

DM3.1 RPS JsonOpenAPI文档

DM3.1 RPS JsonOpenAPI文档

文档信息

1. 使用说明

- 1.1. 服务访问URL
- 1.2. 基本说明
 - 1.2.1. Body与Body参数
 - 1.2.2. Query参数
 - 1.2.3. Form参数
 - 1.2.4. 示例语言
- 1.3. REST API 签名规则
 - 1.3.1. 前提
 - 1.3.2. 系统级Header
 - 1.3.3. 计算签名
 - 1.3.4. 签名与调用API示例
 - 1.3.4.1. Body参数API调用示例
 - 1.3.4.2. Query参数API调用示例
 - 1.3.5. 签名规则小结
 - 1.3.6. 防重放机制
- 1.4. 返回数据结构
- 1.5. 错误代码说明

2. 服务器接口

- 2.1. 添加服务器
- 2.2. 分页查询服务器
- 2.3. 查询服务器详情
- 2.4. 检测服务器名称是否存在
- 2.5. 编辑服务器
- 2.6. 删除服务器

3. 设备接口

- 3.1. 批量添加设备
- 3.2. 分页查询设备信息
- 3.3. 查询设备详情
- 3.4. 检测设备是否注册
- 3.5. 检测设备是否存在
- 3.6. 查询服务器列表
- 3.7. 编辑设备
- 3.8. 批量迁移设备
- 3.9. 批量删除设备
- 3.10. 检测设备状态并返回绑定的服务器url
- 3.11. 批量重置连接

4. 企业ip白名单

- 4.1. 批量添加
- 4.2. 批量删除
- 4.3. 分页获取企业ip白名单列表

5. 拦截记录

- 5.1. 分页查询拦截记录

6. 注意点

- 6.1. 鉴权用户名和密码
- 6.2. 设备接口的默认处理与提示
- 6.3. 设备被其他企业添加
- 6.4. 服务器名称

7. 返回错误信息说明及解决方法

8. 附录

文档信息

日期	文档版本	项目版本	修改内容	作者	审核
2018.11.20	1.0.0.0	V3.1	增加预配置功能	侯万琦	张明勋
2018.12.08	1.0.0.1	V3.1	增加签名规则示例和错误信息说明	侯万琦	张明勋
2020.02.19	1.0.0.2	V3.4	增加支持重置连接、增删企业ip白名单、获取设备被拦截信息等	夏贵灵	张明勋

1. 使用说明

1.1. 服务访问URL

<https://dm.rps.yealink.com>（改为生产环境）

在访问服务器接口或者设备接口时，需要在请求URL前拼接该URL。如2.1. 添加服务器接口：

请求URL:

- POST：/api/open/v1/server/add

其中，POST表示请求方法为POST；/api/open/v1/server/add为请求URL。则访问该接口的完整URL为：<https://dm.rps.yealink.com/api/open/v1/server/add>，其他接口类似。

1.2. 基本说明

1.2.1. Body与Body参数

文档中的Body指HTTP请求的请求体，Body参数表示，**需要将请求的参数放在HTTP请求体中。**

特别注意：

1. 文档中凡是指定参数为Body参数的，则HTTP请求头的**Content-Type**需要指定为 **application/json;charset=UTF-8**，即数据格式为json。
2. 文档中，参数为Body参数的接口，**可以不输入Body参数，但Body不能为空**（即请求的Content-Length不能为0）。如分页查询服务器接口，该接口的所有Body参数都可以不输入，**当所有参数都不输入时，Body的内容应为{}，而不是为空。**如下请求示例：

```
{ }
```

或

```
{
  "key":null
}
```

1.2.2. Query参数

本文中的Query参数表示，需要将参数附带在请求url后，如检测设备是否存在接口：

请求URL:

- GET /api/open/v1/device/checkMac

请求参数:

- Query参数：

请求参数	参数类型	是否必须	长度	参数说明
mac	String	是	17	mac地址

则完整的请求url为：

<https://dm.rps.yealink.com/api/open/v1/device/checkMac?mac=001565AEF921>

特别注意：

- 强烈建议不要将Query参数放到Body中。因为将Query参数放到Body中，没有什么意义，而且还需要计算后文中提到的Content-MD5。
- 后文中，所提到的Query参数，都表示要将参数附带在请求url后，且Body为空（即不将Query参数放到Body中）。后文给出的签名示例以及附录中的示例，都基于Query参数不放到Body里面（Body为空，即Content-Length等于0）。

1.2.3. Form参数

本文中的Form参数指使用Form表单提交的参数。目前的开放接口里面，没有Form参数的API。

1.2.4. 示例语言

文档中如无特殊说明，给出的示例代码都是Java代码或伪代码。

1.3. REST API 签名规则

1.3.1. 前提

在调用API之前，您需要从[亿联设备管理平台RPS端](#)中获得AccessKey ID 与AccessKey Secret，前者用在REST API的请求的Header中作为身份凭证，后者用来完成签名。（**注意修改链接为正式环境的连接**）

1.3.2. 系统级Header

发送HTTP请求调用接口，需要在HTTP Header中添加如下Header参数：

HTTP Header Key	HTTP Header Value
X-Ca-Key	AccessKey ID
X-Ca-Timestamp	unix时间戳，毫秒级
X-Ca-Nonce	随机字符串，建议使用UUID
Content-MD5	对Body的MD5摘要值进行Base64编码的结果
X-Ca-Signature	签名字符串

- X-Ca-Key：AccessKey ID，需要在[亿联设备管理平台RPS端](#)申请。（[修改链接](#)）
- X-Ca-Timestamp：调用API时刻的Unix时间戳，注意是**毫秒级，且添加到header时是字符串形式**。
- X-Ca-Nonce：随机字符串，建议使用UUID，即通用唯一识别码（Universally Unique Identifier）。结合时间戳防重放，具体机制见1.3.6小节。
- X-Ca-Signature：签名字符串，签名方法见1.3.3小节。
- Content-MD5：对Body的MD5摘要值进行Base64编码的结果。
 - **特别注意：**
 - 当Body为空（Content-Length等于0）或者请求参数为Form参数时，Header里面不需要添加Content-MD5。
 - 对于本文档提供的接口而言，**参数为Body参数的API都要添加Content-MD5，参数为Query参数的都不需要添加Content-MD5。**
 - *前文中指出不建议将Query参数放到Body里面，如果要放到Body里面，则需要添加Content-MD5*
 - Content-MD5计算方法：
 - 首先，采用MD5算法对Body进行摘要，得到Body的MD5摘要值。**注意，MD5摘要得到的结果是128bit的二进制数字，不要转换成字符串**
 - 其次，采用Base64编码对Body的MD5摘要值进行编码，即得到Content-MD5。**注意，编码的对象是128bit的摘要值。**
如果使用Java语言，则可按照如下步骤计算：
 - 获取Body的字节数组bodyBytes
 - 采用MD5算法对bodyBytes进行摘要得到bodyMd5Bytes。**注意，md5摘要后得到的是字节数组，不要转换成字符串。**
 - 最后，采用Base64编码对bodyMd5Bytes进行编码得到结果。**注意，这一步是对md5摘要后的字节数组进行Base64编码。**

Java代码示例如下：

```
String contentMD5 =
    new String(Base64.encodeBase64(md5(body.getBytes("UTF-8"))), "UTF-8");
//byte[] md5(byte[] bodyBytes)该方法为md5摘要算法，具体实现见附录
```

1.3.3. 计算签名

计算签名的过程分为如下3大步骤：

1. 拼接参与的签名项，得到待签名字符串，记为stringToSign

签名项	说明
-----	----

签名项	说明
HTTPMethod	全大写，例如：POST，GET
Headers	需要的系统级Header，见下文说明
API_URI	请求URL去掉首部的"/",如api/open/v1/server/add
FormattedQFStr	Query参数或者Form参数格式化后的字符串，格式化方式见下文

- 参与的签名项和拼接方式如下所示：

```
String stringToSign =
    HTTPMethod + "\n" +      //"\n"是换行符
    Headers + "\n" +
    API_URI + "\n" +
    FormattedQFStr;
```

- 签名项说明：

- **HTTPMethod:** HTTP请求方法全大写，如：POST，GET，PUT，DELETE，本文档中的API只涉及POST和GET
- **Headers:** 需要的系统级Header按照自然序对Header Key进行排序，然后按如下方式进行拼接

```
//需要Content-MD5情况: Body参数的接口都需要
String headers =
    "Content-MD5:" + content_md5_value + "\n" +
    "X-Ca-Key:" + x_ca_key_value + "\n" +
    "X-Ca-Nonce:" + x_ca_nonce_value + "\n" +
    "X-Ca-Timestamp:" + x_ca_timestamp_value + "\n";

//不需要Content-MD5情况: Query参数都不需要
String headers =
    "X-Ca-Key:" + x_ca_key_value + "\n" +
    "X-Ca-Nonce:" + x_ca_nonce_value + "\n" +
    "X-Ca-Timestamp:" + x_ca_timestamp_value + "\n";
```

- **API_URI:** 接口的请求URL去掉首部的"/", 如api/open/v1/server/add
- **FormattedQFStr:** Query参数或者Form参数格式化后的字符串，格式化方式为：
 - 首先，对 **Query 与 Form** 参数按照自然顺序对 **参数名（Key）** 进行排序
 - 然后，按照如下方式进行拼接

```
String Params =
    Key1 + "=" + value1 + "&" +
    Key2 + "=" + value2 + "&" +
    ...
    KeyN + "=" + valueN; //注意，最后一个没有"&"
```

特别注意：

- Body参数不需要拼接该签名项
- 当参数值为空时，仅保留参数名，“=”不参与以上拼接。如key1=""或key1=" ",则拼接如下

```
String Params =
    Key1 + "&" +    //value1为空, "="不参与拼接
    Key2 + "=" + value2 + "&" +
    ...
    KeyN + "=" + valueN; //注意, 最后一个没有"&"
```

2. 采用HMAC-SHA256算法，将AccessKey Secret作为密钥，对stringToSign进行摘要，得到stringToSign的消息摘要，**结果记为msgDgst**。

- **特别注意：**这一步得到的消息摘要是256bit的二进制数字，不要转换成字符串
- 如果使用Java语言，则可按如下示例进行：

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
//secret是在设备管理平台RPS端申请的AccessKey Secret
hmacSha256.init(new SecretKeySpec(secret.getBytes("UTF-8"), "HmacSHA256"));
byte[] msgDgst = hmacSha256.doFinal(stringToSign.getBytes("UTF-8"));
```

3. 对msgDgst进行Base64编码，得到最终的签名字符串。

特别注意：Base64编码的对象是256bit的摘要值，不是字符串。

```
String sign = new String(Base64.encodeBase64(msgDgst), "UTF-8"));
```

1.3.4. 签名与调用API示例

下面，分别以2.2. 分页查询服务器和3.5.检测设备是否存在为例，给出调用Body参数和Query参数接口的基本过程，如需完整代码，请查看附录。需要注意，示例中的HttpRequest使用的是[jodd](#)中的HttpRequest。

1.3.4.1. Body参数API调用示例

- 请求URL：POST /api/open/v1/server/list
- 请求参数：Body参数；如下

```
{
    "key": "TestServer",
    "skip": 0
}
```

1. 首先登陆[亿联设备管理平台RPS端](#)中获得AccessKey ID 与AccessKey Secret，如下结果：(地址改为正式环境)

```
AccessKey ID : "2df23f2d9c255e7138dc603b3847b58a"
AccessKey Secret: "d4a4be460a8d43609d8e8a5e7d0d4ad1"
```

2. 创建HTTP Request并添加系统级Header

```
//输入的参数
String input = "{" +
    "\"key\": \"TestServer\", \"skip\": 0" +
    "}";

//创建HTTP Request
HttpRequest request = new HttpRequest()
```

```

.method("POST")
.set("https://dm.yealink.com/api/open/v1/server/list")
.body(input.getBytes("UTF-8")), "application/json");

//计算系统级header的值
String x_ca_key="2df23f2d9c255e7138dc603b3847b58a";//accessKeyId
String x_ca_nonce = "b681e77450a04d22aaffc914a3379561"; //UUID
String x_ca_timestamp = "1544008291631"; //当前时间的unix时间戳ms
//请求参数为Body参数，需要计算Content-MD，按前文所述方法计算结果如下
String content_md5 = "KOK7YpXasjJC+Ms1P+MGWW==";

//设置请求Header
request
.header("X-Ca-Key", x_ca_key)
.header("X-Ca-Timestamp", x_ca_timestamp)
.header("X-Ca-Nonce", x_ca_nonce)
.header("Content-MD5", content_md5);

```

3. 计算签名

- 拼接参与的签名项，得到待签名字符串

```

//已经计算的Header的value
String content_md5 = "KOK7YpXasjJC+Ms1P+MGWW==";
String x_ca_key = "2df23f2d9c255e7138dc603b3847b58a";
String x_ca_nonce = "b681e77450a04d22aaffc914a3379561";
String x_ca_timestamp = "1544008291631"

//注意Header的顺序，自然顺序；
String stringToSign =
    "POST" + "\n" +
    "Content-MD5:" + content_md5 + "\n" +
    "X-Ca-Key:" + x_ca_key + "\n" +
    "X-Ca-Nonce:" + x_ca_nonce + "\n" +
    "X-Ca-Timestamp:" + x_ca_timestamp + "\n" +
    "api/open/v1/server/list"; //最后一行，不要有"\n"
//Body参数的API，不需要签名项FormattedQFStr

```

拼接参与签名项的结果如下，即stringToSign的值如下

```

POST
Content-MD5:KOK7YpXasjJC+Ms1P+MGWW==
X-Ca-Key:2df23f2d9c255e7138dc603b3847b58a
X-Ca-Nonce:b681e77450a04d22aaffc914a3379561
X-Ca-Timestamp:1544008291631
api/open/v1/server/list

```

- 采用HMAC-SHA256算法，将AccessKey Secret作为密钥，对stringToSign进行摘要，得到stringToSign的消息摘要

```
//secret是在设备管理平台RPS端申请的AccessKey Secret
String secret = "d4a4be460a8d43609d8e8a5e7d0d4ad1";

Mac hmacSha256 = Mac.getInstance("HmacSHA256");
hmacSha256.init(new SecretKeySpec(secret.getBytes("UTF-8"),
"HmacSHA256"));
byte[] msgDgst = hmacSha256.doFinal(stringToSign.getBytes("UTF-8"));
```

- 对上一步结果进行Base64编码，得到签名

```
String sign = new String(Base64.encodeBase64(msgDgst), "UTF-8"));
```

- 将得到的签名添加到HTTP Request Header里面

```
request.header("X-Ca-Signature", sign);
```

至此该接口需要的5个系统级Header都已经添加完成，接下来，愉快地发送请求并获取响应即可。

4. 发送请求获取响应

```
HttpResponse response = request.send();
String bodyText = response.bodyText(); //响应体字符串
```

1.3.4.2. Query参数API调用示例

- 请求URL：GET /api/open/v1/device/checkMac
- 请求参数：Query参数；输入参数mac=001565123123则完整的请求url为：

<https://yealink.dm.com/api/open/v1/device/checkMac?mac=001565123123>

1. 同Body参数，首先获得AccessKey ID 与AccessKey Secret，如下结果：

```
AccessKey ID : "2df23f2d9c255e7138dc603b3847b58a"
AccessKey Secret: "d4a4be460a8d43609d8e8a5e7d0d4ad1"
```

2. 创建HTTP Request并添加系统级Header

```
//创建HTTP Request
HttpRequest request = new HttpRequest()
.method("GET")
.set("https://dm.yealink.com/api/open/v1/device/checkMac?mac=001565123123")

//计算系统级header的值，Query参数，不需要添加Content-MD5
String x_ca_key="2df23f2d9c255e7138dc603b3847b58a";//accessKeyId
String x_ca_nonce = "9e730a223b48433785494801fb016d39"; //UUID
String x_ca_timestamp = "1544094691000"; //当前时间的unix时间戳

//设置请求Header
request
.header("X-Ca-Key", x_ca_key)
.header("X-Ca-Timestamp", x_ca_timestamp)
.header("X-Ca-Nonce", x_ca_nonce)
```

3. 计算签名

- 拼接参与的签名项，得到待签名字符串

```
//已经计算的Header的value
String x_ca_key = "2df23f2d9c255e7138dc603b3847b58a";
String x_ca_nonce = "9e730a223b48433785494801fb016d39";
String x_ca_timestamp = "1544094691000"

//注意Header的顺序，自然顺序；
String stringToSign =
    "GET" + "\n" +
    "X-Ca-Key:" + x_ca_key + "\n" +
    "X-Ca-Nonce:" + x_ca_nonce + "\n" +
    "X-Ca-Timestamp:" + x_ca_timestamp + "\n" +
    "api/open/v1/device/checkMac" + "\n" +
    "mac=001565123123"; //签名项FormattedQFStr
```

拼接参与签名项的结果如下，即stringToSign的值如下

```
GET
X-Ca-Key:2df23f2d9c255e7138dc603b3847b58a
X-Ca-Nonce:9e730a223b48433785494801fb016d39
X-Ca-Timestamp:1544094691000
api/open/v1/device/checkMac
mac=001565123123
```

之后的过程与Body参数API的相同，此处不再赘述。

1.3.5. 签名规则小结

以上小节，给出了系统级Header和签名的计算说明，先概括如下：

- 调用Body参数的接口，需要在HTTP Request Header中添加如下header
 - X-Ca-Key
 - X-Ca-Nonce
 - X-Ca-Timestamp
 - X-Ca-Signature
 - Content-MD5

计算签名时，不需要签名项FormattedQFStr；签名项Headers里面，需要Content-MD5

- 调用Query参数（或者Form参数）的接口，需要在HTTP Request Header中添加如下header
 - X-Ca-Key
 - X-Ca-Nonce
 - X-Ca-Timestamp
 - X-Ca-Signature

计算签名时，需要签名项FormattedQFStr；签名项Headers里面，不需要Content-MD5

1.3.6. 防重放机制

为防止重放攻击，使用X-Ca-Nonce结合X-Ca-Timestamp来防重放。在以下3种情况下，会认为请求无效，将会返回错误信息request.replay。

1. 从请求发起时刻到服务器接收到请求的时间间隔超过5分钟，则请求无效。
2. 服务器接收到请求时刻的时间戳小于或等于X-Ca-Timestamp，则请求无效。

3. 5分钟内两次请求所携带的X-Ca-Nonce值一样，则请求无效。如A时刻访问某个接口B，X-Ca-Nonce为“123456789”，则A时刻后的5分钟内，如果访问接口（接口B或其他接口）又携带该X-Ca-Nonce，则请求无效。如果超过了5分钟，则请求有效。

综上，建议每次请求的X-Ca-Nonce使用UUID。

重放返回结果如下：

```
{
  "ret": -1,
  "data": null,
  "errors": {
    "msg": "",
    "errorCode": 401,
    "fieldErrors": [
      {
        "field": [],
        "msg": "request.replay"
      }
    ]
  }
}
```

1.4. 返回数据结构

数据返回格式，采用JSON字符串，字符采用UTF-8编码。统一返回数据结构：

```
{
  "ret": -1,           // -1表示错误，否则为正确
  "data": null,        // 返回数据
  "errors": {
    "msg": "",          // 错误信息
    "errorCode": 400,   // 错误代码
    "fieldErrors": [    // 详细错误信息
      {
        "field": "",     // 错误域
        "msg": ""        // 错误信息
      }
    ]
  }
}
```

- ret：结果代码，数字形式，小于0表示失败，大于或等于0表示成功
- data：返回的数据对象
- errors：返回的错误对象
- msg：错误信息
- errorCode：错误代码
- fieldErrors：详细错误信息，记录错误域与具体错误信息

请求成功示例

```
{
  "ret": 1,
  "data": {
    "existed": false,
    "self": null
  },
  "error": null
}
```

请求失败示例：

```
{
  "ret": -1,
  "data": null,
  "error": {
    "msg": "server.not.found",
    "errorCode": 404,
    "fieldErrors": []
  }
}
```

1.5. 错误代码说明

code	说明	备注
200	请求成功	
400	客户端请求包含错误	如MAC参数不是合法的mac地址
401	鉴权失败	如用户密钥错误
404	资源不存在	如用户修改一个已经删除的资源
405	Http Method不被支持	
406	请求不被接受	
409	资源冲突	如注册邮箱重复
415	不支持的媒体类型	
444	接口参数错误	如缺少了接口所必须的参数
500	服务器内部错误	
502	网关错误	
503	远程服务不可达	
504	网关超时	

2. 服务器接口

2.1. 添加服务器

请求URL:

- POST : /api/open/v1/server/add

请求参数:

- **Body参数 :**

请求参数	参数类型	是否必须	长度	参数说明
serverName	String	是	20	服务器名称
url	String	是	512	服务器Url
authName	String	否	32	鉴权用户名
password	String	否	32	鉴权密码
certificateUrl	String	否	--	证书url
serverCertificateUrl	String	否	--	服务器证书url
serverCertificateEnable	String	否	--	自定义证书, "1"表示开启

- 请求示例 :

```
{
  "serverName": "TestServer",
  "url": "https://https://www.yealink.com",
  "authName": "Seakeer",
  "password": "123456",
  "certificateUrl": "https://www.yealink.com/certificate"
}
```

正确返回结果 :

```
{
  "ret": 1,
  "data": {
    "id": "b38dea23a4e6458188799833b72d950f",
    "serverName": "TestServer",
    "url": "https://www.yealink.com",
    "authName": "Seakeer"
  },
  "error": null
}
```

2.2. 分页查询服务器

请求URL:

- POST : /api/open/v1/server/list

请求参数:

- **Body参数**

请求参数	参数类型	是否必须	长度	参数说明
key	String	否	-	服务器名称或url,用于模糊查询
skip	Integer	否	-	跳过的记录数，默认为0
limit	Integer	否	-	每页查询的最大记录数
autoCount	boolean	否	-	是否自动统计总数，默认为false

- 请求示例：

```
{
  "key": "TestServer",
  "skip": 0,
  "limit": 10,
  "autoCount": true
}
```

正确返回结果：

```
{
  "ret": 2,
  "data": {
    "skip": 0,
    "limit": 10,
    "total": 2,
    "autoCount": true,
    "orderbys": [
      {
        "field": "modifyTime",
        "order": -1
      }
    ],
    "data": [
      {
        "_id": "b38dea23a4e6458188799833b72d950f",
        "id": "b38dea23a4e6458188799833b72d950f",
        "serverName": "TestServer",
        "url": "https://www.yealink.com",
        "userId": "8320f8baee7f4639985fa24b5a1f0131",
        "authName": "Seakeer",
        "password": "***#",
        "enterpriseId": "0e0736cfed5c451baca69351e3b1b6bf",
        "phoneCount": 0,
        "createTime": 1541465552869,
        "modifyTime": 1541465552874,
        "deleted": false
      },
      {...}
    ]
  },
  "error": null
}
```

2.3. 查询服务器详情

请求URL:

- GET : /api/open/v1/server/detail

请求参数:

- Query参数 :

请求参数	参数类型	是否必须	长度	参数说明
id	String	是	32	服务器Id

正确返回结果 :

```
{
  "ret": 1,
  "data": {
    "createTime": 1541465552869,
    "modifyTime": 1542683736995,
    "deleted": false,
    "_id": "b38dea23a4e6458188799833b72d950f",
    "serverName": "YealinkServer",
    "url": "http://www.yealink.com",
    "userId": "8320f8baee7f4639985fa24b5a1f0131",
    "authName": "Yealink",
    "password": "***#",
    "enterpriseId": "0e0736cfed5c451baca69351e3b1b6bf",
    "certificateUrl": "http://cer/cer.cer",
    "id": "b38dea23a4e6458188799833b72d950f"
  },
  "error": null
}
```

2.4. 检测服务器名称是否存在

请求URL:

- GET : /api/open/v1/server/checkServerName

请求参数:

- Query参数 :

请求参数	参数类型	是否必须	长度	参数说明
serverName	String	是	20	服务器名称

正确返回结果 :

- 名称已存在

```
{
  "ret":1,
  "data":true,
  "error":null
}
```

- 名称不存在

```
{
  "ret":1,
  "data":false,
  "error":null
}
```

2.5. 编辑服务器

请求URL:

- POST : /api/open/v1/server/edit

请求参数:

- **Body参数 :**

请求参数	参数类型	是否必须	长度	参数说明
id	String	是	32	服务器Id
serverName	String	是	20	服务器名称
url	String	是	512	服务器Url
authName	String	否	32	鉴权用户名
password	String	否	32	鉴权密码
certificateUrl	String	否	--	证书url
serverCertificateUrl	String	否	--	服务器证书url
serverCertificateEnable	String	否	--	自定义证书, "1"表示开启, 不传该参数, 表示关闭

- 请求示例 :

```
{
  "id": "b38dea23a4e6458188799833b72d950f",
  "serverName": "YealinkServer",
  "url": "http://www.yealink.com",
  "authName": "Yealink",
  "password": "Yealink",
  "certificateUrl": "http://cer/cer.cer"
}
```

正确返回结果：

```
{
  "ret": 1,
  "data": {
    "id": "b38dea23a4e6458188799833b72d950f",
    "serverName": "YealinkServer",
    "url": "http://www.yealink.com",
    "authName": "Yealink"
  },
  "error": null
}
```

2.6. 删除服务器

请求URL:

- POST : /api/open/v1/server/delete

请求参数:

- **Body参数：**

请求参数	参数类型	是否必须	长度	参数说明
ids	List	是	-	要删除的服务器Id集合

- 请求示例：

```
{
  "ids": [
    "2b371ec5a8f046a9adcb5eafd167b30c",
    "9d11957ea8b1475c9336e2d2c6a6b93c"
  ]
}
```

正确返回结果：

```
{
  "ret": 0,
  "data": null,
  "error": null
}
```

3. 设备接口

3.1. 批量添加设备

请求URL:

- POST /api/open/v1/device/add

请求参数:

- **Body参数 :**

请求参数	参数类型	是否必须	长度	参数说明
macs	List	是	--	mac地址列表
serverId	String	否	32	服务器id
uniqueServerUrl	String	否	512	autop服务器url，为个别设备单独设置，当设置了此项后，话机进行rps请求时优先跳转到此项设置的值。没有值时才会跳转到绑定的服务器url。
remark	String	否	256	备注
authName	String	否	32	鉴权用户名
password	String	否	32	鉴权密码
email	String	否	--	企业邮箱
number	String	否	--	企业ID

- 请求示例 :

```
{
  "macs": [
    "aa00000000aa",
    "aa0000000000",
  ],
  "serverId": "ba7c7b13ed114a5fa6f12063ea9dff41",
  "uniqueServerUrl": "https://www.yealink.com",
  "remark": "SeakeerDevice",
  "authName": "Seakeer",
  "password": "654321"
}
```

正确返回结果 :

```
{
  "ret": 1,
  "data": [
    {
      "id": "82c2df80fca745d3b7a1405b02bd55a8",
    }
  ]
}
```

```
    "mac": "aa000a0000aa",
    "serverId": "ba7c7b13ed114a5fa6f12063ea9dff41",
    "serverName": "SeakeerTestServer",
    "remark": "SeakeerDevice",
    "uniqueServerUrl": "https://www.yealink.com",
    "authName": "Seakeer"
  },
  {...}
],
"error": null
}
```

3.2. 分页查询设备信息

请求URL:

- POST /api/open/v1/device/list

请求参数:

- **Body参数：**

请求参数	参数类型	是否必须	长度	参数说明
key	String	否	-	关键字
status	String	否	-	状态标识bound/unbound
skip	Integer	否	-	跳过的记录数，默认为0
limit	Integer	否	-	每页查询的最大记录数
autoCount	Boolean	否	-	是否自动统计总数，默认为false

- 请求示例：

```
{
  "key": "aa000a",
  "status": "bound",
  "skip": 0,
  "autoCount": true,
  "limit": 10,
}
```

正确返回结果：

```
{
  "ret": 1,
  "data": {
    "skip": 0,
    "limit": 10,
    "total": 1,
    "autoCount": true,
    "orderbys": [
      {
        "field": "modifyTime",
        "order": -1
      }
    ]
  }
}
```

```
}
],
"data": [
  {
    "_id": "82c2df80fca745d3b7a1405b02bd55a8",
    "id": "82c2df80fca745d3b7a1405b02bd55a8",
    "mac": "aa000a0000aa",
    "userId": "8320f8baee7f4639985fa24b5a1f0131",
    "enterpriseId": "0e0736cfed5c451baca69351e3b1b6bf",
    "enterpriseName": "SeakeerRPS",
    "serverId": "b25ac1016caf416a90d5ca1ee438153a",
    "serverName": "SeakeerServerTest",
    "ipAddress": null,
    "dateRegistered": 1542680124026,
    "lastConnected": null,
    "remark": "edit",
    "uniqueServerUrl": "http://edit.com",
    "serverUrl": "https://dm30-devtest.yealinkclient.com/dm.cfg",
    "resellerId": "400ed821a4774900aa4ee0e7d1931465",
    "resellerName": "SeakeerReseller",
    "createTime": 1542680124026,
    "formattedCreateTime": null,
    "formattedLastConnected": null,
    "authName": "edit",
    "password": "***#",
    "initEnable": true,
    "redirect": "",
    "locked": false
  }
]
},
"error": null
}
```

3.3. 查询设备详情

请求URL:

- GET /api/open/v1/device/detail

请求参数:

- Query参数

请求参数	参数类型	是否必须	长度	参数说明
id	String	是	32	设备id

正确返回结果：

```
{
  "ret": 1,
  "data": {
    "createTime": 1542680124026,
    "modifyTime": 1542682680211,
    "deleted": false,
    "_id": "82c2df80fca745d3b7a1405b02bd55a8",
```

```
{
  "mac": "aa000a0000aa",
  "userId": "8320f8baee7f4639985fa24b5a1f0131",
  "enterpriseId": "0e0736cfed5c451baca69351e3b1b6bf",
  "enterpriseName": "SeakeerRPS",
  "serverId": "b25ac1016caf416a90d5ca1ee438153a",
  "serverName": "SeakeerServerTest",
  "ipAddress": null,
  "dateRegistered": 1542680124026,
  "lastConnected": null,
  "remark": "edit",
  "uniqueServerUrl": "http://edit.com",
  "resellerId": "400ed821a4774900aa4ee0e7d1931465",
  "resellerName": "SeakeerReseller",
  "authName": "edit",
  "password": "***#",
  "randomStr": null,
  "id": "82c2df80fca745d3b7a1405b02bd55a8"
},
"error": null
}
```

3.4. 检测设备是否注册

请求URL:

- GET /api/open/v1/device/checkDevice

请求参数:

- Query参数 :

请求参数	参数类型	是否必须	长度	参数说明
mac	String	是	17	mac地址

正确返回结果 :

- 设备未在平台找到

```
{
  "ret": 1,
  "data": "Unknown",
  "error": null
}
```

- 未注册

```
{
  "ret": 1,
  "data": "Unregistered",
  "error": null
}
```

- 被其他用户注册

```
{
  "ret":1,
  "data":"Registered Elsewhere",
  "error":null
}
```

- 已注册

```
{
  "ret":1,
  "data":"Registered",
  "error":null
}
```

3.5. 检测设备是否存在

请求URL:

- GET /api/open/v1/device/checkMac

请求参数:

- Query参数：

请求参数	参数类型	是否必须	长度	参数说明
mac	String	是	17	mac地址

正确返回结果：

- 不存在

```
{
  "ret": 1,
  "data": {
    "existed": false,
    "self": null
  },
  "error": null
}
```

- 存在且属于自己

```
{
  "ret": 1,
  "data": {
    "existed": true,
    "self": true
  },
  "error": null
}
```

- 存在但不属于自己

```
{
  "ret": 1,
  "data": {
    "existed": true,
    "self": false
  },
  "error": null
}
```

3.6. 查询服务器列表

请求URL:

- GET /api/open/v1/device/serverList

请求参数:

无

正确返回结果 :

```
{
  "ret": 2,
  "data": [
    {
      "id": "b25ac1016caf416a90d5ca1ee438153a",
      "serverName": "seakeerServerTest"
    },
    {
      "id": "ba7c7b13ed114a5fa6f12063ea9dffa1",
      "serverName": "seakeerTestServer"
    },
  ],
  "error": null
}
```

3.7. 编辑设备

请求URL:

- POST /api/open/v1/device/edit

请求参数:

- **Body参数 :**

请求参数	参数类型	是否必须	长度	参数说明
id	String	是	32	设备id
serverId	String	否	32	服务器id
uniqueServerUrl	String	否	512	单独设置的autop服务器url，优先级高于绑定服务器中的url
remark	String	否	256	备注

authName 请求参数	参数 类型	是否 必须	长 度	鉴权用户名 参数说明
password	String	否	32	鉴权密码

- 请求示例：

```
{
  "id": "82c2df80fca745d3b7a1405b02bd55a8",
  "serverId": "b25ac1016caf416a90d5ca1ee438153a",
  "remark": "edit",
  "uniqueServerUrl": "http://edit.com",
  "authName": "edit",
  "password": "edit"
}
```

正确返回结果

```
{
  "ret": 1,
  "data": {
    "id": "82c2df80fca745d3b7a1405b02bd55a8",
    "mac": "aa000a0000aa",
    "serverId": "b25ac1016caf416a90d5ca1ee438153a",
    "serverName": "SeakeerServerTest",
    "remark": "edit",
    "uniqueServerUrl": "http://edit.com",
    "authName": "edit"
  },
  "error": null
}
```

3.8. 批量迁移设备

请求URL:

- POST /api/open/v1/device/migrate

请求参数:

- Body参数：

请求参数	参数类型	是否必须	长度	参数说明
ids	List	是	-	设备id列表
serverId	String	是	32	服务器id

- 请求示例：

```
{
  "ids": [
    "82c2df80fca745d3b7a1405b02bd55a8",
    "fe3aa53c15ee4e02af2aadf719ebf60d",
  ],
  "serverId": "b25ac1016caf416a90d5ca1ee438153a"
}
```

正确返回结果：

```
{
  "ret": 2,
  "data": [
    {
      "id": "82c2df80fca745d3b7a1405b02bd55a8",
      "mac": "aa000a0000aa",
      "serverId": "b25ac1016caf416a90d5ca1ee438153a",
      "serverName": "SeakeerServerTest",
      "remark": "edit",
      "uniqueServerUrl": "http://edit.com",
      "authName": "edit"
    },
    {...}
  ],
  "error": null
}
```

3.9. 批量删除设备

请求URL:

- POST /api/open/v1/device/delete

请求参数:

请求参数	参数类型	是否必须	长度	参数说明
ids	List	是	-	设备id列表

- 请求示例：

```
{
  "ids": [
    "82c2df80fca745d3b7a1405b02bd55a8",
    "fe3aa53c15ee4e02af2aadf719ebf60d"
  ]
}
```

正确返回结果：

```
{
  "ret":0,
  "data":null,
  "error":null
}
```

3.10. 检测设备状态并返回绑定的服务器url

请求URL:

- GET /api/open/v1/device/checkDeviceBoundUrl

请求参数:

- Query参数 :

请求参数	参数类型	是否必须	长度	参数说明
mac	String	是	17	mac地址

正确返回结果 :

- 设备未在平台找到

```
{
  "ret": 1,
  "data": {
    "status": "Unknown",
    "boundUrl": null
  },
  "error": null
}
```

- 未注册

```
{
  "ret": 1,
  "data": {
    "status": "Unregistered",
    "boundUrl": null
  },
  "error": null
}
```

- 被其他用户注册

```
{
  "ret": 1,
  "data": {
    "status": "Registered Elsewhere",
    "boundUrl": null
  },
  "error": null
}
```

- 已注册

```
{
  "ret": 1,
  "data": {
    "status": "Registered",
    "boundUrl": "https://www.yealink.com"
  },
  "error": null
}
```

- 备注：只有设备状态为“Registered”时，才会返回绑定的服务器url。

3.11. 批量重置连接

请求URL:

- POST /api/open/v1/device/batchEnableInit

请求参数:

- Body参数：

请求参数	参数类型	是否必须	长度	参数说明
ids	String	是	32	设备ids

- 请求示例：

```
{
  "ids": ["5825c0739c344762a8cc486acdd48dd2"]
}
```

正确返回结果

```
{
  "ret": 1,
  "data": 1,
  "error": null
}
```

4. 企业ip白名单

4.1. 批量添加

请求URL:

- POST /api/open/v1/ipWhiteList/batchAdd

请求参数:

- Body参数：

请求参数	参数类型	是否必须	长度	参数说明
ips	List	是	--	ip列表

- 请求示例：

```
{
  "ips":["10.200.200.200","10.201.201.201"]
}
```

正确返回结果：

```
{
  "ret": 0,
  "data": null,
  "error": null
}
```

4.2. 批量删除

请求URL:

- POST /api/open/v1/ipWhiteList/batchDelete

请求参数:

- Body参数：

请求参数	参数类型	是否必须	长度	参数说明
ids	List	是	--	id列表

- 请求示例：

```
{
  "ids":
  ["6dde28d7ba7146c8a1afab99f25da29e","e78b82f594304a6eb624f734558001d2"]
}
```

正确返回结果：

```
{
  "ret": 1,
  "data": 2,
  "error": null
}
```

4.3. 分页获取企业ip白名单列表

请求URL:

- POST /api/open/v1/ipWhiteList/getPagedList

请求参数:

- Body参数：

请求参数	参数类型	是否必须	长度	参数说明
key	String	否	-	关键字 (name、ip)
skip	Integer	否	-	跳过的记录数，默认为0
limit	Integer	否	-	每页查询的最大记录数
autoCount	Boolean	否	-	是否自动统计总数，默认为false

- 请求示例：

```
{
  "autoCount": false,
  "data": null,
  "key": "204.42.31.58",
  "limit": null,
  "orderbys": null,
  "skip": null,
  "total": 0
}
```

正确返回结果：

```
{
  "ret": 1,
  "data": {
    "skip": null,
    "limit": 100,
    "total": 0,
    "autoCount": false,
    "orderbys": [
      {
        "field": "modifyTime",
        "order": -1
      }
    ],
    "data": [
      {
        "createTime": 1579594331453,
        "modifyTime": 1579594331454,
        "deleted": false,
        "_id": "7aae0bea0d7149a0812bf350c48d33e8",
        "enterpriseId": "aac8059b08cc4f7c8414ddda3a563464",
        "ip": "104.42.31.53",
        "name": "xxxx@yealink.com",
        "id": "7aae0bea0d7149a0812bf350c48d33e8"
      }
    ]
  },
  "error": null
}
```

5. 拦截记录

5.1. 分页查询拦截记录

请求URL:

- POST /api/open/v1/redirect/getForbiddenList

请求参数:

- **Body参数 :**

请求参数	参数类型	是否必须	长度	参数说明
searchKey	String	否	-	关键字 (ip、 mac)
skip	Integer	否	-	跳过的记录数，默认为0
limit	Integer	否	-	每页查询的最大记录数
autoCount	Boolean	否	-	是否自动统计总数，默认为false
startTime	Long	否	-	开始时间戳
endTime	Long	否	-	结束时间戳

- 请求示例 :

```
{
  "searchKey": "10.82.4.5",
  "skip": 0,
  "autoCount": true,
  "limit": 10,
  "startTime": 1581995820033,
  "endTime": 1581996124366,
}
```

正确返回结果 :

```
{
  "ret": 1,
  "data": {
    "skip": 0,
    "limit": 10,
    "total": 1,
    "autoCount": true,
    "orderbys": [
      {
        "field": "modifyTime",
        "order": -1
      }
    ],
    "data": [
      {
        "id": "ba72314430714dec80104cd22a79d967",
        "type": "ResponseTimeout",
      }
    ]
  }
}
```

```

    "mac": "001565bbb1n9",
    "ip": "10.82.4.5",
    "detail": "Response timeout",
    "createTime": 1581995860033,
    "enterpriseName": "xxx",
    "enterpriseEmail": "xxx@yealink.com",
    "deviceId": "4c3016da62e3410eabd3b78a4703ff07",
    "reportModel": "SIP-T46S",
    "reportFirmware": "66.82.0.90",
    "correctModel": null,
    "correctFirmware": null,
    "forbidden": true,
    "ipStatus": "BLACK",
    "ipCountry": "US",
    "initEnable": null
  }
],
"error": null
}

```

6. 注意点

6.1. 鉴权用户名和密码

- authName和密码必须成对出现
- 编辑服务器和设备时，password不能设置为如下情况，如设置，则不生效

```

{
  "password": "***#"
}

```

6.2. 设备接口的默认处理与提示

- 如果uniqueServerUrl=""或者uniqueServerUrl=" "，则返回url.invalid，表示输入的uniqueServerUrl无效
- 如果输入的serverId=""或serverId=" "等，则默认处理serverId=null，serverName=null

6.3. 设备被其他企业添加

- 如果返回如下所示的结果，则表示：您添加的MAC已经被其他企业添加使用。
 - 请核实您输入的MAC地址是否准确；
 - 若需要继续添加该MAC，请到 <https://ticket.yealink.com> 创建ticket进行申诉，我们将进行审核。您需要提供的资料包括：
 - (1)设备的MAC和SN
 - (2)企业名称
 - (3)设备固件版本号
 - (4)登录帐号

```
{
  "ret": -1,
  "data": "000000000000",
  "error": {
    "msg": "device.mac.added.by.other",
    "errorCode": 409,
    "fieldErrors": []
  }
}
```

6.4. 服务器名称

服务器名称在整个系统层次上要求唯一，即如果服务器名称A被其他企业（用户）使用了，则不能再使用A服务器名，否则会提示错误信息：server.name.existed，表示服务器名称已经存在（可能自己企业使用也可能是其他企业使用）

7. 返回错误信息说明及解决方法

错误信息	说明	解决方法
service.common.token.unauthorized	鉴权失败	检查accesskey和签名是否正确
Content.MD5.not.null	Content-MD5不能为空	调用Body参数的接口，要添加Content-MD5
Content.MD5.invalid	Content-MD5无效	Content-MD5计算错误，请重新计算
accesskey.id.invalid	AccessKey ID 无效	检查AccessKey ID是否失效或重新获取
request.header.invalid	Signature 无效	签名计算错误，请重新计算
request.header.invalid	请求头错误	请求头错误，参考1.3.2节
request.replay	请求重放	参考1.3.6节
url.invalid	url格式无效	注意url支持的协议和规范 http://...;https://...;ftp://...;tftp://...
url.too.long	url太长	url长度不能超过512个字符
id.repeated	输入的id列表里面有重复项	去掉重复id，每个id输入一次即可
server.id.invalid	服务器id无效	使用有效的服务器id
auth.name.too.long	鉴权用户名太长	不能超过32个字符
auth.name.password.must.be.couple	鉴权用户名和密码必须成对出现	用户名和密码要么都输入，要么都不输入
auth.name.or.password.inputted.not.empty	输入的用户名和密码不能为空字符串	输入了用户名和密码，用户名和密码不能为""，或者" "，" "...
Ids.not.empty	id列表不能为空	输入id
device.operate.forbidden	操作设备权限不足	设备是其他企业的，如有疑问，参考4.3
device.not.found	设备未找到	检查输入的设备id或mac
device.mac.existed	设备已经存在	设备已经添加过，如果要修改，可使用编辑设备接口
device.mac.added.by.other	设备已经被其他企业添加了	见4.3
device.mac.invalid	MAC格式无效	支持如下格式 001565121212 00 15 65 12 12 12 00-15-65-12-12-12 00:15:65:12:12:12
device.mac.needed	需要输入MAC	输入mac
device.mac.repeated	输入mac列表里面有重复项	去掉重复mac,每个mac输入一次即可
device.remark.too.long	备注太长	备注不能超过256个字符
device.macs.contains.empty.item	mac列表里面存在空MAC	去掉mac列表里面的空值

错误信息	说明	解决方法
server.name.not.blank	服务器名称不能为空	输入不为空字符串的服务器名称
server.url.not.blank	服务器url不能为空	输入不为空的服务器url
server.name.existed	服务器名称已经存在	见4.4
server.not.found	服务器未找到	检查输入的服务器id或名称
server.operate.forbidden	操作服务器权限不足	服务器是其他企业的，使用自己企业的服务器
server.name.too.long	服务器名称太长	服务器名称不能超过256个字符
not.email.account	不存在该邮箱对应的账号	输入正确的企业邮箱
not.email.enterprise	不存在该邮箱企业	输入正确的企业邮箱
not.email.permission	没有该邮箱企业权限	输入正确的企业ID

8. 附录

附录提供了使用Java语言计算签名和调用api的示例代码，仅供参考。

<https://github.com/yealink/RpsJsonOpenApiDemo.git>