签名生成规则

> 获取userId 、 appId 、 appSecret关键信息，具体见获取开发密钥， `appSecret密钥用于生成签名，作为通信数据安全的关键信息，严禁保存在客户端直接使用，所有API都必须通过客户自己服务器中转调用POLYV服务器获取响应数据`

签名生成规则

sign的生成规则如下：

1. 将**全部**请求参数（参数值非空的参数）按key（参数名）**字典顺序**（ASCII值大小）升序排列
2. 将参数连接参数名与参数值拼成 key1value1key2value2...keyNvalueN
3. 将拼接字符串**首尾**加上appSecret
4. 对拼接字符串采用UTF-8编码计算**MD5**，将**MD5**结果转为**大写字母**，作为sign

注：

- appSecret只是拼在字符串首尾，并不参与key的排序
- appSecret只是计算签名时需要，在发送请求时不需要带appSecret
- 字符串拼接时需要将参数值为null的参数剔除
- 签名字符集必须为UTF-8，若不指定，可能采用平台默认字符集，导致错误

示例

1. 查询频道连麦使用量接口需要以下参数：

```
channelIds: 2477096,2272655
startDay: 2022-05-20
endDay: 2022-06-18
appId: g4rqgmmjuo
timestamp: 1660270926732
page: null
size: null
```

2. 以上非空参数按参数名字典排序后的顺序为：

```
appId: g4rqgmmjuo
channelIds: 2477096,2272655
endDay: 2022-06-18
startDay: 2022-05-20
timestamp: 1660270926732
```

3. 按以上顺序拼接字符串为：

```
appIdg4rqgmmjuochannelIds2477096,2272655endDay2022-06-18startDay2022-05-20timestamp1660270926732
```

4. 首尾拼上appSecret后的字符串为：（此处示例中appSecret：fsq2k5weced1h8vui657xtdva66whf0g为虚拟值）

```
fsq2k5weced1h8vui657xtdva66whf0gappIdg4rqgmmjuochannelIds2477096,2272655endDay2022-06-18startDay2022-05-20timestamp1660270926732fsq2k5weced1h8vui657xtdva66whf0g
```

5. 最后对字符串计算MD5并转换为大写得到sign值为：

```
0D2BDA2FD04D93A2B8832B91FD973C4D
```

注：如果按照此方式计算出来签名调用接口无法通过签名校验，请将上面第四步的字符串、第五步的MD5后的sign值和请求的接口地址，通过右下角在线客服提供给客服排查，我们将快速解决您的问题。

其他

SHA256签名算法（可选）

本平台也支持**SHA256**签名算法计算签名，具体加密计算方法如下：

1. 指定请求参数signatureMethod=SHA256，如无此参数默认使用MD5签名
2. 将请求参数（参数值非空的参数）按key（参数名）**字典顺序**（ASCII值大小）升序排列
3. 将参数连接参数名与参数值拼成 key1value1key2value2...keyNvalueN
4. 将拼接字符串**首尾**加上appSecret
5. 对拼接字符串采用UTF-8编码计算**SHA256**，将**SHA256**结果转为**大写字母**，作为sign

请求一次性校验参数（可选）

使用参数signatureNonce=唯一随机数，用于防止网络重放攻击，例如：signatureNonce=584F3849-E5A0-4B59-98A5-2F373EFD0559

快速接入基础代码请下载相关依赖源码， 点击下载源代码 ,下载后加入到自己的源码工程中即可。测试用例中的**HttpUtil.java 和 LiveSignUtil.java** 都包含在下载文件中。

> 强烈建议您使用直播Java SDK完成API的功能对接，直播Java SDK 对API调用逻辑、异常处理、数据签名、HTTP请求线程池进行了统一封装和优化。

```
package net.polyv.common;

import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;

import org.junit.Test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import net.polyv.util.LiveSignUtil;

/**
 * @author: thomas
 **/
public class LiveSignTest {

    private static final Logger log = LoggerFactory.getLogger(LiveSignTest.class);

    @Test
    public void buildSign() throws UnsupportedEncodingException, NoSuchAlgorithmException {
        String appId = "XXXXXXXX";
        String userId = "XXXXXXXX";
        String appSecret = "XXXXXXXXXXXXXXXXXXXXXXXX";

        long timestamp = System.currentTimeMillis();
        Map<String, String> paramMap = new HashMap<String, String>();
        // 公共参数
        paramMap.put("appId", appId);
        paramMap.put("timestamp", Long.toString(timestamp));
        // 业务参数
        paramMap.put("channelId", "2149813");

        // 一次性签名(可选参数)
        // paramMap.put("signatureNonce", UUID.randomUUID().toString());

        // MD5签名(默认)
        String sign = LiveSignUtil.getSign(paramMap, appSecret);

        // SHA256签名
        // paramMap.put("signatureMethod", "SHA256");
        // String sign = LiveSignUtil.getSHA256Sign(paramMap, appSecret);

        log.debug("生成签名：{}", sign);

    }

}
```