第三方接入公共参数说明

CHANGELOG

	版本	修改人	时间
初始化文档	v5.0	吴秋锋	2022-06-30

1.开发者账号申请

目前暂无公开的申请开发者账号的渠道，合作伙伴在签订了《***保密协议***》后，可以联系云迹相关负责人申请开通，该账号包括：

- **appname**: 合作伙伴的应用名称，可以由合作伙伴自己提供，如果未提供，云迹会随机生成一个
- **secret**: 调用接口时的密钥，合作伙伴须保护好该密钥，不要对外传播，也不要放在接口参数中传递

> **注意**， `appname` 在云迹开放平台内部需要保证全局唯一性，因此，如果合作伙伴自己提供 `appname` ，建议尽量使用自己的品牌、域名等专有名称，以减少冲突的可能性。

2.接口签名算法

云迹机器人开放平台提供的所有接口均为 `HTTPS` 接口，仅支持GET和POST两种方式（见具体的API文档），一般查询状态或数据为GET，控制操作为POST，调用接口时需要提供 `签名` 。

以下面的这个接口为例:

```
# 呼叫指定的机器人到502房间
POST /openapi/v1/robot/call?productId=HOTQY00SZ2000408155800001&target=502
```

上面的接口包含 `productId` 和 `target` 两个参数，只要有这两个参数就能完整实现将指定机器人召唤到指定目的地的功能。但在实际调用时，除了它们，还需要传三个参数:

参数名	含义
appname	从云迹申请获得的开发者账号名称

ts	调用接口时的时间戳，精确到 毫秒 ，云迹服务端会对调用时间进行验证，偏差超过10分钟的请求会被拒绝，因此调用者的服务器上要确保时间同步
sign	使用开发者账号的 密钥 生成的签名，云迹的服务端会用同样的算法进行签名计算并比较，签名不一致的请求会被拒绝

上面的这3个额外参数如果有缺失，调用接口时会返回如下错误:

```
{
    "errcode": 1,
    "errmsg": "必要参数缺失"
}
```

签名 sign 生成算法（以前面的接口为例）：

```
# 第1步，将接口自身的“业务”参数，按照参数名的字典顺序排序，并用“|”连接（顺序一定不能错）
#       空白参数不参与其中
#       实际调用时没有传的可选参数也不参与其中
productId:HOTQY00SZ200040815580001|target:502

# 第2步，再将appname、secret和ts按顺序依次拼在后面
productId:HOTQY00SZ200040815580001|target:502|appname:xxx|secret:b926a253863e501afef8755ad930a65b|ts:1500371626000

# 第3步，对上面第2步得到的字符串计算md5即得到sign的值（32位、小写）
sign=md5(productId:HOTQY00SZ200040815580001|target:502|appname:xxx|secret:b926a253863e501afef8755ad930a65b|ts:1500371626000)

# 第4步，最后调用接口时，传递的参数变成下面这样（调接口时传递的参数顺序没有要求）
POST /openapi/v1/robot/call?productId=HOTQY00SZ200040815580001&target=502&appname=xxx&ts=1500371626000&sign=55a8d9c7ac28cf06d83e3b38ae645232
```

示例代码（JAVA）：

```
import org.apache.commons.lang.StringUtils;
import org.springframework.util.DigestUtils;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;
```

```
public class SignUtils {
    public static String sign(Map<String, String> params, String appname, String secret, lon
g ts) {
        List<String> kvs = new ArrayList<>();
        for (Map.Entry<String, String> entry : params.entrySet()) {
            String key = StringUtils.trim(entry.getKey());
            String value = StringUtils.trim(entry.getValue());
            if (key.equalsIgnoreCase("appname") ||
                    key.equalsIgnoreCase("secret") ||    // secret一定不要通过参数传递!!
                    key.equalsIgnoreCase("ts") ||
                    key.equalsIgnoreCase("sign") ||
                    StringUtils.isBlank(value))          // 值为空的参数，不参与sign计算
                continue;

            kvs.add(key + ":" + value);
        }
        Collections.sort(kvs);

        kvs.add("appname:" + appname);
        kvs.add("secret:" + secret);
        kvs.add("ts:" + ts);

        return DigestUtils.md5DigestAsHex(StringUtils.join(kvs, "|").getBytes());
    }
}
```

参数为JSON时的签名算法

参数为JSON时，以下面的参数为例：

```
{
    "product": "ABC123",
    "query": {
        "keyword": "xyz",
        "start": 0,
        "count": 1
    }
}
```

计算签名时，只取第1层的参数，其它嵌套的都看成1个String，例如上面的例子里， `query` 里的参数又是个JSON，但不用解析，直接看作一个字符串。

> **注意**，嵌套的JSON转字符串时，要对里面所有同一层的，按名字排序，去掉非值内容中的空白符。

最后实际传的参数是如下格式：

```
{
    "product": "ABC123",
    "query": {
        "keyword": "xyz",
        "start": 0,
        "count": 1
    },
    "appname": "xxx",
    "ts": xxxx,
    "sign": "xxxxxx"
}
```

示例代码（JAVA）：

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.alibaba.fastjson.serializer.SerializerFeature;
import org.apache.commons.lang.StringUtils;
import org.springframework.util.DigestUtils;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;

public class SignUtils {
    public static String signJson(JSONObject params, String appname, String secret, long ts)
{
        List<String> kvs = new ArrayList<>();
        for (Map.Entry<String, Object> entry : params.entrySet()) {
            String key = StringUtils.trim(entry.getKey());
            String value = StringUtils.trim((entry.getValue().getClass().isPrimitive() || en
try.getValue() instanceof String) ?
                    String.valueOf(entry.getValue()) :
                    JSON.toJSONString(entry.getValue(), SerializerFeature.MapSortField));
            if (key.equalsIgnoreCase("appname") ||
                    key.equalsIgnoreCase("secret") ||
                    key.equalsIgnoreCase("ts") ||
                    key.equalsIgnoreCase("sign") ||
                    StringUtils.isBlank(value))
                continue;
```

```
            kvs.add(key + ":" + value);
        }
        Collections.sort(kvs);

        kvs.add("appname:" + appname);
        kvs.add("secret:" + secret);
        kvs.add("ts:" + ts);

        return DigestUtils.md5DigestAsHex(StringUtils.join(kvs, "|").getBytes());
    }
}
```

3.返回结果

所有接口返回结果均为 JSON ，其基本格式统一如下:

```
{
    "errcode": 0,
    "errmsg": "error",
    "result": ...
}
```

各字段含义如下:

字段名	含义
errcode	错误码，0表示调用无异常，非0表示调用时发生异常， errcode==0 并不表示业务功能一定执行成功，还要结合 result 的内容进一步判断
errmsg	异常消息， errcode!=0 时才会有该字段
result	errcode==0 时才会有该字段，不同接口内容不一样，具体的内容见各个接口的定义，比较老的接口里这个字段名为 data

> **注意**，云迹开放接口平台会保证不将任何 业务 异常抛给调用者的客户端，只要能够访问到开放平台，所有接口调用一定返回上面格式的内容，但客户端需要自己处理**网络异常**等相关情况。

4.接口地址

环境	地址
测试环境	https://openapi-test.yunjichina.com.cn
生产环境	https://openapi.yunjichina.com.cn