
- `404` : 获取用户基本信息相关错误
- `405` : 刷新token相关错误

支付类

- `500` : 订单创建失败

用户授权

请求地址

- `正式` 地址：http://web.51h5.com/sso.html
- `测试` 地址：http://dev.web.51h5.com/sso.html

请求方式

`GET` (前台页面**重定向**)

请求参数

- `appid` : {string} {**必选**} 应用appid
- `redirect` : {string} {**必选**} 跳回地址

> 必须（MUST）进行 `URL编码` (如 `PHP` 的 `urlencode`), 以免引起不可预料的问题

- `login_type` : {string} {**可选**} 登录类型, 默认为 `火舞帐号登录`

  可选取值如下:

  - `weibo` : 微博登录
  - `qq` : QQ登录
  - `wechat` : 微信登录

- `force_login` :{number} {**可选**} 是否强制登录(`忽略` 登录状态检查)

  > 若为 `空` 则等价于 `0`

  - `0` : 否

    > 会自动检查玩家的 `登录状态` 和 `登录类型`
    >
    > 若 `已登录` 且 `类型` 与 `login_type` 匹配，则直接返回游戏, 反之则跳转到登录页

  - `1` : 是

    > 直接跳转到登录页（忽略登录状态）
    >
    > 如果有切换账户的需求，可以尝试该选项

返回

```
{
    "code": "xxxxx"
}
```

- `code` : {string}

  > 用于获取 `token` 的 `授权码` , 过期时间为 `2分钟`
  >
  > 会自动附加在 `回调地址` 里执行重定向, 参数名为 `code`

示例

游戏页面执行重定向

```
// 1.默认：不强制出现登录界面，非登录的用户，将出现火舞账户的登录界面
location.href = 'http://web.51h5.com/sso.html?appid=xxx&redirect=' + encodeURIComponent('http://xxx.com/?a=1&b=2&c=3');

// 2.自定义：QQ账户登录（看到的是QQ登录授权界面），强制出现登录界面
location.href = 'http://web.51h5.com/sso.html?appid=xxx&login_type=qq&force_login=1&redirect=' + encodeURIComponent('http://xxx.com/?a=1&b=2&c=3');
```

`授权成功` 后页面自动重定向

```
http://xxx.com/?a=1&b=2&c=3&code=xxxxxxx
```

获取token

> 该接口数据涉及 `隐私安全` , 请务必（MUST）保证仅在后台使用, 不要（SHOULD NOT）在前台页面暴露

请求地址

- `正式` 地址：http://api.web.51h5.com/auth/token
- `测试` 地址：http://dev.api.web.51h5.com/auth/token

请求方式

`POST`

请求参数

- `appid` : {string} {**必选**} 应用appid
- `sign` : {string} {**必选**} 签名（签名算法见**附录**）
- `code` : {string} {**必选**} 授权码(通过 `用户授权` 机制获取的)

返回结果

```
{
    "access_token": "xxxx",
    "refresh_token": "xxxx",
    "expire_in": xxxx
}
```

- `access_token` : {string} 授权 `token`

  > 用于获取用户信息等, 具有过期时间, 一般为 `2小时`

- `refresh_token` : {string} 刷新 `access_token` 凭证

  > 用于刷新**快过期**的 `access_token`

- `expire_in` : {number} `access_token` 失效时间, 单位为 `秒` , 例如：7200

刷新token

> 该接口数据涉及 `隐私安全` , 请务必（MUST）保证仅在后台使用, 不要（SHOULD NOT）在前台页面暴露

用于当前 `token` 快失效时, 重新获取刷新，而不用再重新走 `用户授权获取CODE` 的方式(会因跳转中断游戏)

请求地址

- `正式` 地址：http://api.web.51h5.com/auth/refresh
- `测试` 地址：http://dev.api.web.51h5.com/auth/refresh

请求方式

`POST`

请求参数

- `appid` : {string} {**必选**} 应用appid
- `sign` : {string} {**必选**} 签名（签名算法见**附录**）

- `refresh` : {string} {**必选**} 获取到的 `refresh_token`

返回结果

```
{
    "access_token": "xxxx",
    "refresh_token": "xxxx",
    "expire_in": xxxx
}
```

> 返回数据同 `获取token` 接口

获取用户基本信息

> 该接口数据涉及 `隐私安全` , 请务必（MUST）保证仅在后台使用, 不要（SHOULD NOT）在前台页面暴露

请求地址

- `正式` 地址：http://api.web.51h5.com/auth/info
- `测试` 地址：http://dev.api.web.51h5.com/auth/info

请求方式

`POST`

请求参数

- `appid` : {string} {**必选**} 应用appid
- `sign` : {string} {**必选**} 签名（签名算法见**附录**）
- `token` : {string} {**必选**} 获得的 `token`

返回结果

```
{
    "openid": "xxxx",
    "nick": "xxxx",
    "avatar": "xxxx",
    "gender": "xxxx",
    "province": "xxxx",
    "city": "xxxx"
}
```

- `openid` : {string} 用户ID

  > 用户在 `当前应用` 中的 `唯一` 标识，永不变，**建议用于关系映射**

- `nick` : {string} 昵称

  > 注意, 目前存在为 `空` 的情况

- `avatar` : {string} 头像地址

- `gender` : {number} 性别

  可能的取值

  - `1` : 男
  - `0` : 女

- `province` : {string=} 所在省份

- `city` : {string=} 所在城市

订单创建

> 该接口数据涉及 `隐私安全` , 请务必（MUST）保证仅在后台使用, 不要（SHOULD NOT）在前台页面暴露

请求地址

- `正式` 地址：http://api.web.51h5.com/pay/order

- `测试` 地址：http://dev.api.web.51h5.com/pay/order

请求方式

`POST`

请求参数

- `appid` : {string} {**必选**} 应用appid
- `sign` : {string} {**必选**} 签名（签名算法见**附录**）
- `token` : {string} {**必选**} 获得的 `access_token`
- `total_fee` : {number} {**必选**} 充值金额，单位 `元` ，必须（MUST） `整数`
- `subject` : {string} {**必选**} 商品名
- `body` : {string} {**必选**} 商品描述
- `server_id` : {number} {**可选**} 游戏区服,
- `exten` :{string} {**可选**} 透传数据

  > 字符串格式，最大长度256

返回结果

```
{
    "order_num": "xxxxxxx",
    "pay_url": "http://xxxx",
}
```

- `order_num` : {string} 订单号
- `pay_url` : {string} 充值地址的url，该地址只用于 `前台页面跳转`

充值

使用 `订单创建` 接口返回的 `pay_url` 进行**页面跳转**，完成充值

示例

> 订单创建接口返回

```
{
    "order_num": "123456",
    "pay_url": "http://test.com/?a=1&b=2&c=3",
}
```

> 前端页面执行跳转

```
location.href = 'http://test.com/?a=1&b=2&c=3';
```

事件通知

用于重要业务操作后, **火舞联运系统**自动发送相关通知到应用注册的 `回调地址`

通知方式

`POST`

通知请求

应用注册时填写的 `callback` 参数对应的地址

通知数据

```
{
    "notify_type": 1,
    "sign":"..."
    "...": "..."
}
```

- `notify_type` : {number}
  - `1` : 充值成功

> 目前通知类型较少, 未来会陆续补充

- `sign` :{string}
    - 数据签名，收到事件通知后需要进行签名验证，签名算法见附录

> 不同 `类型` , 包含的 `数据字段` 不同, 具体请参考对应类型的事件通知

通知重试

若遇到 `回调地址` 访问故障 或 无应答 等情况，通知请求会尝试重试 `最多5次` 后终止, 请务必（MUST）保证回调地址稳定性

通知应答

`回调地址` 必须（MUST）对通知请求作出 `明确应答` , 从而避免被**通知服务器**判为 `失败` 通知

应答内容:

- `success` : {string} 确认收到通知
- `fail` : {string} 否定通知, **通知服务器**会继续重试通知, 直到超过重试次数

示例(PHP):

```
// 确认收到通知
echo "success";

// 否定通知/希望继续通知
echo "fail";
```

事件通知-充值成功

用户 `充值成功` 后，火舞充值系统会 `自动` 调用应用注册时提供的回调地址 (`callback`), 并附带相关参数数据

通知数据

```
{
    "notify_type": 1,
    "sign":"xxxx",
    "type": "xxxx",
    "order_num": "xxxx",
    "openid": "xxxx",
    "amount": "xxxx",
    "server_id": "xxxx",
    "exten":"xxxx"
}
```

- `notify_type` : {number} 通知类型, 固定为 `1`

  > 每个事件通知必须包含 `notify_type` 参数, 这里不再详述, 详细参考 `事件通知` 部分

- `sign` :{string} 签名

  > 数据签名，算法见 `附录 - 签名算法` 部分

- `type` : {number} 充值类型
  - `2` : 该笔充值为离线充值，是用户在 `火舞充值中心` 直接为游戏充值

    > 请游戏最好支持此类型, 否则 `当前应用` 不会出现在 `火舞充值中心` 的在线直充列表里

  - `5` : 游戏内充值，是用户在游戏内充值
- `order_num` : {string} 订单号
- `openid` : {string} 用户对应该应用的唯一ID
- `amount` : {number} 充值金额
- `server_id` : {number} 游戏区服, 默认为 `0` ,
- `exten` :{string} 透传数据

附录

签名算法

生成步骤

1. 设所有发送或者接收到的数据为 `集合M`，将 `集合M` 内非空参数值的参数按照参数名**ASCII码从小到大排序**（字典序），使用**URL键值对**的格式（即 `key1=value1&key2=value2..`）拼接成字符串 `stringA`。
2. 在 `stringA` 最后拼接 `secret` (注册应用时分配得到)得到 `stringTemp`，再进行进行**MD5运算**，得到签名 `sign` 的值 `signValue`。

注意

> 参数 `sign` 不参与计算

举例

> 假设 `secret = "abcd"`，需要传送的参数如下

```
{
  appid: "123456",
  sparams1: "p1",
  fparams2: "p2",
  wparams3: "p3",
  aparams4: "p4",
}
```

1. 对参数按照 `key=value` 的格式，并按照参数名 `ASCII` 字典序排序如下：

   ```
   stringA = "aparams4=p4&appid=123456&fparams2=p2&sparams1=p1&wparams3=p3"
   ```

2. 拼接密钥 `secret`

   ```
   stringTemp = "aparams4=p4&appid=123456&fparams2=p2&sparams1=p1&wparams3=p3abcd"
   ```

3. 计算 `md5` 得到 `sign`

```
sign = "d15a7430b83bbc4dae16dc09f2bb8b41"
```

4. 最终传送的参数

```
{
    appid: "123456",
    sparams1: "p1",
    fparams2: "p2",
    wparams3: "p3",
    aparams4: "p4",
    sign: "d15a7430b83bbc4dae16dc09f2bb8b41"
}
```

PHP签名生成算法示例

```
function geneSign($params, $secret){
    // 去掉不能参与计算的值(如 sign)
    $para_filter = array();
    while (list ($key, $val) = each ($params)) {
        if ($key == "sign" || $val == "") {
            continue;
        }
        else {
            $para_filter[$key] = $params[$key];
        }
    }

    // 参数排序
    ksort($para_filter);
    reset($para_filter);

    // 连成字符串
    $arg  = "";
    while (list($key, $val) = each($para_filter)) {
        $arg .= $key . "=" . $val . "&";
    }

    // 去掉最后一个&字符
    $arg = substr($arg, 0, count($arg) - 2);

    // 如果存在转义字符，那么去掉转义
    if (get_magic_quotes_gpc()) {
        $arg = stripslashes($arg);
    }

    // 拼接secret & MD5运算 后，返回签名值
    return md5($arg . $secret);
}
```

NodeJs签名生成算法示例

```
var crypto = require('crypto');
var Buffer = require("buffer").Buffer;
function geneSign(params, secret){
    var stingA = "";

    for (var key of Object.keys(params).sort()) {
        if (!!!params[key] || key === 'sign') {
            continue;
        }

        if (!!!stingA.length) {
            stingA = key + "=" + params[key];
        }
        else {
            stingA += ("&" + key + "=" + params[key]);
        }
    }

    var strTemp = stingA + secret;
    var buf = new Buffer(strTemp);
    //保证汉字md5一致
    strTemp = buf.toString("binary");
    return crypto.createHash('md5').update(strTemp).digest(
'hex');
};
```