五矿数据回传接口文档清单

文件修订历史

版本	修订时间	修订说明	作者	审核
1.0	2023 年 8 月 31 日	外部对接接口说明		
2.0	2023 年 9 月 7 日	入参以及例子说明		
3.0	2023 年 9 月 13 日	2.6/2.7 接口支持组织 id		
3.10	2023 年 9 月 20 日	2.6 接口返回字段增加 totalPerson(应参训总人数)字段		
3.11	2023 年 9 月 21 日	2.4/2.5 接口增加 orgId,talentCode 入参 2.6/2.7 接口入参增加 trainingTypes(培训班分类)		

1 环境信息

1、 测试环境主机地址:

open 开放平台	https://open-admin-stg3.zhi-niao.com
enterpriseId	744BFF8A27394555BE92B5615C285F95

2、生产环境主机地址:

open 开放平台	https:// open-api.zhi-niao.com
enterpriseId	D8E88B17C65B494B96AE9C133BCCBE25

3、 知鸟侧配合在开放平台页面配置厂商对接信息：
服务器 ip 网段:该配置信息在调用 getNewCode 时会获取 ip 地址与配置的 ip 对应校验
Appid 对应 agentId(代理商 id 下面接口会用到),
Appkey 对应 secret(代理商密钥下面接口会用到)
Salt 盐值(代理商用于签名验证)
配置页面如下图

测试环境开放平台页面配置厂商信息：

agentId	4BBC7B13C92A4B76BF3E6EFB3F557BD1
secret	5E8A26D76CD783E91E6BEE3A950ED434
salt	865AB1C31F384FEC94D775E3DDCA6F42

生产环境开放平台页面配置厂商信息：

agentId	2CD5AB2C805944548849E0C8F22CD984
secret	3AC41FC841FE5AB150274E1142BBC534
salt	1F2D05BC569C48D5AAD690D594D32A1E

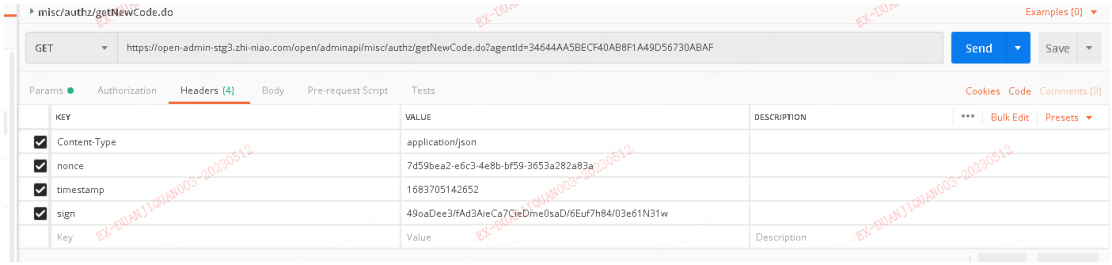
2 接口说明

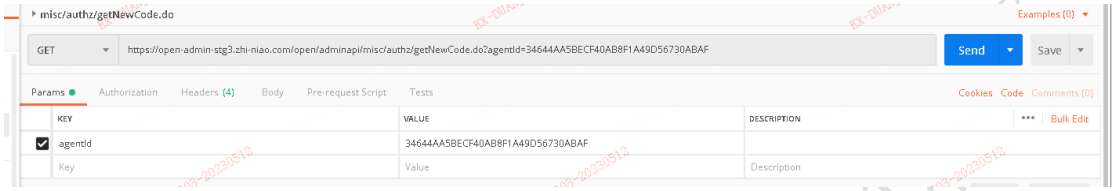
外部三方在调用知鸟的业务接口前需要完成三次握手来完成身份验证

2.1 获取 getNewCode 接口

在项目启动时就要调用我方 getNewCode 接口,建议使用 ApplicationInitRunner 或者定时任务来完成

2.1.1 服务基本信息





服务名称	获取 code 完成调用方第一次握手
url	{ open 开放平台}/open/adminapi/misc/authz/getNewCode.do 域名参考上述开放平台环境信息
技术协议	HTTPS
数据格式	JSON
接口描述	1、 厂商请求头参数 nonce ,timestamp 防重放攻击 2、 地址栏或者表单参数 agentId 厂商身份 3、 签名参数 sign 用于参数签名保证接口调用的合法性 4、 参数签名策略：参考附件代码 5、salt 为知鸟开放平台页面配置的盐值签名时会用到
请求方式	GET
返回参数	验证通过后返回 code 值

请求响应	{
	"body": "ATH167807190001848772309",
	"code": 200,
	"error": 0,
	"message": "success",
	"url": ""
	}

2.1.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
sign	String	是	接口 url+（head 参数+body 参数的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String		application/json

报文头 params			
参数名	类型	必选	说明
agentId	String	是	厂商身份厂商要在知鸟页面配置

参数签名示例代码:

```
String nonce = UUID.randomUUID().toString();
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数签名
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
String headerStr = SignUtils.sortParams(headerMap);
//  地址栏或者表单参数
Map<String, String> paramsMap = Maps.newHashMap();
paramsMap.put("agentId", agentId);
String paramsStr = SignUtils.sortParams(paramsMap);
//全部参数签名加 md5
String signMd5Salt = SignUtils.signMd5Salt(actionUrl + "?" + headerStr + "&" + paramsStr, salt);
String requestUrl = new StringBuilder(hostUrl).append(actionUrl).append("?agentId=").append(agentId).toString();

//远程调用接口示例代码：
StringBuilder absoluteUrl = new StringBuilder(requestUrl);
//设置请求头
HttpGet httpGet = new HttpGet(absoluteUrl.toString());
httpGet.setHeader(CONTENT_TYPE, "application/json");
```

```
httpGet.setHeader(NONCE, nonce);
httpGet.setHeader(TIMESTAMP, timestamp);
httpGet.setHeader(SIGN, signMd5Salt);

//创建远程调用对象
CloseableHttpClient httpclient = HttpClients.createDefault();
try {
    httpclient = HttpClientRemoteUtils.createSSLInsecureClient();
    CloseableHttpResponse httpResp = httpclient.execute(httpGet);
    log.info(" ApplicationInitRunner httpResp:{}", JSON.toJSONString(httpResp));
    if (Objects.nonNull(httpResp)) {
        HttpEntity entity = httpResp.getEntity();
        responseResult = EntityUtils.toString(entity, "utf8");
        log.info("ApplicationInitRunner responseResult:{}", JSON.toJSONString(responseResult));
    }
    return responseResult;
```

工具类 SignUtils 如附件

2.1.3 服务输出模型（请求回复报文）

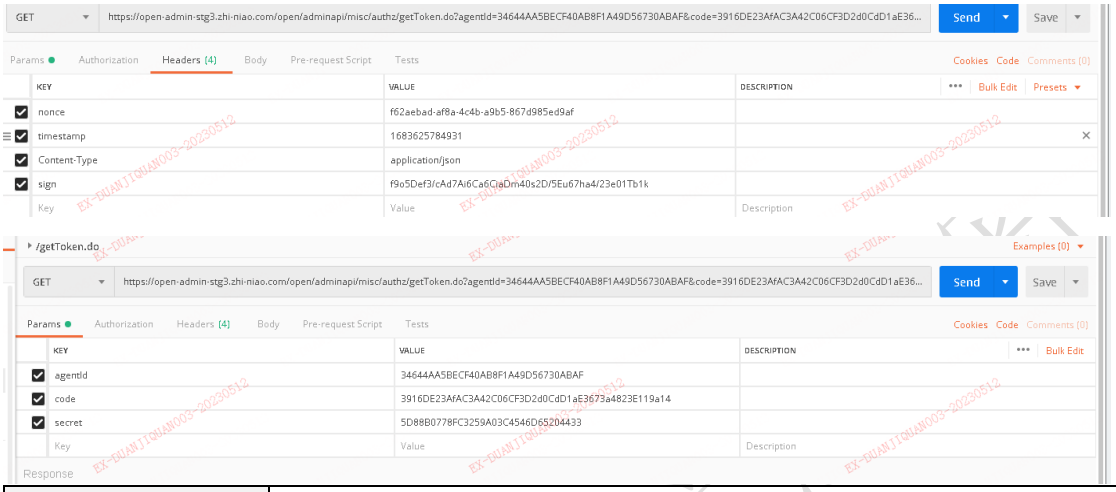
响应报文样例

```
{
    "body": "ATH167807190001848772309",
    "code": 200,
    "error": 0,
    "message": "success",
    "url": ""
}
```

参数名	类型	说明
code	String	返回码，200-成功，其他-失败
message	String	返回信息提示
body	String	返回体

2.2 获取开放平台 token 接口

2.2.1 服务基本信息



服务名称	厂商获取 token 接口
url	{ open 开放平台}/open /adminapi/misc/authz/getToken.do 域名参考上述开放平台环境信息
接口描述	1.厂商请求头参数:nonce,timestamp, 2.地址栏或者表单参数,agentId,agentSecret,code,厂商身份参数 3.签名参数 sign,用于参数签名保证接口调用的合法性 4.参数签名策略：参考代码 5.验证通过后返回 token, refreshToken, expiredTime 6.用于签名的 salt 为知鸟开放平台页面配置的盐值
技术协议	HTTPS
数据格式	JSON
接口类型	开发厂商接口
请求方式	GET

请求响应	{
	"code": "200",
	"message": "ok",
	"body": {
	"token": "xxxxxxx",
	"refreshToken": "xxxxxxx",
	“expired_times”:”xxxxxx”
	}
	}

2.2.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
sign	String	是	接口 url+（head 参数+body 参数的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String		application/json

报文头 params			
参数名	类型	必选	说明
agentId	String	是	厂商身份厂商要在知鸟页面配置
secret	String	是	厂商密钥
code	String	是	知鸟会 redis 缓存 code,1 分钟超时

备注：code 用于厂商登入时反向验证，保证请求来自于厂商服务器，而非某个攻击者用于完成第二次握手

入参参数代码示例：

```
String nonce = UUID.randomUUID().toString();
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数签名
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
String headerStr = SignUtils.sortParams(headerMap);
// 地址栏或者表单参数
Map<String, String> paramsMap = Maps.newHashMap();
paramsMap.put("agentId", params.get("agentId"));
paramsMap.put("secret", params.get("secret"));
paramsMap.put("code", params.get("code"));
String paramsStr = SignUtils.sortParams(paramsMap);
```

```
//全部参数签名加md5
String signMd5Salt = SignUtils.signMd5Salt(params.get("actionUrl") + "?" + headerStr + "&" + paramsStr, salt);
//远程调用
String response = HttpClientRemoteUtils.httpRemoteGetRequestToken(requestUrl, nonce, timestamp, signMd5Salt);


//远程调用时请求头设置,表单参数拼接在url后面 agentId、secret、code
//设置请求头
HttpGet httpGet = new HttpGet(absoluteUrl.toString());
httpGet.setHeader(CONTENT_TYPE, "application/json");
httpGet.setHeader(NONCE, nonce);
httpGet.setHeader(TIMESTAMP, timestamp);
httpGet.setHeader(SIGN, signMd5Salt);

//创建远程调用对象
CloseableHttpClient httpclient = HttpClients.createDefault();
```

2.2.3 服务输出模型（请求回复报文）

响应报文样例

```
{
    "code": "200",
    "message": "ok",
    "body": {
        "token": "xxxxxxx",
        "refreshToken": "xxxxxxx",
        “expired_times”:”xxxxxx”
    }
}
```

响应参数

参数名	类型	说明
Code	String	返回码，200-成功，其他-失败
message	String	返回信息
body		返回结果

数据结构 body 对象

参数名	类型	说明
token	String	获取的 token 令牌
expiredTime	String	token 超时时间，初始值为 2 小时
refreshToken	String	再次刷新 token 时用，后续接口使用

2.3 获取新 token 的接口(通常在调用业务接口设置 token 时做补偿措施时使用)

2.3.1 服务基本信息

服务名称	获取新 token，用于 token 过期时补偿
url	{ open 开放平台}/open/adminapi/misc/authz/refreshToken.do 域名参考上述开发平台环境信息
接口描述	1、 调用每个业务接口时必须有验签，验签中须有 token,当 token 失效时重新调 refreshToken.do 接口获取新的 token,或者手动调 refreshToken.do 接口重置 token 2、 厂商请求头参数:nonce,timestamp, 3、 地址栏或者表单参数,agentId ,refreshToken,厂商身份参数 4、 签名参数 sign,用于参数签名保证接口调用的合法性 5、 参数签名策略：参考代码 6、验证通过后返回 token, refreshToken, expiredTime 用于签名的 salt 为知鸟开放平台页面配置的盐值
技术协议	HTTPS
数据格式	JSON
接口类型	开发厂商接口
请求方式	GET

请求响应	{
	"code": "200",
	"message": "ok",
	"body": {
	"token": "xxxxxxx",
	"refreshToken": "xxxxxxx",
	“expired_times”:”xxxxxx”
	}
	}

2.3.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
sign	String	是	接口 url+（head 参数+body 参数的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String		application/json

报文头 params			
参数名	类型	必选	说明
agentId	String	是	厂商身份厂商要在知鸟页面配置
refreshToken	String	是	上一个接口返回的认证信息

入参设置代码示例：

```
String nonce = UUID.randomUUID().toString();
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数排序
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
String headerStr = SignUtils.sortParams(headerMap);
// 地址栏或者表单参数排序
String refreshTokenRedisKey = MessageFormat
        .format(RedisKeyConstant.MANUFACTURER_LOGIN_REFRESH_TOKEN_KEY, agentId);
String oldRefreshToken = redisTemplate.opsForValue().get(refreshTokenRedisKey);
Map<String, String> paramsMap = Maps.newHashMap();
paramsMap.put("agentId", agentId);
paramsMap.put("refreshToken", oldRefreshToken);
String paramsStr = SignUtils.sortParams(paramsMap);
```

```
//全部参数签名加md5
String signMd5Salt = SignUtils.signMd5Salt(actionUrl + "?" + headerStr + "&" + paramsStr, apiConstant.getSalt());
log.info(actionUrl + "?" + headerStr + "&" + paramsStr, apiConstant.getSalt());
//拼接get请求参数
String requestFullUrl = new StringBuilder(hostUrl).append(actionUrl)
        .append("?agentId=").append(agentId)
        .append("&refreshToken=").append(oldRefreshToken).toString();
//远程调用
String response = HttpClientRemoteUtils
        .httpRemoteGetRequestToken(requestFullUrl, nonce, timestamp, signMd5Salt);

//远程调用请求头设置示例,表单参数拼接在url后面 agentId、refreshToken
//设置请求头
HttpGet httpGet = new HttpGet(absoluteUrl.toString());
httpGet.setHeader(CONTENT_TYPE, "application/json");
httpGet.setHeader(NONCE, nonce);
httpGet.setHeader(TIMESTAMP, timestamp);
httpGet.setHeader(SIGN, signMd5Salt);

//创建远程调用对象
CloseableHttpClient httpclient = HttpClients.createDefault();
```

2.3.3 服务输出模型（请求回复报文）

响应报文样例

```
{
    "code": "200",
    "message": "ok",
    "body": {
        "token": "xxxxxxx",
        "refreshToken": "xxxxxxx",
        “expired_times”:”xxxxxx”
    }
}
```

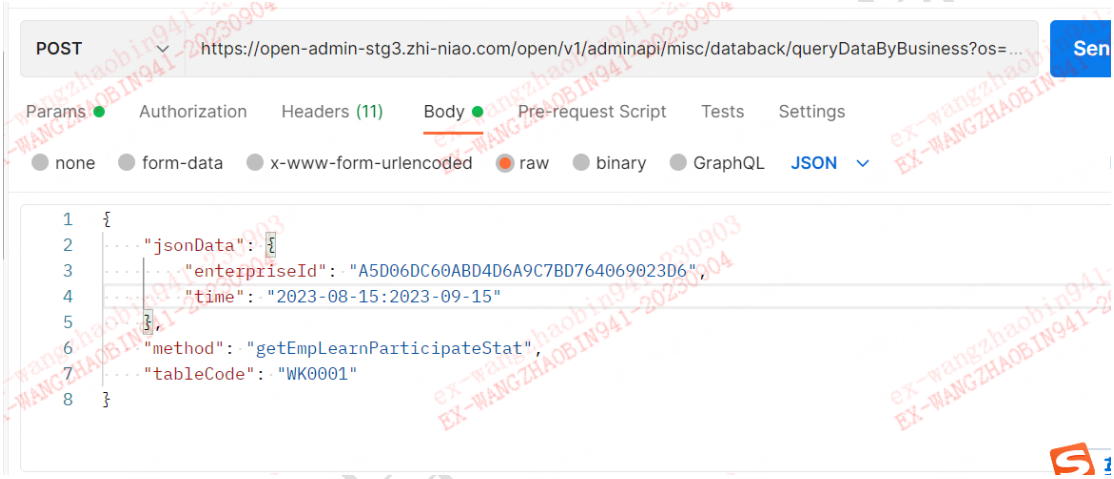
响应参数

参数名	类型	说明
Code	String	返回码，200-成功，其他-失败
message	String	返回信息
body		返回结果

数据结构 body 对象

参数名	类型	说明
token	String	获取的 token 令牌
expiredTime	String	token 超时时间，初始值为 2 小时
refreshToken	String	再次刷新 token 时用

2.4《员工参训情况》指标数据回传接口定制



2.4.1 服务基本信息

服务名称	【五矿数据回传】《贯彻落实党和国家重大决策部署的集中轮训情况-汇总表》指标数据回传接口
url	{open 开放平台}/open/v1/adminapi/misc/databack/queryDataByBusiness 域名参考上述开放平台环境信息
接口描述	1.厂商请求头参数:nonce,timestamp, 2.签名参数 sign,用于参数签名保证接口调用的合法性 3.参数签名策略：参考代码 4.用于签名的 salt 为知鸟开放平台页面配置的盐值
技术协议	HTTPS
数据格式	JSON
接口类型	开发厂商接口
请求方式	POST

2.4.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
token	String	是	厂商 auth 登陆生成的 token
sign	String	是	接口 url+（head 参数+body 参数的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String	是	application/json

请求体 body			
参数名	类型	必选	说明
jsonData	String	是	查询条件组成的字符串（Json 字符串）
method	String	是	方法名称(固定值) getEmpLearnParticipateStat
tableCode	String	是	编码（固定值）： WK0001

jsonData 组成如下：

参数名	类型	必选	说明
enterpriseId	String	是	企业 ID
time	String	是	时间 yyy-MM-dd:yyyy-MM-dd 最大跨度 366 天 eg.2022-01-01:2022-12-31
orgId	String	否	组织 id
talentCode	String	否	人才分类编码 (多个以英文逗号隔开)

入参设置代码示例：

String nonce = UUID.randomUUID().toString();

```
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数排序
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
String headerStr = SignUtils.sortParams(headerMap);
String actionUrl ="/open/v1/adminapi/misc/databack/queryDataByBusiness";
// 验签 sign 参数
String signMd5Salt = SignUtils.signMd5Salt(actionUrl + "?" + headerStr + "&", salt);

//post 远程调用请求参数设置示例
CloseableHttpClient httpclient = HttpClients.createDefault();
HttpPost postRequest = new HttpPost(fullUrl);
//设置请求头
postRequest.setHeader("nonce", userInfo.getNonce());
postRequest.setHeader("timestamp", userInfo.getTimestamp());
postRequest.setHeader("sign", userInfo.getSign());
postRequest.setHeader("token", userInfo.getToken());
String result = "";
try {
    HttpEntity paramsEntity = new StringEntity(paramsBody,
            ContentType.create(MediaType.APPLICATION_JSON_VALUE, UTF8));
    postRequest.setEntity(paramsEntity);

    Builder customReqConf = RequestConfig.custom();
    customReqConf.setConnectTimeout(CONN_TIMEOUT);
    customReqConf.setSocketTimeout(READ_TIMEOUT);
    postRequest.setConfig(customReqConf.build());
    if (fullUrl.startsWith("https")) {
        //执行 https 请求
        httpclient = createSSLInsecureClient();
    }
    //执行请求
    CloseableHttpResponse responseResult = httpclient.execute(postRequest);
    HttpEntity respEntity = responseResult.getEntity();
    result = EntityUtils.toString(respEntity, UTF8);
```

2.4.3 服务输出模型（请求回复报文）

响应报文样例

```
{
```

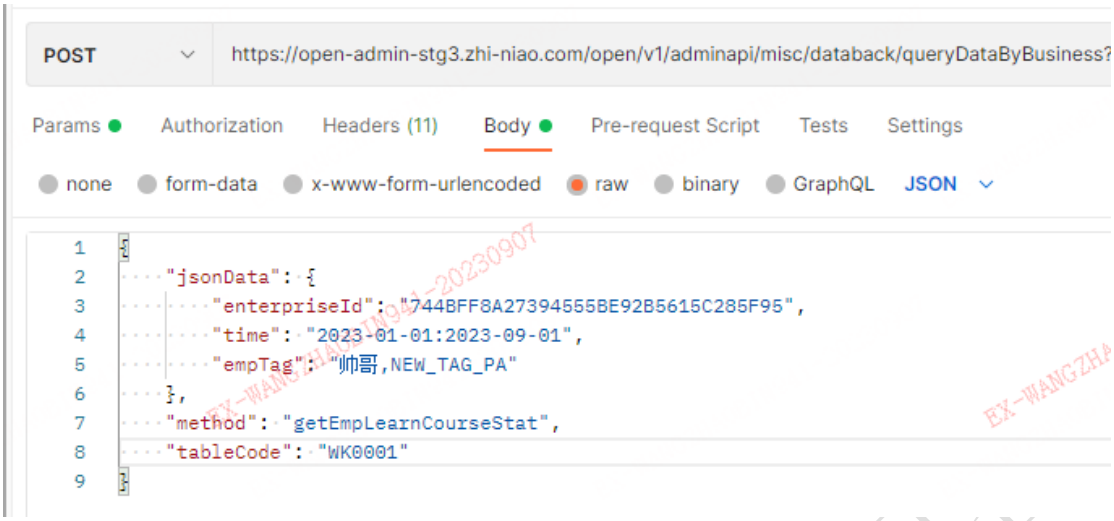
```
        "body": {
            " empAttendRate": "0.04%",
            " platNum": 0,
            " studyDurAvg": "63.25",
            " trainCnt": 1
          },
        "code": 200,
        "error": 0,
        "message": "success",
        "url": ""
}
```


响应参数

数据结构 body 对象

参数名	类型	说明
empAttendRate	String	员工参训率
platNum	Long	平台总人数
studyDurAvg	String	人均学时
trainCnt	Long	企业培训总人次

2.5《员工学课情况》指标数据回传接口定制



2.5.1 服务基本信息

服务名称	【五矿数据回传】《贯彻落实党和国家重大决策部署的集中轮训情况-汇总表》数据回传接口定制
url	{ open 开放平台}/open/v1/adminapi/misc/databack/queryDataByBusiness 域名参考上述开放平台环境信息
接口描述	1.厂商请求头参数:nonce,timestamp, 2.签名参数 sign,用于参数签名保证接口调用的合法性 3.参数签名策略：参考代码 4.用于签名的 salt 为知鸟开放平台页面配置的盐值
技术协议	HTTPS
数据格式	JSON
接口类型	开发厂商接口
请求方式	POST

2.5.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
token	String	是	厂商 auth 登陆生成的 token
sign	String	是	接口 url+（head 参数+body 参数

			的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String		application/json

请求体 body			
参数名	类型	必选	说明
jsonData	String	是	查询条件组成的字符串（JSON 格式）
method	String	是	方法名称（固定值）： getEmpLearnCourseStat
tableCode	String	是	编码（固定值）： WK0001

jsonData 组成如下：

参数名	类型	必选	说明
enterpriseId	String	是	企业 ID
empTag	String	是	员工标签(多个标签用 ，隔开)
time	String	是	时间 yyy-MM-dd:yyyy-MM-dd 最大跨度 366 天 eg.2022-01-01:2022-12-31
orgId	String	否	组织 id
talentCode	String	否	人才分类编码 (多个以英文逗号隔开)

入参设置代码示例：

```
String nonce = UUID.randomUUID().toString();
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数排序
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
String headerStr = SignUtils.sortParams(headerMap);
String actionUrl = "/open/v1/adminapi/misc/databack/queryDataByBusiness";
// 验签 sign 参数
String signMd5Salt = SignUtils.signMd5Salt(actionUrl + "?" + headerStr + "&", salt);

//post 远程调用请求参数设置示例
CloseableHttpClient httpclient = HttpClients.createDefault();
HttpPost postRequest = new HttpPost(fullUrl);
```

```
//设置请求头
postRequest.setHeader("nonce", userInfo.getNonce());
postRequest.setHeader("timestamp", userInfo.getTimestamp());
postRequest.setHeader("sign", userInfo.getSign());
postRequest.setHeader("token", userInfo.getToken());
String result = "";
try {
    HttpEntity paramsEntity = new StringEntity(paramsBody,
            ContentType.create(MediaType.APPLICATION_JSON_VALUE, UTF8));
    postRequest.setEntity(paramsEntity);

    Builder customReqConf = RequestConfig.custom();
    customReqConf.setConnectTimeout(CONN_TIMEOUT);
    customReqConf.setSocketTimeout(READ_TIMEOUT);
    postRequest.setConfig(customReqConf.build());
    if (fullUrl.startsWith("https")) {
        //执行 https 请求
        httpclient = createSSLInsecureClient();
    }
    //执行请求
    CloseableHttpResponse responseResult = httpclient.execute(postRequest);
    HttpEntity respEntity = responseResult.getEntity();
    result = EntityUtils.toString(respEntity, UTF8);
```

2.5.3 服务输出模型（请求回复报文）

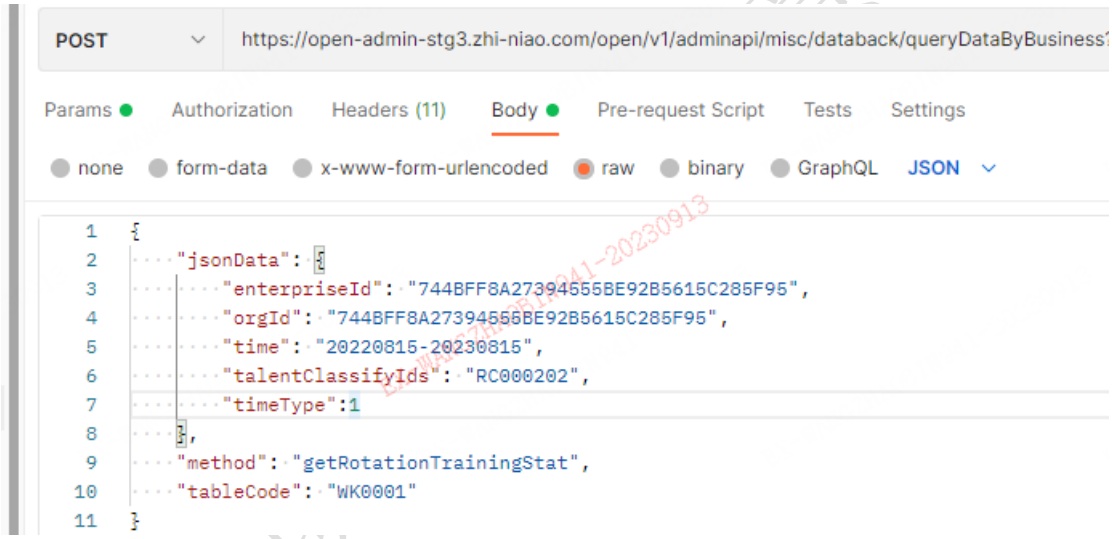
响应报文样例

```
{
   "body":  {
    " studyCoverRate ": "6.06%",
    " studyDurationAvg ": "0.26",
    " studyCnt ": 300
   },
   "code": 200,
   "error": 0,
   "message": "success",
   "url": ""
}
```

响应参数

参数名	类型	说明
studyCoverRate;	String	学习覆盖率
studyDurationAvg	String	人均学习时长
studyCnt	Long	学习总人次

2.6【五矿数据回传】《贯彻落实党和国家重大决策部署的集中轮训情况-汇总表》指标数据回传接口



2.6.1 服务基本信息

服务名称	【五矿数据回传】培训班基础数据同步接口
url	{ open 开放平台} /open/v1/adminapi/misc/databack/queryDataByBusiness 域名参考上述开放平台环境信息
接口描述	1.厂商请求头参数:nonce,timestamp, 2.签名参数 sign,用于参数签名保证接口调用的合法性 3.参数签名策略：参考代码 4.用于签名的 salt 为知鸟开放平台页面配置的盐值
技术协议	HTTPS
数据格式	JSON
接口类型	开发厂商接口
请求方式	POST

2.6.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
token	String	是	厂商 auth 登陆生成的 token
sign	String	是	接口 url+（head 参数+body 参数的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String		application/json

请求体 body			
参数名	类型	必选	说明
jsonData	String	是	查询条件组成的字符串（JSON 格式）
method	String	是	方法名称（固定值）：getRotationTrainingStat
tableCode	String	是	编码（固定值）：WK0001

jsonData 组成如下：

参数名	类型	必选	说明
enterpriseId	String	是	企业 Id
orgId	String	否	组织 Id
talentClassifyIds	String	是	人才分类 Id,多个用英文逗号分隔
timeType	Integer	是	统计时间类型（1=按日、2=按月、3=按季、4=按年、5=累计）
time	String		统计时间：根据上述时间类型，若传 1-4，则需要填写时间区间，格式年月日-年月日，即

			yyyyMMdd-yyyyMMdd。若传 5，则不需要填写时间区间
trainingTypes	String	否	培训班分类，多个用英文逗号分隔

入参设置代码示例：

```
String nonce = UUID.randomUUID().toString();
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数排序
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
String headerStr = SignUtils.sortParams(headerMap);
String actionUrl = "/open/v1/adminapi/misc/databack/queryDataByBusiness";
// 验签 sign 参数
String signMd5Salt = SignUtils.signMd5Salt(actionUrl + "?" + headerStr + "&", salt);

//post 远程调用请求参数设置示例
CloseableHttpClient httpclient = HttpClients.createDefault();
HttpPost postRequest = new HttpPost(fullUrl);
//设置请求头
postRequest.setHeader("nonce", userInfo.getNonce());
postRequest.setHeader("timestamp", userInfo.getTimestamp());
postRequest.setHeader("sign", userInfo.getSign());
postRequest.setHeader("token", userInfo.getToken());
String result = "";
try {
    HttpEntity paramsEntity = new StringEntity(paramsBody,
            ContentType.create(MediaType.APPLICATION_JSON_VALUE, UTF8));
    postRequest.setEntity(paramsEntity);

    Builder customReqConf = RequestConfig.custom();
    customReqConf.setConnectTimeout(CONN_TIMEOUT);
    customReqConf.setSocketTimeout(READ_TIMEOUT);
    postRequest.setConfig(customReqConf.build());
    if (fullUrl.startsWith("https")) {
        //执行 https 请求
        httpclient = createSSLInsecureClient();
    }
    //执行请求
    CloseableHttpResponse responseResult = httpclient.execute(postRequest);
    HttpEntity respEntity = responseResult.getEntity();
```

result = EntityUtils.*toString*(respEntity, *UTF8*);

2.6.3 服务输出模型（请求回复报文）

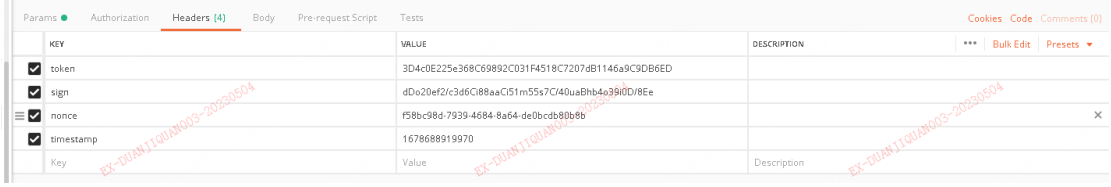
响应报文样例

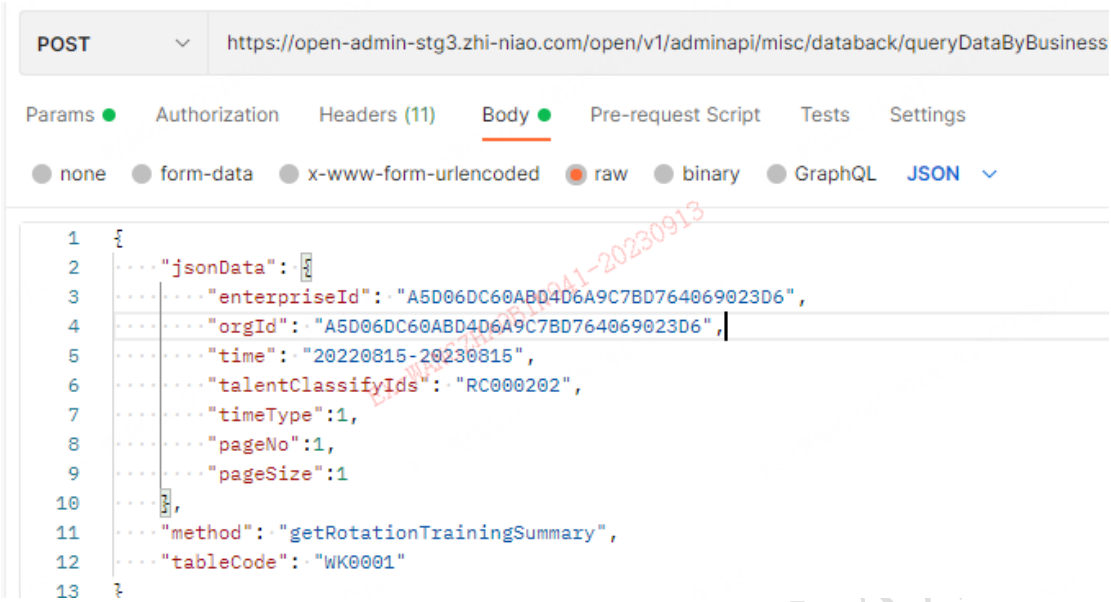
```
{
    "body":  {
       "perCapitaRotationalHours": "23"
       "rotationCoverageRate": "23%",
       "nonParticipantsNum": "23",
       "totalPerson": 2
    },
    "code": 200,
     "error": 0,
     "message": "success",
     "url": ""
}
```

响应参数

参数名	类型	说明
perCapitaRotationalHours	String	人均轮训学时　　无数据时返回　"-"
rotationCoverageRate	String	轮训覆盖率　　无数据时返回　"-"
nonParticipantsNum	String	未参训人数　　无数据时返回　"-"
totalPerson	Integer	应参训总人数

2.7【五矿数据回传】《贯彻落实党和国家重大决策部署的集中轮训情况-汇总表》数据回传接口定制





2.7.1 服务基本信息

服务名称	【五矿数据回传】员工参训情况数据接口
url	{ open 开放平台}/open/v1/adminapi/misc/databack/queryDataByBusiness 域名参考上述开放平台环境信息
接口描述	1.厂商请求头参数:nonce,timestamp, 2.签名参数 sign,用于参数签名保证接口调用的合法性 3.参数签名策略：参考代码 4.用于签名的 salt 为知鸟开放平台页面配置的盐值
技术协议	HTTPS
数据格式	JSON
接口类型	开发厂商接口
请求方式	POST

2.7.2 服务输入参数

请求参数

报文头 hearders			
参数名	类型	必选	说明
nonce	String	是	每次请求生成的随机值,防重放攻击
timestamp	String	是	时间戳，控制调用频率
token	String	是	厂商 auth 登陆生成的 token
sign	String	是	接口 url+（head 参数+body 参数

			的排序串：如/openapi/XX?aa=bb&token=xx），使用企业配置的签名摘要盐值进行摘要算法
Content-Type	String		application/json

请求体 body			
参数名	类型	必选	说明
jsonData	String	是	查询条件组成的字符串
method	String	是	方法名称（固定值）：getRotationTrainingSummary
tableCode	String	是	编码（固定值）：WK0001

jsonData 组成字符串如下

参数名	类型	必选	说明
enterpriseId	String	是	企业 Id
orgId	String	否	组织 Id
talentClassifyIds	String	是	人才分类 Id,多个用英文逗号分隔
timeType	Integer	是	统计时间类型（1=按日、2=按月、3=按季、4=按年、5=累计）
time	String		统计时间：根据上述时间类型，若传 1-4，则需要填写时间区间，格式年月日-年月日，即 yyyyMMdd-yyyyMMdd。若传 5，则不需要填写时间区间
trainingTypes	String	否	培训班分类,多个用英文逗号分隔
pageNo	Integer	是	页码
pageSize	Integer	是	页大小

入参设置代码示例：

```
String nonce = UUID.randomUUID().toString();
String timestamp = System.currentTimeMillis() + Strings.EMPTY;
//请求头参数排序
Map<String, String> headerMap = Maps.newHashMap();
headerMap.put("nonce", nonce);
headerMap.put("timestamp", timestamp);
```

```
String headerStr = SignUtils.sortParams(headerMap);
String actionUrl = "/open/v1/adminapi/misc/databack/queryDataByBusiness";
// 验签 sign 参数
String signMd5Salt = SignUtils.signMd5Salt(actionUrl + "?" + headerStr + "&", salt);

//post 远程调用请求参数设置示例
CloseableHttpClient httpclient = HttpClients.createDefault();
HttpPost postRequest = new HttpPost(fullUrl);
//设置请求头
postRequest.setHeader("nonce", userInfo.getNonce());
postRequest.setHeader("timestamp", userInfo.getTimestamp());
postRequest.setHeader("sign", userInfo.getSign());
postRequest.setHeader("token", userInfo.getToken());
String result = "";
try {
    HttpEntity paramsEntity = new StringEntity(paramsBody,
            ContentType.create(MediaType.APPLICATION_JSON_VALUE, UTF8));
    postRequest.setEntity(paramsEntity);

    Builder customReqConf = RequestConfig.custom();
    customReqConf.setConnectTimeout(CONN_TIMEOUT);
    customReqConf.setSocketTimeout(READ_TIMEOUT);
    postRequest.setConfig(customReqConf.build());
    if (fullUrl.startsWith("https")) {
        //执行 https 请求
        httpclient = createSSLInsecureClient();
    }
    //执行请求
    CloseableHttpResponse responseResult = httpclient.execute(postRequest);
    HttpEntity respEntity = responseResult.getEntity();
    result = EntityUtils.toString(respEntity, UTF8);
```

2.7.3 服务输出模型（请求回复报文）

响应报文样例

```
{
  "body": {
        "data": [
          {
            "orderNo": 0,
            "empId": "",
            "empName": "",
            "orgFullPathCn": "",
```

```
            "talentClassifyFullPath": null,
            "talentClassifyId": null,
            "orgId": "",
            "trainingNum": "",
            "trainingTotalHours": "",
            "actualTotalHours": ""
          }
        ],
          "pageNo": 1,
          "pageSize": 10,
          "total": 0
    },
      "code": 200,
      "error": 0,
      "message": "success",
      "url": ""
}
```


响应参数

参数名	类型	说明
data		返回结果

数据结构 data 对象

参数名	类型	说明
orderNo	Integer	序号
empId	String	员工编号
empName	String	员工名称
orgFullPathCn	String	所属组织全路径
talentClassifyFullPath	String	人才分类全路径(中文)
talentClassifyId	String	人才分类 ID
orgId	String	所属组织 ID
trainingNum	String	培训班总数
trainingTotalHours	String	培训班总学时(h)，总学时取累加数。保留两位小数
actualTotalHours	String	实际获得总学时(h),实际获得总学时取累加数。保留两位小数
分页参数名	类型	说明
pageNo	Integer	页码
pageSize	Integer	页大小
total	Long	总条数

3 工具类：

3.1 加密工具类：Md5Utils

```
package com.util;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import org.apache.commons.lang3.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

/**
 * 生成MD5码工具类
 */
public class Md5Utils {

    private static final Log log = LogFactory.getLog(Md5Utils.class);
    private static final String MD5 = "MD5";
    public static final String UTF8 = "UTF-8";
    /***
     * 返回该字符串的md5码
     *
     * @return 返回该字符串的md5码
     */
    public static String getMd5(String str) {

        MessageDigest md5 = null;
        try {
            md5 = MessageDigest.getInstance(MD5);
        } catch (Exception e) {
            log.error("初始化MD5工具类错误e:" + e.getMessage());
        }

        StringBuilder sb = new StringBuilder(40);
        if (StringUtils.isNotEmpty(str) && md5 != null) {
            byte[] bs;
            try {
                bs = md5.digest(str.getBytes(UTF8));
            } catch (UnsupportedEncodingException e) {
                bs = md5.digest(str.getBytes());
            }
            for (byte x : bs) {
                if ((x & 0xff) >> 4 == 0) {
```

```
                    sb.append("0").append(Integer.toHexString(x & 0xff));
                } else {
                    sb.append(Integer.toHexString(x & 0xff));
                }
            }
        }
        return sb.toString();
    }

    /**
     * 返回md5偏移后的字符串
     * @param str
     * @param salt
     * @return java.lang.String
     */

    public static String getSaltMd5(String str, String salt) {
        String md5 = getMd5(str + salt);
        char[] cs = new char[48];
        for (int i = 0; i < 48; i += 3) {
            cs[i] = md5.charAt(i / 3 * 2);
            char c = salt.charAt(i / 3);
            cs[i + 1] = c;
            cs[i + 2] = str.charAt(i / 3 * 2 + 1);
        }
        return String.valueOf(cs);
    }
}
```

3.2 签名工具类：SignUtils

```
package com.synchronization.project.utils;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.google.common.collect.Maps;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import javax.servlet.http.HttpServletRequest;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
```

```
@Slf4j
public class SignUtils {
    public static String sortParams(Map<String, String> paramsMap) {
        List<Entry<String, String>> infoIds = new ArrayList<>(paramsMap.entrySet());
        infoIds.sort(Entry.comparingByKey());
        // 构造签名键值对的格式
        StringBuilder sb = new StringBuilder();
        for (Entry<String, String> item : infoIds) {
            if (StringUtils.isNotBlank(item.getKey())) {
                String key = item.getKey();
                String val = item.getValue();
                if (StringUtils.isNotBlank(val)) {
                    sb.append(key).append("=").append(val).append("&");
                }
            }
        }
        if (StringUtils.isNotBlank(sb)) {
            String str = sb.substring(0, sb.length() - 1);
            return validStringBuffer(str);
        }
        return StringUtils.EMPTY;
    }

    private static String validStringBuffer(String valueString) {
        if (StringUtils.isBlank(valueString)) {
            return valueString;
        }
        if (valueString.endsWith(",")) {
            return valueString.substring(0, valueString.length() - 1);
        }
        return valueString;
    }

    public static String signMd5Salt(String str, String salt) {
        return Md5Utils.getSaltMd5(str, salt);
    }

    /**
     * 表头参数获取
     *
     * @param request
     * @return
     */
```

```
public static String getSignHeaderNew(HttpServletRequest request) {
    // 验证签名是否正确
    // 1、header
    Map<String, String> headerMap = Maps.newHashMap();
    headerMap.put("nonce", request.getHeader("nonce"));
    headerMap.put("timestamp", request.getHeader("timestamp"));
    // 验签时不验证 token 信息
    // headerMap.put("token",request.getHeader("token"));
    headerMap.put("sid", request.getHeader("sid"));
    headerMap.put("authToken", request.getHeader("authToken"));
    log.info("getSignHeader headerMap:{},sid:{}", JSON.toJSONString(headerMap), request.getHeader("sid"));
    return SignUtils.sortParams(headerMap);
}

/**
 * 地址栏或表单签名参数
 *
 * @param request
 * @return
 */
public static String getSignParameterNew(HttpServletRequest request) {
    // 地址栏或者表单参数
    Map<String, String> paramMap = Maps.newHashMap();
    Map<String, String[]> requestParamsMap = request.getParameterMap();
    log.info("SignUtils getSignParameter:{}", JSON.toJSONString(requestParamsMap));
    if (requestParamsMap != null && !requestParamsMap.isEmpty()) {
        requestParamsMap.keySet().forEach(paramKey -> {
            String[] valueArray = requestParamsMap.get(paramKey);
            if (valueArray != null && valueArray.length > 0) {
                StringBuilder valueBuilder = new StringBuilder();
                for (String tmpValue : valueArray) {
                    valueBuilder.append(tmpValue).append(",");
                }
                if (valueBuilder.length() > 1) {
                    paramMap.put(paramKey, valueBuilder.substring(0, valueBuilder.length() - 1));
                } else {
                    paramMap.put(paramKey, "");
                }
            }
        });
    }
    return SignUtils.sortParams(paramMap);
}
```

```
/**
 * json 的 body 签名
 *
 * @param jsonObject
 * @return
 */
public static String getSignParameterJsonNew(JSONObject jsonObject) {
    // 地址栏或者表单参数
    Map<String, String> paramMap = Maps.newHashMap();
    jsonObject.keySet().forEach(jsonKey -> {
        if (!"sign".equals(jsonKey)) {
            if (jsonObject.get(jsonKey) instanceof JSONObject) {
                paramMap.put(jsonKey, getSignParameterJson((JSONObject) jsonObject.get(jsonKey)));
            } else if (jsonObject.get(jsonKey) instanceof JSONArray) {
                getJsonArray(jsonObject, paramMap, jsonKey);
            } else {
                paramMap.put(jsonKey, jsonObject.getString(jsonKey));
            }
        }
    });
    return "{" + SignUtils.sortParams(paramMap) + "}";
}

/**
 * json 的 body 签名
 */
public static String getSignParameterJson(JSONObject jsonObject) {
    // 地址栏或者表单参数
    Map<String, String> paramMap = Maps.newHashMap();
    jsonObject.keySet().forEach(jsonKey -> {
        if (!"sign".equals(jsonKey)) {
            if (jsonObject.get(jsonKey) instanceof JSONObject) {
                paramMap.put(jsonKey, getSignParameterJson((JSONObject) jsonObject.get(jsonKey)));
            } else if (jsonObject.get(jsonKey) instanceof JSONArray) {
                getJsonArray(jsonObject, paramMap, jsonKey);
            } else {
                paramMap.put(jsonKey, jsonObject.getString(jsonKey));
            }
        }
    });
    return "{" + SignUtils.sortParams(paramMap) + "}";
}
```

```
private static void getJsonArray(JSONObject jsonObject, Map<String, String> paramMap, String jsonKey) {
    JSONArray jsonArray = jsonObject.getJSONArray(jsonKey);
    StringBuilder builder = new StringBuilder("[");
    for (Object tmpObject : jsonArray) {
        if (tmpObject instanceof JSONObject) {
            builder.append(getSignParameterJson((JSONObject) tmpObject)).append(",");
        } else {
            builder.append(tmpObject.toString()).append(",");
        }
    }
    if (builder.length() > 1) {
        paramMap.put(jsonKey, builder.substring(0, builder.length() - 1) + "]");
    } else {
        paramMap.put(jsonKey, builder + "]");
    }
}
```