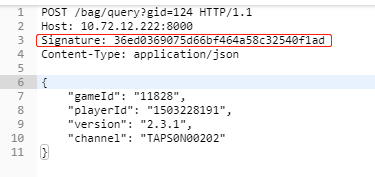
**海豚标准API签名规则说明**

**签名说明**

**1、基础知识**

1、所有请求都必须校验签名

2、签名通过请求头 Signature 进行校验



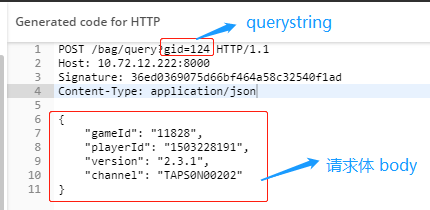
3、Signature 通过请求参数加上 appSecrect 进行计算获得

4、每个游戏都有自己的 appSecrect，appSecrect在海豚后台配置

总结：即需要在海豚配置游戏的 appSecrect 并在每个请求的请求头中带上计算好的 Signature 请求头

**2、签名规则**

0、不区分请求方法， querystring 以及 请求体 body 均需要参与签名（没有则不用）



1、对所有参与签名的参数按参数名做字典序升序排列（所谓字典序升序排列，直观上就如同在字典中排列单词一样排序，按照字母表或数字表里递增顺序的排列次序，即先考虑第一个“字母”，在相同的情况下考虑第二个“字母”，依此类推），key排序后按照 “参数名称”=“参数值”的形式拼接字符串，得到**签名字符串**。

1. Key1=value1&Key2=value2&Key3=value3&…………

2、将秘钥 appSecrect 拼接在**签名字符串之后**

1. Key1=value1&Key2=value2&Key3=value3&………**&appSecrect**

**对于请求头Content-Type=application/json时，请求体整体的key固定为“requestBody”,**  **value为请求的整个body。“requestBody”参与参数名排序。**

例如：

1. gid=124&requestBody={
2. "gameId": "11828",
3. "playerId": "1503228191",
4. "version": "2.3.1",
5. "channel": "TAPS0N00202"
6. }&8e134a

其中8e134a 是游戏 11828 的秘钥appSecrect

requestBody 是post上来的JSON串，当成一个整体，**里面的内容不用排序**。

3、md5生成签名串:

1. Signature=md5(Key1=value1&Key2=value2&Key3=value3&………&appSecrect)

**3、签名示例**

**postman**

直接复制这段js脚本到Pre-request Script中，发起的请求会自动带上签名

1. // 首先： 复制这端json 到Headers粘贴[{"key":"Signature","value":"{{resty-sign}}","enabled":true}]
2. // 然后： 修改 secret 为待接入游戏的 secret
4. var secret = '8e134a'
6. var params = {}
8. var type = request.headers['content-type']
9. if( type === 'application/json'){
10. params['requestBody'] = request.data
11. }else{
12. params = request.data
13. }
15. var members = pm.request.url.query.members
16. for (var i in members) {
17. var m = members[i]
18. params[m.key] = m.value
19. }
21. var keys = []
22. for (var k in params){
23. keys.push(k)
24. }
26. keys.sort()
27. var paramStr = "";
28. for(var i in keys){
29. var key = keys[i];
30. paramStr += key+"="+params[key]+"&";
31. }
33. var toSignStr = paramStr+secret
34. // console.log(toSignStr)
36. var sign = CryptoJS.MD5(toSignStr).toString()
37. // console.log(sign)
39. pm.environment.set("resty-sign", sign);

**LUA签名**

这是线上resty-ab签名组装的源码

1. local secret = '8e134a'
3. -- 拼装签名参数
4. local to\_sign\_tab
6. local ct = ngx.var.content\_type or ''
7. if string.sub(ct, 1, 16) == 'application/json' then
8. -- json 串直接装进 requestBody字段
9. local request\_body = ngx.req.get\_body\_data()
10. to\_sign\_tab = table\_util.merge\_hash({[REQUERST\_BODY\_NAME] = request\_body}, uri\_args)
11. else
12. -- form格式默认就是全部参数
13. to\_sign\_tab = all\_args
14. end
16. -- key按字母排序
17. local keys = new\_table(nkeys(to\_sign\_tab), 0)
18. for key, \_ in pairs(to\_sign\_tab) do
19. table.insert(keys, key)
20. end
21. table.sort(keys)
23. -- 拼装签名串
24. -- 为了性能，避免多余字符串gc，使用table组装
25. -- 格式 key = value & secret
26. local arr = new\_table(4 \* nkeys(to\_sign\_tab) + 1, 0)
27. local arr\_idx = 1
29. for \_, key in ipairs(keys) do
30. local value = to\_sign\_tab[key]
31. if type(value) == 'table' then
32. arr[arr\_idx] = key
33. arr[arr\_idx + 1] = '='
34. arr[arr\_idx + 2] = cjson\_safe.encode(value) -- 表单数据类型是数组时要编成json后签名
35. arr[arr\_idx + 3] = '&'
36. arr\_idx = arr\_idx + 4
37. else
38. arr[arr\_idx] = key
39. arr[arr\_idx + 1] = '='
40. arr[arr\_idx + 2] = value
41. arr[arr\_idx + 3] = '&'
42. arr\_idx = arr\_idx + 4
43. end
44. end
45. arr[arr\_idx] = secret
47. local to\_sign\_str = table.concat(arr, '')
48. local real\_sign = str\_util.to\_md5\_hex(to\_sign\_str)