

操作文档

2025-02-09 16:26

Table of Contents

1、接口调用及参数说明

1.1 查看我的接口（API）

1.2 测试接口

1.3. 调用接口

1.3.1 request url

1.3.2 Header

1.3.3 Request Body

1.3.4 response

2. 分页查询

2.1 普通分页

2.2 游标分页

3 全量拉取

3.1、判断文件是否下载完整

4、举个例子

4.1、postman 导入下面文件

4.2、服务市场token

4.3 调用自己接口

5、加解密配置

5.1 dos加盐规则

5.2 dos特殊数据加解密处理方法

5.3. RSA加解密说明

5.4. java版本RSA分段加解密方法

6、idc调用OA

测试环境: <https://uat-hrdos.woa.com>

线上环境 : <https://hrdos.woa.com>

主数据通用接口 : [W 10 主数据公共接口服务清单\(归档\)](#)

内部服务注册: [W 系统和服务注册和修改](#)

外部bg应用注册 [W 应用注册](#)

1、接口调用及参数说明

1.1 查看我的接口 (API)

登录dos平台 (注意区分测试环境和线上环境) , 找到我的API点击→接口文档, 如下图所示 :

The screenshot shows the 'My APIs' page of the DOS platform. The left sidebar has tabs for 'My APIs' (highlighted with a red box), 'Data Source', 'Model Management', 'Interface Configuration', 'Interface Authorization', and 'Application Configuration'. The main area displays a table of 10 APIs. Each API row includes columns for 'Name', 'Application', 'Data Source', 'Creator', 'Summary', and 'Operations'. The 'Operations' column contains links for 'Copy Link' and 'API Document'. A red arrow points to the 'API Document' link for the first API entry.

接口名称	所属应用	数据源	创建人	摘要	操作
1 debug-column-code/my-daily-test	my-daily-test	pg-test-nebula	nebulaliao	sdfdsf	复制链接 接口文档
2 dos-test-uat/my-daily-test	my-daily-test	pg-test-nebula	nebulaliao	test	复制链接 接口文档
3 md-api-public-core-staff-email-realtime/v-zehhaowu-test	v-zehhaowu-test	hrmd-datasource-r-pg-md3-global	nebulaliao	员工工作邮箱	复制链接 接口文档
4 md-api-public-core-staff-info/my-daily-test	my-daily-test	ds-r-pg-hrmd-md3	nebulaliao	员工职务数据	复制链接 接口文档
5 test-auth-policy/my-daily-test	my-daily-test	daily-postgres	nebulaliao	2	复制链接 接口文档
6 mock1-lbiplus-staff-change-info/my-daily-test	my-daily-test	数据集迁移-mock-测试	nebulaliao	来自数据集的配置	复制链接 接口文档
7 cache-test/my-daily-test	my-daily-test	pg-test-nebula	nebulaliao	test	复制链接 接口文档
8 staff-auto-encryption/v-zehhaowu-test	v-zehhaowu-test	auto-test-encryption	v_azhizhang	自动化测试勿动	复制链接 接口文档
9 auto-test-outwhere/v-zehhaowu-test	v-zehhaowu-test	auto-test-outwhere	v_azhizhang	自动化测试勿动	复制链接 接口文档
10 auto-test3-v2/v-zehhaowu-test	v-zehhaowu-test	auto-test3-v2	v_azhizhang	自动化测试勿动	复制链接 接口文档

测试

授权信息

所属应用: dos-test-app(dos-test-app) **接口授权给了那个应用使用**

过滤范围: StaffId = 1 **交付人员配置的sql过滤条件, 已AND拼接到原sql的where 条件中**

老接口: -

状态: 使用中

授权理由: test

接口调用url, 不同环境使用不同的地址前缀

POST地址: /api/esb/dos-interface-server/open-api/config/my-daily-test/qhitest222/dos-test-app/data 地址前缀 ^

请求头: hrgw-appname: hrgw-signature: hr-timestamp: 参考 ^ **接口鉴权所需参数**

配置信息

数据源: daily-mysql **数据源头**

配置名称: qhitest222

接口摘要: test

分页方式: 普通分页 可见应用: tencent.hr.recruit.globalrecruit.nts-gateway, **后面会讲分页方式** tencent.hr.recruit.globalrecruit.cping, tencent.hr.recruit.globalrecruit.demo710, dos-test-app, dos-test-wu

用户limit缓存: **接口缓存, 开启及缓存时间设置**

调入参数 **接口调用时可以传入的参数**

参数名	参数重命名	参数类型	默认值	是否必填	操作	描述	备注
staffId		string		是	=		

授权字段 **接口授权了那些字段, 即接口调用返回的字段**

(全部 已订阅 未订阅)

是否订阅	字段名	返回字段名	字段类型	字段描述	备注	主键字段	是否加密	查询条件	是否转化为String
<input checked="" type="checkbox"/>	StaffID	StaffID	INT	StaffID	-	<input type="checkbox"/>	<input type="checkbox"/>	-	<input type="checkbox"/>
<input checked="" type="checkbox"/>	EngName	EngName	VARCHAR	EngName	-	<input type="checkbox"/>	<input type="checkbox"/>	-	<input type="checkbox"/>
<input checked="" type="checkbox"/>	ChnName	ChnName	VARCHAR	ChnName	-	<input type="checkbox"/>	<input type="checkbox"/>	-	<input type="checkbox"/>
<input checked="" type="checkbox"/>	FullName	FullName	VARCHAR	FullName	-	<input type="checkbox"/>	<input type="checkbox"/>	-	<input type="checkbox"/>

1. **分页方式**: 默认为游标分页, 请看下文2.1

2. **授权字段**: 具体授权了哪些字段, 这些字段的一些信息, 含义如下:

名称	含义
是否订阅	结果集是否包含该字段
字段名	结果集中默认返回的字段名
返回字段名	调用方可以将结果集中默认返回的字段名修改为该字段
字段类型	字段的数据类型
字段描述	字段名的解释（中文名）
备注	字段备注信息
主键字段	是否是主键
是否加密	是否获取加密的字段值，详见3.2
查询条件	调用方传入参数里面的whereSql里可以自定义字段的过滤条件，但只支持查询条件中的过滤操作符。如字段名为a，查询条件为=，则whereSql可以写“a = 4”，不可以写“a > 4”
是否转化为String	是否获取字符串形式的数据

1.2 测试接口

点击接口文档右上角的测试按钮，可以测试当前申请的接口是否返回了正常结果，如下图所示，

测试接口

2. 点击发送，即可测试接口

Request Body (application/json)

```
1 v {  
2 v     "queryCondition": {  
3 v         "argMap": {  
4 v             "staffId": "1"  
5 v         }  
6 v     },  
7 v     "pageIndex": 1  
8 }
```

发送

Response

```
1 v {  
2 v     "hasNext": false,  
3 v     "prevId": null,  
4 v     "pageIndex": null,  
5 v     "pageSize": 0,  
6 v     "sequenceNo": null,  
7 v     "content": []  
8 }
```

执行sql

1, 输入参数值

3,点击可以查看执行具体sql

注意：线上数据已脱敏，非真实数据

关闭

在左侧修改参数，点击发送后可以得到接口返回结果。注意：在uat和线上环境的数据集进行了脱敏。

测试接口

Request Body (application/json)发送Response执行sql

```
1 v {
2 v   "queryCondition": {
3 v     "argMap": {
4 v       "staffIdList": [
5 v         "integer",
6 v         "integer"
7 v       ],
8 v       "staffAccountNameList": [
9 v         "string",
10 v        "string"
11 v      ],
12 v      "orgIdList": [
13 v        "integer",
14 v        "integer"
15 v      ],
16 v      "hrStatusIdList": [
17 v        "integer",
18 v        "integer"
19 v      ],
20 v      "leaderStaffIdList": [
21 v        "integer",
22 v        "integer"
23 v      ],
24 v      "manageUnitIdList": [
25 v        "integer",
26 v        "integer"
27 v      ]
28 v    },
29 v    "prevId": null
30 v  }
31 }
```

整型需要去掉引号任何参数都可以修改为其他值

注意：线上数据已脱敏，非真实数据

关闭

1.3. 调用接口

我的API > 接口文档

授权信息

所属应用: dos-test-app(dos-test-app)

过滤范围: StaffId = 1

老接口: -

状态: 使用中

授权理由: test

POST地址: /api/esb/dos-interface-server/open-api/config/my-daily-test/qhitest222/dos-test-app/data 地址前缀 

请求头: hrgw-appname; hrgw-signature; hr-timestamp; 参

调入参数

参数名	参数重命名
staffId	

授权字段

(全部) (已订阅) (未订阅)

是否订阅	字段名	返回字段名	字段类型	字段描述	备注	主键字段	是否加密
<input checked="" type="checkbox"/>	StaffID	StaffID	INT	StaffID	-	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	EngName	EngName	VARCHAR	EngName	-	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	ChnName	ChnName	VARCHAR	ChnName	-	<input type="checkbox"/>	
<input checked="" type="checkbox"/>	FullName	FullName	VARCHAR	FullName	-	<input type="checkbox"/>	

1.3.1 request url

查看我的API详情页面

核心人事主数据和数仓	前缀	后缀
测试环境	http://uat-ntsgw.woa.com/api/esb/dos-interface-server/open-api/config	/{{domain}}/{{interface_name}}/{{apply_app_name}}/data
OA线上	http://ntsgw.woa.com/api/esb/dos-interface-server/open-api/config	/{{domain}}/{{interface_name}}/{{apply_app_name}}/data
IDC线上	http://idc.ntsgw.woa.com/ntsgw/api/esb/dos-interface-server/open-api/config	/{{domain}}/{{interface_name}}/{{apply_app_name}}/data

1.3.2 Header

1. 需要添加的header

hrgw-timestamp : 秒级时间戳

hrgw-appname : 服务市场应用名

hrgw-signature : CryptoJS.SHA256(appName+token+timestamp).toString(CryptoJS.enc.Hex).toUpperCase()

注释 : token 取自服务市场[线上环境](#) , 服务市场[测试环境](#)

Content-Type : 固定值“application/json”

postman配置计算Header值

生成签名的脚本

```
1 let token = '2948*****e40a1a18be8';
2 let appName = 'dos-***-yu';
3
4 //appName
5 pm.request.headers.add({key: 'hrgw-appname', value: appName });
6
7 // timestamp
8 let timestamp = Math.round(new Date().getTime() / 1000);
9 console.log(timestamp);
10 pm.request.headers.add({key: 'hrgw-timestamp', value: timestamp });
11
12 // signature
13 let signature =
14 CryptoJS.SHA256(appName+token+timestamp).toString(CryptoJS.enc.Hex);
15 console.log(signature);
16 pm.request.headers.add({key: 'hrgw-signature', value: signature });
17
18 //requestId
19 pm.request.headers.add({key: 'requestId', value: uuid() });
20 function uuid() {
21   return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxx'.replace(/[xy]/g, function(c) {
22     var r = Math.random() * 16 | 0, v = c == 'x' ? r : (r & 0x3 | 0x8);
23     return v.toString(16);
24   });
25 }
```

1.3.3 Request Body

找到自己的授权接口([uat 环境](#)，[线上环境](#))，打开详情。

授权信息

配置信息

所属应用: 测评管理(tencent.hr.recruit.globalrecruit.cping)

数据源: v2-nebula-test

过滤范围: -

配置名称: test-qhi

老接口: -

接口摘要: testbbb

状态: 草稿态

分页方式: 普通分页

可见应用: tencent.hr.recruit.globalrecruit.cping

授权理由: test

用户limit缓存:

POST地址: /hrmd/test-qhi/tencent.hr.recruit.globalrecruit.cping/data 地址前缀 ^ 接口版本: v8

API文档: <https://test2-hrdos.woa.com/interface/document?code=test-qhi%2Ftencent.hr.recruit.globalrecruit.cping>

调入参数

参数名	参数重命名	参数类型	默认值	是否必填	操作	描述	备注
id		integer		是	IN		
release_state		string		否	=		
warning_type		integer		是	=		

```

1 {
2     "timeZone": "Asia/Shanghai",      //选填， 配置返回时区，如果数据库字段本身带时区则返
3     "timeout": 20,                  //选填，设置超时时间，单位是 second
4     "queryCondition": {
5         "prevId": null,            //如果是游标分页，需填prevId, 第一页从 null 开
6         "limit": n,                //选填，相当于top n，只返回前n条数据，如果想查全量
7         数据，不传应该查询该参数，limit最大传入1000
8         "argMap": {                //选填，在接口申请页，查看可调用参数
9             "release_state": "",    //按照调用入参的 name 设置 argMap; required =
10            true 则必填，false 则非必填
11             "id": [1, 2],           //操作类型是 in 的需要传数组，最多传入 2000 个，如
12             如果type=integer，则入参是 [1,2,3]； 如果type=String 则入参是 ["1", "2", "3"]
13             "warning_type": 0
14         }
15         "dollarReplaceMap": {
16             "test": "1 = 1" // sql 中的$占位符，内容会替换占位的位置
17         },
18         "userOrderBys": ["orderByField1", "orderByField2 desc"] //用以替换 @OrderBy
19         占位符
20     }
21 }
```

1.3.4 response

```

1  {
2      "success": true,
3      "data": {
4          "hasNext": true, //是否会有下一页
5          "prevId": null, //游标id, 用于下一次查询的游标入参
6          "content": [
7              {
8                  "id": 1,
9                  "start_time": "1683820800000",
10                 "release_state": "0",
11                 "warning_type":0
12             },
13             {
14                 "id": 2,
15                 "start_time": "1683820800000",
16                 "release_state": "0",
17                 "warning_type":0
18             }
19         ]
20     },
21     "message": "success",
22     "code": 0
23 }

```

2.分页查询

2.1 普通分页

The screenshot shows the configuration for a '普通分页' (Simple Paging) setup. On the right side, under the '配置信息' (Configuration Information) section, the '分页方式' (Paging Method) is set to '普通分页' (Simple Paging), which is highlighted with a red box. Other settings shown include '配置名称' (Configuration Name: test-group-by), '数据源' (Data Source: dw3-rw-tbase-dpc), and '可见应用' (Visible Applications: 全部 - All). On the left side, there is a sidebar with navigation links: '我的API', '数据源', '模型管理', '接口配置', '接口授权', and '应用配置'. The main content area displays '授权信息' (Authorization Information) and '状态' (Status: 使用中 - In Use).

注意: 如果分页id有重复，可能会出现分页乱序的问题

```
1  {
2      "pageIndex": 1,                      //选填，起始页
3      "pageSize": 2,                       //选填，每页数量，最大填1000
4      "timeZone": "Asia/Shanghai",         //选填，配置返回时区，如果数据库字段本身带时区则返
回时间数据时使用该时区，如果数字库字段不带时区，则返回UTC时间
5      "timeout": 20,                      //选填，设置超时时间，单位是 second
6      "queryCondition": {
7          "argMap": {                     //选填，在接口申请页，查看可调用参数
8              "release_state": ""},       //按照调用入参的 name 设置 argMap; required =
true 则必填，false 则非必填
9              "id": [1, 2],               //操作类型是 in 的需要传数组，最多传入 2000 个，如
果type=integer，则入参是 [1,2,3]； 如果type=String 则入参是 ["1", "2", "3"]
10             "warning_type": 0
11         }
12         "userOrderBys": ["columnName1 desc", "columnName2 asc"]    //选填按照给定的
column 字段去排序
13         "dollarReplaceMap:{"
14             "test": "1 = 1"           //选填 sql 中的$占位符，内容会替换占位的位置
15         }
16         "dataScopeRightQuery": {   //选填 行列权限配置
17             "appCode": "appCode",     //选填，授权应用code
18             "noRightHint": "**",      //选填 列权限默认隐藏字段显示值
19             "staffId": "00316174",    //选填 行列权限查看人工号
20             "viewedStaffIdList": ["00316175", "00316177"] //选填，人才透视人看人
21         }
22     }
23 }
24 }
```

每次传入pageIndex表示第几页，响应中会包含是否有下一页，响应如：

```

1  {
2      "success": true,
3      "data": {
4          "pageIndex": null,
5          "pageSize": 2,           //每页数量
6          "total": 1000,         //总数
7          "totalPage": 1         //总页码数
8          "content": [           //返回数据列
9              {
10                 "id": 1,
11                 "start_time": "2023-05-11T16:00:00.000000Z",
12                 "release_state": 0,
13                 "warning_type": 0
14             },
15             {
16                 "id": 2,
17                 "start_time": "2023-05-11T16:00:00.000000Z",
18                 "release_state": 0,
19                 "warning_type": 0
20             }
21         ]
22     },
23     "message": "success",
24     "code": 0
25 }

```

2.2 游标分页

The screenshot shows the '我的API' (My API) section of the DOS interface. On the left, there's a sidebar with navigation links: 我的API, 数据源, 模型管理, 接口配置, 接口授权, and 应用配置. The main area displays the configuration for an API endpoint. It includes sections for 授权信息 (Authorization Information) and 配置信息 (Configuration Information). In the Authorization section, it shows 属性应用: hrmr-test(hrmr-test), 过滤范围: -, 老接口: -, 状态: 使用中 (In Use), 授权理由: test, 分页拉取: /hrmr/api-private-workday-scene-staff-post-timing-copy/hrmr-test/data 地址前缀, 全量拉取: /hrmr/api-private-workday-scene-staff-post-timing-copy/hrmr-test/file 示例, and 请求头: hrgw-appname; hrgw-signature; hrgw-timestamp; 参考. In the Configuration section, it shows 数据源: hrmr-datasource-r-pg-md3-global, 配置名称: api-private-workday-scene-staff-post-timing-copy, 接口摘要: 场景化: 通过super.org_id获取岗位信息, 分页方式: 游标分页 (highlighted with a red box), 可见应用: oversea-interface-gateway, hrmr-test, 用户limi缓存: (checkbox), and 领域: hrmr.

用户根据游标id查询本页数据，游标为空时默认是第一页，每一页数据都会返回下一页游标id，这样也就可以一页地翻上翻下，直至没有数据返回。

注意：不保证返回每页数据大小是固定的

请求为：

```

1  {
2      "timeZone": "Asia/Shanghai",      //选填， 配置返回时区，如果数据库字段本身带时区则返
3      //回时间数据时使用该时区，如果数据库字段不带时区，则返回UTC时间
4      "timeout": 20,                //选填，设置超时时间，单位是 second
5      "queryCondition": {
6          "prevId": null,           //第一次不用赋值，如果返回值里hasNext = true，则
7          //把返回值的 prevId 作为参数传到下一页查询请求中去
8          "limit": n,              //选填，相当于top n，只返回前n条数据。由于传了
9          //limit不用count总数，可以提高双倍的查询效率
10         "argMap": {               //选填，在接口申请页，查看可调用参数
11             "release_state": "",   //按照调用入参的 name 设置 argMap; required =
12             //true 则必填，false 则非必填
13             "id": [1, 2],           //操作类型是 in 的需要传数组，最多传入 100 个，如果
14             type=integer, 则入参是 [1,2,3]； 如果type=String 则入参是 ["1", "2", "3"]
15             "warning_type": 0
16         }
17     }
18 }

```

响应为：

```

1  {
2      "success": true,
3      "data": {
4          "hasNext": true,        //是否会有下一页的标志
5          "prevId": 10,          //下一页的游标值，下一次入参 queryCondition.prevId = 10
6          "pageIndex": 1,
7          "pageSize": 10,         //返回值的数量，注意：这个值可能每次不一样
8          "content": [
9              {
10                 "id": 1,
11                 "start_time": "2023-05-11T16:00:00.000000Z",
12                 "release_state": 0,
13                 "warning_type": 0
14             },
15             {
16                 "id": 2,
17                 "start_time": "2023-05-11T16:00:00.000000Z",
18                 "release_state": 0,
19                 "warning_type": 0
20             }
21         ]
22     },
23     "message": "success",
24     "code": 0
25 }

```

3 全量拉取

数据开放服务 DOS

我的API > 接口文档

授权信息

所属应用: dos-test-yu(dos-test-yu)

过滤范围: -

老接口: -

状态: 使用中

授权理由: 自动化测试勿动 (多表)

POST地址: /hrmd/whereSql-01/dos-test-yu/data 地址前缀 ^

文件地址: /hrmd/whereSql-01/dos-test-yu/file 示例 ↴ **全量拉取** 

请求头: hrgw-appname; hrgw-signature; hr-timestamp; 参考 ^

配置信息

数据源: whereSql

配置名称: whereSql-01

接口摘要: 自动化测试勿动

分页方式: 普通分页 可见应用: 全部

用户limit缓存:

领域: hrmd

调入参数

参数名	参数重命名	参数类型	默认值	是否必填	操作
staffid		integer		是	IN
engname		string		否	IN

授权字段

④ 全部 ① 已订阅 ② 未订阅 请输入字段名

是否订阅	字段名	返回字段名	字段类型	字段描述	备注	主键字段	是否加密
✓	staffid	staffid	int4	staffid	-	✓	✓
✓	engname	engname	varchar	engname	-	✓	✓
✓	org_id	org_id	int4	org_id	-	✓	✓
✓	staff_category	staff_category	int4	staff_category	-	✓	✓
✓	staff_property	staff_property	int4	staff_property	-	✓	✓
✓	managementsubject	managementsubject	int4	managementsubject	-	✓	✓

header 参考1.3.2

requestBody

```

1  {
2      "timeZone": "Asia/Shanghai",      //选填， 配置返回时区，如果数据库字段本身带时区则返
3      "timeout": 20,                  //选填，设置超时时间，单位是 second
4      "queryCondition": {
5          "argMap": {                  //选填，在接口申请页，查看可调用参数
6              //必填参数需要填，非必填参数可不填
7          }
8      }
9  }

```

全量拉取下来是个文件，每行都是一个json，每行都以回车键结束

返回值参考

文件名.json

{ key1: val1, key2: val2 }

```
{ key11: val11, key22: val22 }
```

3.1、判断文件是否下载完整

通过响应数据Http响应头中返回"File-Key"的值来查询

请求接口地址,方法 : post

环境	前缀	后缀
测试环境	http://uat-ntsgw.woa.com/api/esb/dos-interface-server/open-api/config	/{domain}/{interface_name}/{apply_app_name}/verify
OA线上	http://ntsgw.woa.com/api/esb/dos-interface-server/open-api/config	/{domain}/{interface_name}/{apply_app_name}/verify
IDC线上	http://idc.ntsgw.woa.com/ntsgw/api/esb/dos-interface-server/open-api/config	/{domain}/{interface_name}/{apply_app_name}/verify

请求body

```
1 {  
2   "fileKey": "7c13558e310a4ccca967ae2a48ec7a9c"  
3 }
```

响应body

```
1 {  
2   "success": true,  
3   "data": {  
4     "fileName": "hrmd_md-api-public-core-selector-dictionary-item-realtime_DIY-Tool_d42676e9128c2db8b766af0a95ed060e.json",  
5     "rowCount": 6784, //文件行数  
6     "finish": true //是否已完成数据传输, false: 传输异常  
7   },  
8   "message": "success",  
9   "code": 0  
10 }
```

4、举个例子

4.1、postman 导入下面文件

[call_dos.json](#)

My Workspace

New Import < model POST hr- POST dos- POST qui- POST sso POST 核心 POST 核心 POST 其他 POST 其他 POST call dos > + test

call dos / 其他接口-普通分页

POST http://uat-ntsgw.woa.com/api/esb/dos-interface-server/open-api/config/dos-test-yu/staff-auto-test5/dos-test-yu/data

Params Authorization Headers (0) Body Pre-request Script Tests Settings

Body

```
1 {
2     "queryCondition": {
3         "argMap": {
4             "staffid": [
5                 "1"
6             ]
7         },
8         "dataScopeRightQuery": {
9             "noRightHint": "*",
10            "staffId": "90316174"
11        }
12    },
13    "pageIndex": 1
14 }
```

Body Cookies (3) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 success : true,
2 "data": [
3     {
4         "hasNext": false,
5         "prevId": null,
6         "pageIndex": 1,
7         "pageSize": 1,
8         "sequenceNo": null,
9         "content": [
10             {
11                 "staffid": "*",
12                 "engname": "*",
13                 "workplace_id": "*",
14                 "org_id": "*",
15                 "org_path": "*",
16                 "staff_category": "*",
17                 "staff_status": "*",
18                 "staff_property": "*"
19             }
20         ]
21     }
22 ]
```

Status: 200 OK Time: 160 ms Size: 723 B Save as example

Send Cookies Beautify

导入配置

4.2、服务市场token

My Workspace

New Import

POST hr- POST dos- POST qui- POST sso- POST 核心/ POST 核心/ POST 其他/ POST 其他/ call dos > + < test

call dos / 其他接口-普通分页

Save Send

Pre-request Scripts are written in JavaScript and are run before the request is sent.

Snippets

- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable

需要去服务市场里把 token 和 appName 换成自己对应的应用

POST http://uat-ntsgw.woa.com/api/esb/dos-interface-server/open-api/config/dos-test-yu/staff-auto-test5/dos-test-yu/data

Params Authorization Headers (9) Body Pre-request Script Tests Settings

```
1 let token = '2948b03d-c35b-4f46-a542-ce40a1a18be8';
2 let appName = 'dos-test-yu';
3
4 pm.request.headers.add({key: 'hrgw-appname', value: appName});
5
6 // timestamp
7 let timestamp = Math.round(new Date().getTime() / 1000);
8 console.log(timestamp);
9 pm.request.headers.add({key: 'hrgw-timestamp', value: timestamp});
10
11 // signature
12 let signature = CryptoJS.SHA256(appName+token+timestamp).toString(CryptoJS.enc.Hex);
13 console.log(signature);
14 pm.request.headers.add({key: 'hrgw-signature', value: signature});
```

Status: 200 OK Time: 160 ms Size: 723 B Save as example

Body Cookies (3) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
success : true,
"data": {
  "hasNext": false,
  "prevId": null,
  "pageIndex": 1,
  "pageSize": 1,
  "sequenceNo": null,
  "content": [
    {
      "staffid": "*",
      "engname": "*",
      "workplace_id": "*",
      "org_id": "*",
      "org_path": "*",
      "staff_category": "*",
      "staff_status": "*",
      "staff_property": "*",
    }
  ]
}
```

数据开放服务 DOS

帮助 nebulaliao

我的API | 接口授权 > 详情

测试 复制 授权版本 导出配置 < 返回

授权信息

所属应用 dos-test-yu(dos-test-yu)

过滤范围: -

老接口: -

状态: 使用中

授权理由: autotest

分页拉取: /api/esb/dos-interface-server/open-api/config/dos-test-yu/staff-auto-test5/dos-test-yu/data 地址前缀 ^

全量拉取: /api/esb/dos-interface-server/open-api/config/dos-test-yu/staff-auto-test5/dos-test-yu/file 示例 ^

API文档: <https://uat-hrdos.woa.com/interface/document?code=staff-auto-test5%2Fdos-test-yu>

授权版本

版本记录	状态	操作
20231027155043575	草稿态	详情 上线
20231026180200645	待使用	详情 上线
20231026175259201	使用中	详情 下线
20231026175256509	待使用	详情 上线

领域: dos-test-yu

uat-servicemarket.woa.com/appMarket

服务市场 API Market 我的 应用市场 服务列表 nebulaliao

输入关键字进行过滤

腾讯集团 > 搜索条件:daily

注册外部应用

daily 共为您搜索到: 1条结果

应用名	描述	状态	创建时间	操作
my-daily-test	my-daily-test	激活	2022-10-19 20:44:19	详情

共 1 条 1 / 10条/页 前往 1 / 1 页

服务市场uat环境

<https://uat-servicemarket.woa.com/appMarket>

服务市场线上

<https://uat-servicemarket.woa.com/appMarket>

4.3 调用自己接口

```

1 let token = '2948b03d-c35b-4f46-a542-ce40a1a18be8';
2 let appName = 'dos-test-yu';
3
4 pm.request.headers.add({key: 'hrgw-appname', value: appName });
5 // timestamp
6 let timestamp = Math.round(new Date().getTime() / 1000);
7 console.log(timestamp);
8 pm.request.headers.add({key: 'hrgw-timestamp', value: timestamp });
9
10 // signature
11 let signature = CryptoJS.SHA256(appName+token+timestamp).toString(CryptoJS.enc.Hex);
12 console.log(signature);
13 pm.request.headers.add({key: 'hrgw-signature', value: signature });
14
    
```

需要把上面例子中的 appName, token 和 请求地址后缀换成自己的即可

5、加解密配置

如果调用方只希望获取明文的数据，可以忽略该配置

5.1 dos加盐规则

只有调用方需要加密数据时才会考虑是否加盐，加盐时需要配置字段的encrypt_salt_type为“yes”，否则配置为“no”，加盐规则为：

32位小写字母的UUID + 竖线 + 真实数据，如真实数据为abc，那么加盐结果为：

ecfcf741929043adab7e434800e66238 | abc

5.2 dos特殊数据加解密处理方法

1. 如果数据为null，则会将数据null转为特殊字符串"!@#%&!#%@"，再加盐加密
2. 如果数据为空字符串，且需要加盐加密，则加盐后的结果为：32位小写字母UUID + 竖线

5.3. RSA加解密说明

The screenshot shows the 'Application Configuration' page for the 'dos-test-wu' application. The 'RSA' encryption algorithm is selected, and the public key is being input. The 'Public Key' field has a red border around it, indicating it is a required field.

应用ID: dos-test-wu
所属应用: v_zehhaowu
应用描述: v_zehhaowu测试
管理员: kunkkawang, v_qizhang, v_zehhaowu, v_dingjiaye, v_bupingu, chloeyueli, nebulaiao, v_chuliwang, v_pegnzhang
操作人: 共0人
应用加密算法: RSA
密钥位数: 1024
加密公钥: 可输入或生成公钥
必填
tips: 私钥解密, 请“参考”

当调用方需要获取密文数据时，需要在应用授权界面，填写自己应用的公钥或由dos生成符合条件的密钥（dos只保存公钥，不保存生成的私钥），目前仅支持1024位RSA加密算法。

受限于原生的RSA加密仅支持最长为117字节数据的加解密，dos采用通用的分段加密方法，加密时每117字节加密一次，解密时每128字节解密一次，调用方如果配置了加密，为确保数据正确性，需要以这种方式将数据解密。下面给出了Java版本的加解密方法。

5.4. java版本RSA分段加解密方法

```
1 import java.io.UnsupportedEncodingException;
2 import java.nio.charset.StandardCharsets;
3 import java.util.Arrays;
4 import java.util.Base64;
5 import javax.crypto.BadPaddingException;
6 import javax.crypto.Cipher;
7
8 import com.tencent.hr.dos.common.utils.EncrytionUtil;
9
10 import java.security.KeyFactory;
11 import java.security.KeyPair;
12 import java.security.KeyPairGenerator;
13 import java.security.SecureRandom;
14 import java.security.interfaces.RSAPrivateKey;
15 import java.security.interfaces.RSAPublicKey;
16 import java.security.spec.PKCS8EncodedKeySpec;
17 import java.security.spec.X509EncodedKeySpec;
18 import java.util.HashMap;
19 import java.util.Map;
20 import javax.crypto.IllegalBlockSizeException;
21 import org.slf4j.Logger;
22 import org.slf4j.LoggerFactory;
23
24 public class RSAUtil{
25
26     private static final Logger log = LoggerFactory.getLogger(RSAUtil.class);
27
28     /**
29      * 密钥长度 于原文长度对应 以及越长速度越慢
30      * 密钥大小范围从384位到16, 384位
31      * 默认1024
32      */
33
34     public final static String encryptionPublicKey = "encryptionPublicKey";
35
36     public final static String encryptionPrivateKey = "encryptionPrivateKey";
37
38     private final static int KEY_SIZE = 1024;
39
40
41     /**
42      * 随机生成密钥对
43      *
44      * @return
45      * @throws Exception
46      */
47
48     public static Map<String, String> genKeyPair() throws Exception {
49         // KeyPairGenerator类用于生成公钥和私钥对，基于RSA算法生成对象
50         KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
51         // 初始化密钥对生成器
52         keyPairGen.initialize(KEY_SIZE, new SecureRandom());
53         // 生成一个密钥对，保存在keyPair中
```

```
54     KeyPair keyPair = keyPairGen.generateKeyPair();
55     // 得到私钥
56     RSAPrivateKey privateKey = (RSAPrivateKey) keyPair.getPrivate();
57     // 得到公钥
58     RSAPublicKey publicKey = (RSAPublicKey) keyPair.getPublic();
59     String publicKeyString =
60         Base64.getEncoder().encodeToString(publicKey.getEncoded());
61         // 得到私钥字符串
62     String privateKeyString =
63         Base64.getEncoder().encodeToString(privateKey.getEncoded());
64
65     Map<String, String> keyMap = new HashMap<>();
66
67     // 将公钥和私钥保存到Map
68     // encryptionPublicKey表示公钥
69     keyMap.put(encryptionPublicKey, publicKeyString);
70     // encryptionPrivateKey表示私钥
71     keyMap.put(encryptionPrivateKey, privateKeyString);
72     return keyMap;
73 }
74
75 public static String encrypt(String str, String publicKey) throws Exception {
76     log.info("RSA公钥加密前的数据|str:{}|publicKey:{}", str, publicKey);
77     //base64编码的公钥
78     byte[] decoded = Base64.getDecoder().decode(publicKey);
79     RSAPublicKey pubKey = (RSAPublicKey) KeyFactory.getInstance("RSA").
80             generatePublic(new X509EncodedKeySpec(decoded));
81     //RSA加密
82     Cipher cipher = Cipher.getInstance("RSA");
83     cipher.init(Cipher.ENCRYPT_MODE, pubKey);
84
85     //当长度过长的时候，需要分割后加密 117个字节
86     byte[] resultBytes = getMaxResultEncrypt(str, cipher);
87
88     String outStr = Base64.getEncoder().encodeToString(resultBytes);
89     log.info("公钥加密后的数据|outStr:{}", outStr);
90     return outStr;
91 }
92
93 private static byte[] getMaxResultEncrypt(String str, Cipher cipher) throws
94     IllegalBlockSizeException, BadPaddingException {
95     byte[] inputArray = str.getBytes();
96     int inputLength = inputArray.length;
97     // log.info("加密字节数|inputLength:{}", inputLength);
98     // 最大加密字节数，超出最大字节数需要分组加密
99     int MAX_ENCRYPT_BLOCK = 117;
100    // 标识
101    int offSet = 0;
102    byte[] resultBytes = {};
103    byte[] cache = {};
104    while (inputLength - offSet > 0) {
105        if (inputLength - offSet > MAX_ENCRYPT_BLOCK) {
```

```
103             cache = cipher.doFinal(inputArray, offSet, MAX_ENCRYPT_BLOCK);
104             offSet += MAX_ENCRYPT_BLOCK;
105         } else {
106             cache = cipher.doFinal(inputArray, offSet, inputLength -
107             offSet);
108             offSet = inputLength;
109         }
110         resultBytes = Arrays.copyOf(resultBytes, resultBytes.length +
111         cache.length);
112         System.arraycopy(cache, 0, resultBytes, resultBytes.length -
113         cache.length, cache.length);
114     }
115     /**
116      * RSA私钥解密
117      *
118      * @author gggcgb 【wechat:13031016567】
119      * @param str          加密字符串
120      * @param privateKey 私钥
121      * @return 铭文
122      * @throws Exception 解密过程中的异常信息
123      */
124     public static String decrypt(String str, String privateKey) throws
Exception {
125         log.info("RSA私钥解密前的数据|str:{}", str);
126         //64位解码加密后的字符串
127         byte[] inputByte = Base64.getDecoder().decode(str.getBytes("UTF-8"));
128         //base64编码的私钥
129         byte[] decoded = Base64.getDecoder().decode(privateKey);
130         RSAPrivateKey priKey = (RSAPrivateKey) KeyFactory.getInstance("RSA")
131             .generatePrivate(new PKCS8EncodedKeySpec(decoded));
132         //RSA解密
133         Cipher cipher = Cipher.getInstance("RSA");
134         cipher.init(Cipher.DECRYPT_MODE, priKey);
135         //        String outStr = new String(cipher.doFinal(inputByte));
136         //当长度过长的时候，需要分割后解密 128个字节
137         String outStr = new String(getMaxResultDecrypt(str, cipher));
138         log.info("RSA私钥解密后的数据|outStr:{}", outStr);
139         return outStr;
140     }
141
142
143
144     private static byte[] getMaxResultDecrypt(String str, Cipher cipher)
145         throws IllegalBlockSizeException, BadPaddingException,
UnsupportedEncodingException, UnsupportedEncodingException {
146         byte[] inputArray = Base64.getDecoder().decode(str.getBytes("UTF-8"));
147         int inputLength = inputArray.length;
148         //        log.info("解密字节数|inputLength:{}", inputLength);
149         // 最大解密字节数，超出最大字节数需要分组加密
150         int MAX_ENCRYPT_BLOCK = 128;
```

```
151     // 标识
152     int offSet = 0;
153     byte[] resultBytes = {};
154     byte[] cache = {};
155     while (inputLength - offSet > 0) {
156         if (inputLength - offSet > MAX_ENCRYPT_BLOCK) {
157             cache = cipher.doFinal(inputArray, offSet, MAX_ENCRYPT_BLOCK);
158             offSet += MAX_ENCRYPT_BLOCK;
159         } else {
160             cache = cipher.doFinal(inputArray, offSet, inputLength -
161             offSet);
162             offSet = inputLength;
163         }
164         resultBytes = Arrays.copyOf(resultBytes, resultBytes.length +
165             cache.length);
166         System.arraycopy(cache, 0, resultBytes, resultBytes.length -
167             cache.length, cache.length);
168     }
169     return resultBytes;
170 }
171 }
```

6、idc调用OA

(只有生产环境) POST http://idc.ntsgw.woa.com/ntsgw/api/esb/dos-interface-server/open-api/config/{domain}/{interface_name}/{caller_app_name}/data

~~(已经不需要手动添加白名单了) 新增应用需要在idc网络调用,请联系 @chengchaolv 或 @yiwuhe 在网关添加白名单,需要告知其应用名的code~~