Yealink Json API for RPS Management Platform

1. Introduction

1.1. The URL of Service Access

https://api-dm.yealink.com:8443

When accessing the server interface or the device interface, you need add the above URL before the request URL. For example, when adding the server interface:

Request URL:

- POST：/api/open/v1/server/add

The request method is POST and the request URL is /api/open/v1/server/add. The complete URL to access the server interface is https://api-dm.yealink.com:8443/api/open/v1/server/add. It is also applicable to access other interfaces.

1.2. Basic Introduction

1.2.1. Body and Body Parameter

In this document, Body refers to the HTTP request body and Body Parameter means you need add the request parameters to the Body of HTTP request.

Note:

1. For the Body parameter specified in this document, the **Content-Type** of the HTTP request header should be **application/json;charset=UTF-8**, that is to say, the data format is in JSON.
2. In this document, if the parameter of an interface is Body parameter, you can choose **not to enter the Body parameter, but the Body cannot be null** (that is the request Content-length cannot be 0). Take requesting the server interface by paging as an example, **all the Body parameters of this interface can be unentered, and under this circumstance, the Body is {} rather than null**. Following is the example request.

   ```
   {  }
   ```

   or

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

1.2.2. Query Parameter

The Query parameter means that you need to add the request parameter behind the request URL. For example, detecting whether or not the device has interface:

Request URL:

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

Request parameter:

- **Query Parameter:**

Parameter	Type	Required	Length	Description
mac	String	Yes	17	The MAC address.

The complete request URL is:

https://api-dm.yealink.com:8443/api/open/v1/device/checkMac?mac=001565AEF921

Note:

- It is highly recommended that you should not add the Query parameter to the Body, because it doesn't make any sense and you need to calculate the Content-MD5 (mentioned in the following parts).
- The Query parameter mentioned in the following parts means that you need add the parameter behind the request URL and the Body should be null (do not add the Query parameter to the Body).
- The signature examples and the examples in the Appendix mentioned in the following parts are all based on the circumstance of not adding the Query parameter to the Body, that is to say, the Body is null and the Content-Length is 0.

1.2.3. Form Parameter

The Form parameter is the parameter submitted by the Form table. There is no Form parameter in this API we provided now.

1.2.4. Example Language

If there is no additional notes, the example codes in this document are in Java or pseudo code.

1.3. Signature Rule of REST API

1.3.1. Prerequisites

Before calling the API, you need to obtain the AccessKey ID and the AccessKey Secret from Yealink Device Management Cloud Service for RPS Enterprise. The AccessKey ID is used as an identity in the request Header of REST API and the AccessKey Secret is used to complete the signature.

1.3.2. System Header

When sending the HTTP request to call the interface, you need add the following Header parameters to the HTTP Header.

HTTP Header Key	HTTP Header Value
X-Ca-Key	AccessKey ID.
X-Ca-Timestamp	Unix timestamp, in millisecond.
X-Ca-Nonce	Random string. UUID is recommended.
Content-MD5	The result after Base64 encoding the MD5 digest value of the Body.
X-Ca-Signature	Signature string.

- X-Ca-Key: AccessKey ID, applied in Yealink Device Management Cloud Service for RPS Enterprise.
- X-Ca-Timestamp: the Unix timestamp for calling the API time. Note that it is in millisecond and it is in string format when adding to the Header.
- X-Ca-Nonce: random string, Universally Unique Identifier (UUID) is recommended. It can be used together with X-Ca-Timestamp to avoid replaying, refer to Section 1.3.6 for more information.
- X-Ca-Signature: the signature string, refer to Section 1.3.3 for more information.
- Content-MD5: the result after Base64 encoding the MD5 digest value of the Body.
  - **Note:**
    - When the Body is null (the content-Length is 0) or when the request parameter is Form parameter, the Content-MD5 is not required in the Header.
    - For the interface provided in this document, **when it is Body parameter, you need add Content-MD5, but when it is Query parameter, you needn't**.
    - As we mentioned above, we do not recommend that you add the Query parameter to the Body, if you do, you need add the Content-MD5.
  - The calculation method of Content-MD5 is described as below:
  -
    - First, digest the Body by using MD5 algorithm to get the MD5 digest value. **Note that the MD5 digest value is 128-bit binary number. Do not convert the MD5 digest value into the string.**
    - Second, encode the MD5 digest value by using Base64 to get Content-MD5. **Note that the MD5 digest value should be 128 bits.**

      If you use Java, use the following method to calculate:
    - Get bodyBytes, the byte array of the Body.
    - Digest the bodyBytes by using MD5 algorithm to get bodyMd5Bytes. **Do not convert the bodyMd5Bytes into string.**
    - Encode the bodyMd5Bytes by using Bse64 to get the result. **Note that the encoding object is bodyMd5Bytes which should not be converted into string.**

```
  String contentMD5 =
  new String(Base64.encodeBase64(md5(body.getBytes("UTF-8"))), "UTF-8");
  //byte[] md5(byte[] bodyBytes)This method is the MD5 digest algorithm, refer to
Appendix for the realization.
```

1.3.3. Calculating the Signature

There are 3 steps to calculate the signature:

1. Splice the signatures to get the string to be signed (stingToSign).

Signature	Short Description
HTTPMethod	In capital letters, for example: POST and GET.
Headers	The required system Header, see the detailed description below.
API_URI	Remove the "/" in front of the request URL, for example, api/open/v1/server/add.
FormattedQFStr	The string you get after formatting the Query parameter or the Form parameter, and the formatting method is described as below.

   - The signature and the splicing method are as below:

```
String stringToSign =
HTTPMethod + "\n" +    //"\n" is the line break
Headers + "\n" +
API_URI + "\n" +
FormattedQFStr;
```

   - Detailed description of the signature:
      - **HTTPMthod:** the HTTP request method is in capital letters, for example, POST, GET, PUT, and DELETE. This document only covers POST and GET.
      - **Headers:** the required system Header. Sequence the Header Key according to the natural order and splice them by the following method:

```
//When Content-MD5 is required: Content-MD5 is required in all interfaces of
the Body parameter
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";

//When Content-MD5 is not required: Content-MD5 is not required in all
interfaces of the Query parameter
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:** remove the "/" in front of the request URL, for example, api/open/v1/server/add
      - **FormattedQFStr:** the string you get after formatting the **Query parameter or the Form parameter**, and the formatting method is described as below:
         - First, sequence the Key of the **Query parameter and the Form parameter** according to the natural order.

- Second, splice the Key by the method below:

```
String Params =
    Key1 + "=" + Value1 + "&" +
    Key2 + "=" + Value2 + "&" +
    ...
    KeyN + "=" + ValueN;
```

Note:

- For Body parameter, you do not need to splice the signature.
- When the parameter value is empty, only keep the parameter name and “=” is not participated in splicing. For example, when the key1="" or key1=" ", splice the following:

```
String Params =
 Key1 + "&" +   //value 1 is empty and "=" is not participated in
splicing
 Key2 + "=" + Value2 + "&" +
 ...
 KeyN + "=" + ValueN;
```

2. Use HMAC-SHA256 algorithm and take AccessKey Secret as the key to digest stringToSign, and you can get the message digest (msgDst).
    - **Note:** The message digest is 256-bit binary number , and do not convert it into string.
    - If you use Java, you can follow the example below:

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
//The secret is the AccessKey Secret applied on Yealink Device Management Cloud
Service for RPS Emterprise
hmacSha256.init(new SecretKeySpec(secret.getBytes("UTF-8"), "HmacSHA256"));
byte[] msgDgst = hmacSha256.doFinal(stringToSign.getBytes("UTF-8");
```

3. Encoding msgDst by using Base64, and you can get the final signature string.

   Note: The encoding object is the 256-bit megDst rather than the megDst in string format.

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

1.3.4. Examples of the Signature and Calling the Interface

Take requesting the server by paging and detecting whether or not the device exists as examples, we introduce the process of calling the interface of Body parameter and Query parameter. For the complete code, refer to Appendix. Note that the HTTPRequest used in the example is the HTTPRequest in Jodd.

1.3.4.1. Example of Calling the Interface of the Body Parameter

- Request URL: POST /api/open/v1/server/list
- Request parameter: the Body parameter below:

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

1. Log into Yealink Device Management Cloud Service for RPS Enterprise and get the AccessKey ID and AccessKey Secret.

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

2. Create HTTP Request and add the system Header.

   ```
   //Enter the parameter
   String input = "{" +
                       "\"key\":\"TestServer\",\"skip\":0" +
                  "}";

   //Creat the 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");

   //Calculate the value of the system header
   String x_ca_key="2df23f2d9c255e7138dc603b3847b58a";//accesskeyId
   String x_ca_nonce = "b681e77450a04d22aaffc914a3379561"; //UUID
   String x_ca_timestamp = "1544008291631"; //The unix timestamp of the current time
   //When the request parameter is Body parameter, you need calculate the Content-MD5,
   the result of the calculated Content-MD5 is as below:
   String content_md5 = "KOK7YpXasjJC+MslP+MGWw==";

   //Set the request 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. Calculate the signature.
   - Splice the signature to get the string to be signed.

     ```
     //The calculated value of the Header
     String content_md5 = "KOK7YpXasjJC+MslP+MGWw==";
     String x_ca_key = "2df23f2d9c255e7138dc603b3847b58a";
     String x_ca_nonce = "b681e77450a04d22aaffc914a3379561";
     String x_ca_timestamp = "1544008291631"

     //Note that the order of the Header is natural;
     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"; //Do not have "\n" in the last line
//The interface of the Body parameter, and the FormattedQFStr is not required
```

The value of the stringToSign is as below:

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

- Use HMAC-SHA256 algorithm and take AccessKey Secret as the key to digest stringToSign, and you can get the message digest (msgDst).

```
//The secret is the AccessKey Secret applied in Yealink Device Management Cloud
Service for RPS Emterprise
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");
```

- Encode msgDst by using Base64, and you can get the final signature string.

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

- Add the final signature string to the HTTP Request Header.

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

After adding at least 5 system Header required by this interface, you can send the request and receive the response.

4. Send the request and receive the response.

```
HttpResponse response = request.send();
String bodyText = response.bodyText();//The response string
```

1.3.4.2. Example of Calling the Interface of the Query Parameter

- Request URL: GET /api/open/v1/device/checkMac
- Request parameter: Query parameter. Enter the parameter: mac=001565123123, and the complete request URL is:

  https://api-dm.yealink.com:8443/api/open/v1/device/checkMac?mac=001565123123

1. Get the AccessKey ID and the AccessKey Secret and the result is as below:

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

2. Create the HTTP Request and add the system Header.

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

//Calculate the value of the system Header and Content-MD5 is not required in Query
parameter
String x_ca_key="2df23f2d9c255e7138dc603b3847b58a";//accesskeyId
String x_ca_nonce = "9e730a223b48433785494801fb016d39"; //UUID
String x_ca_timestamp = "1544094691000"；//The unix timestamp of the current time

//Set the request Header
request
.header("X-Ca-Key", x_ca_key)
.header("X-Ca-Timestamp",  x_ca_timestamp)
.header("X-Ca-Nonce", x_ca_nonce)
```

3. Calculate the signature.
   - Splice the signature and get the string to be signed (stringToSign).

```
//The calculated value of the Header
String x_ca_key = "2df23f2d9c255e7138dc603b3847b58a";
String x_ca_nonce = "9e730a223b48433785494801fb016d39";
String x_ca_timestamp = "1544094691000"

//Note that the order of the Header is natural；
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"; //FormattedQFStr
```

The value of the stringToSign is as below:

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

The subsequent process is the same as that of calling the interface of the Body parameter, in Section 1.3.4.1.

1.3.5. Conclusion about the Signature Rule

The above parts introduce how to calculate the system Header and the signature, it can be concluded as below:

- Call the interface of the Body parameter, and add the following Header to HTTP Request Header.
    - X-Ca-Key
    - X-Ca-Nonce
    - X-Ca-Timestamp
    - X-Ca-Signature
    - Content-MD5

  When calculating the signature, FormattedQFStr is not required; among the signature Headers, Content-MD5 is required.
- Call the interface of the Query parameter or the Form parameter, and add the following Header to HTTP Request Header.
    - X-Ca-Key
    - X-Ca-Nonce
    - X-Ca-Timestamp
    - X-Ca-Signature

  When calculating the signature, FormattedQFStr is required; among the signature Headers, Content-MD5 is not required.

1.3.6. Mechanism of Avoiding Replaying

X-Ca-Nonce and X-Ca-Timestamp are used to prevent replay attack. In the following situations, the request is invalid and the error message (request.replay) will be returned.

1. If the time between the device sending the request and the server receiving the request exceeds 5 minutes, the request is invalid.
2. If the timestamp (when the server receives the request) is no more than X-Ca-Timestamp, the request is invalid.
3. Within 5 minutes, if two requests carry the same X-Ca-Nonce, the request is invalid. For example, if a device accesses the interface A at 12:00 with the X-Ca-Nonce as 123456789, and if the device accesses interface A or other interfaces with the same X-Ca-Nonce before 12:05, the request is invalid. But if the device accesses the any interface after 12:05, the request is valid.

To sum up, we recommend that you use UUID for each X-Ca-Nonce.

The result of replaying is as below:

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

1.4. Returned Data Structure

The returned data form uses JSON encoded by UTF-8. The returned data structure is unified as follows:

```
{
    "ret": -1,                //-1 means error, otherwise it means correct
    "data": null,            //The returned data
    "errors": {
        "msg": "",           //The error message
        "errorCode": 400,    //The error code
        "fieldErrors": [     //The detailed error message
            {
                "field": "",    //The detailed field
                "msg": ""       //The error message
            }
        ]
    }
}
```

- ret: the result code. Value smaller than 0 means error response, otherwise it means successful response.
- data: the returned message when requesting.
- errors: the returned error object.
- msg: the error message.
- errorCode: the error code.
- fieldErrors: the error message which records the error field and the detailed error message.

Example of Successful Response:

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

Example of Error Response:

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

1.5. Errors

code	Description	Example
200	SUCCESS	/
400	BAD_REQUEST	Illegal MAC Address.
401	UNAUTHORIZED	Error user key.
404	NOT_FOUND	Try to modify an deleted object.
405	METHOD_NOT_ALLOWED	/
406	NOT_ACCEPTABLE	/
409	CONFLICT	Register the email repeatedly.
415	UNSUPPORTED_MEDIA_TYPE	/
444	PROTOCOL_NOT_MATCH	A lack of the required parameter in the interface.
500	INTERNAL_ERROR	/
502	GATEWAY_ERROR	/
503	SERVICE_UNAVAILABLE	/
504	GATEWAY_TIMEOUT	/

2. Server Interface

2.1. Adding a Server

Request URL:

- POST：/api/open/v1/server/add

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
serverName	String	Yes	20	The server name.
url	String	Yes	512	The server URL.
authName	String	No	32	The authentication name.
password	String	No	32	The authentication password.
certificateUrl	String	No	--	The certificate URL.

- Example Request:

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

Example of Successful Response:

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

2.2. Viewing the Server by Paging

Request URL:

- POST：/api/open/v1/server/list

Request Parameter:

- **Body parameter**

Parameter	Type	Required	Length	Description
key	String	No	-	The server name or URL, used for fuzzy match.
skip	Integer	No	-	The skipped records, and it defaults to 0.
limit	Integer	No	-	The maximum number of the obtained records per paging.
autoCount	boolean	No	-	The total number is accounted automatically or not, and it defaults to false.

- Example Request:

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

Example of Successful Response:

```
{
  "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. Viewing the Detailed Information of the Server

Request URL:

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

Request Parameter:

- **Query parameter:**

Parameter	Type	Required	Length	Description
id	String	Yes	32	The server ID.

Example of Successful Response:

```
  {
    "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. Detecting Whether or not the Server Name Exsits

Request URL:

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

Request Parameter:

- **Query parameter:**

Parameter	Type	Required	Length	Description
serverName	String	Yes	20	The server name.

Example of Successful Response:

- The server name exists.

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

- The server name does not exist.

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

2.5. Editing a Server

Request URL:

- POST：/api/open/v1/server/edit

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
id	String	Yes	32	The server ID.
serverName	String	Yes	20	The server name.
url	String	Yes	512	The server URL.
authName	String	No	32	The authentication name.
password	String	No	32	The authentication password.
certificateUrl	String	No	--	The certificate URL.

- Example Request:

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

Example of Successful Response:

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

2.6. Deleting a Server

Request URL:

- POST：/api/open/v1/server/delete

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
ids	List	Yes	-	The server IDs to be deleted.

- Example Request:

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

Example of Successful Response:

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

3. Device Interface

3.1. Adding a Batch of Devices

Request URL:

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

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
macs	List	Yes	--	The list of the MAC addresses.
serverId	String	No	32	The server ID.
uniqueServerUrl	String	No	512	The URL of the autop server, and it is set for certain devices. After you set a URL for this parameter, the phone will go to the URL firstly when sending the RPS request. If you do not set a URL, the phone will go to the linked server URL.
remark	String	No	256	The remark.
authName	String	No	32	The authentication name.
password	String	No	32	The authentication password.

- Example Request:

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

Example of Successful Response:

```
{
  "ret": 1,
  "data": [
      {
        "id": "82c2df80fca745d3b7a1405b02bd55a8",
        "mac": "aa000a0000aa",
        "serverId": "ba7c7b13ed114a5fa6f12063ea9dff41",
        "serverName": "SeakeerTestServer",
        "remark": "SeakeerDevice",
        "uniqueServerUrl": "https://www.yealink.com",
        "authName": "Seakeer"
      },
      {...}
  ],
  "error": null
}
```

3.2. Viewing the Device Information by Paging

Request URL:

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

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
key	String	No	-	The key words.
status	String	No	-	The status mark, one of bound or unbound.
skip	Integer	No	-	The skipped records, and it defaults to 0.
limit	Integer	No	-	The maximum number of the obtained records per paging.
autoCount	Boolean	No	-	The total number is accounted automatically or not, and it defaults to false.

- Example Request:

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

Example of Successful Response:

```
{
  "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": "**#***"
      }
    ]
  },
  "error": null
}
```

3.3. Viewing the Detailed Inforamtion of the Device

Request URL:

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

Request Parameter:

- **Query parameter:**

Parameter	Type	Required	Length	Description
id	String	Yes	32	The device ID.

Example of Successful Response:

```
{
  "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. Detecting Whether or not the Device Is Registered

Request URL:

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

Request Parameter:

- **Query parameter:**

Parameter	Type	Required	Length	Description
mac	String	Yes	17	The MAC address.

Example of Successful Response:

- The device cannot be found on the paltform

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

- The device is unregistered.

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

- The device is registered by other users.

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

- The device is registered.

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

3.5. Detecting Whether or not the Device Exists

Request URL:

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

Request Parameter:

- **Query parameter:**

Parameter	Type	Required	Length	Description
mac	String	Yes	17	The MAC address.

Example of Successful Response:

- The device does not exsit.

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

- The device exsits and belongs to you.

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

- The device exsits but belongs to others.

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

3.6. Viewing the Server List

Request URL:

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

Request Parameter:

> None

Example of Successful Response:

```
{
  "ret": 2,
  "data": [
    {
      "id": "b25ac1016caf416a90d5ca1ee438153a",
      "serverName": "SeakeerServerTest"
    },
    {
      "id": "ba7c7b13ed114a5fa6f12063ea9dff41",
      "serverName": "SeakeerTestServer"
    },
  ],
  "error": null
}
```

3.7. Editing the Device

Request URL:

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

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
id	String	Yes	32	The device ID.
serverId	String	No	32	The server ID.
uniqueServerUrl	String	No	512	The URL of the autop server, set for certain devices, has a higher priority than the linked server URL.
remark	String	No	256	The remark.
authName	String	No	32	The authentication name.
password	String	No	32	The authentication password.

- Example Request:

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

Example of Successful Response:

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

3.8. Migrating a Bacth of Devices

Request URL:

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

Request Parameter:

- **Body parameter:**

Parameter	Type	Required	Length	Description
ids	List	Yes	-	The list of the device ID.
serverId	String	Yes	32	The server ID.

- Example Request:

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

Example of Successful Response:

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

3.9. Deleting a Bacth of Devices

Request URL:

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

Request Parameter:

Parameter	Type	Required	Length	Description
ids	List	Yes	-	The list of the device ID.

- Example Request:

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

Example of Successful Response:

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

4. Notes

4.1. The Authentication Name and the Password

- The authentication name and the password should appear in pairs.
- When editing the server and the device, you cannot set the password as below, otherwise, the password is invalid.

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

4.2. The Default Handling Method and Prompt of the Device Interface

- If uniqueServerUrl="" or uniqueServerUrl=“ ”, url.invalid is returned, which means the uniqueServerUrl is invalid.
- If you enter the parameter, for example, serverId="" or serverId=" " , the serverId is null or the serverName is null by default.

4.3. The Device Is Added by Other Enterprises

- If the following result is returned, it means that the MAC address you add has been used by other enterprises.

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

You can do one of the following:

- Check whether or not the MAC address you entered is correct.
- If you want to add the MAC address, you can file an appeal on https://ticket.yealink.com and Yealink will review the appeal. To file an appeal, you need provide the materials below:

(1)The device MAC address and the SN

(2) The enterprise name

(3)The device firmware version

(4)The login account

4.4. Server Name

The server name should be unique at the entire system level. That is, if the server name A is used by an enterprise (user), the server name A cannot be used again. Otherwise, you can see the error message: server.name.existed, indicating that the server name already exists (It may be used by your enterprise or other enterprise).

5. The Error and Its Solution

Error	Description	Solution
service.common.token.unauthorized	Authentication fails.	Check the accesskey and the signature.
Content.MD5.not.null	Content-MD5 is required.	Add the Content-MD5 when calling the interface of the Body parameter.
Content.MD5.invalid	Content-MD5 is invalid.	Re-calculate the Content-MD5.
accesskey.id.invalid	AccessKey ID is invalid.	Check whether the AccessKey ID is valid or re-obtain the AccessKey ID.
request.header.invalid	Signature is invalid.	Re-calculate the signature.
request.header.invalid	Error Header	Refer to Section 1.3.2.
request.replay	The request replays.	Refer to Section 1.3.6.
url.invalid	The URL format is invalid.	The supported URLs are http://...;https://...;ftp://...;tftp://...
url.too.long	The length of the URL exceeds 512 characters.	The length of the URL cannot exceed 512 characters.
id.repeated	There are repeated items in the entered ID list.	Remove the repeated items.
server.id.invalid	The server ID is invalid.	Use the valid server ID.
auth.name.too.long	The length of the authentication name exceeds 32 characters.	The length of the authentication name cannot exceed 32 characters.
auth.name.password.must.be.couple	The authentication name and the password should appear in pairs.	Enter both the authentication name and the password, or do not enter both of them.
auth.name.or.password.inputted.not.empty	The username and the password you enter cannot be an empty string.	The username and the password you enter cannot be “”, or “ ”, “ ”...
Ids.not.empty	The ID list cannot be empty.	Enter the ID.
device.operate.forbidden	Limited device operational permission.	The device belongs to other enterprises, refer to Section 4.3 for more information.
device.not.found	The device cannot be found.	Check the Device ID or the MAC address you entered.

Error	Description	Solution
device.mac.existed	The device already exists.	The device is added, if you want to edit the information, you can call the interface of editing the device, in Section 3.7.
device.mac.added.by.other	The device is added by other enterprises.	Refer to Section 4.3.
device.mac.invalid	The format of the MAC address is invalid.	The supported formats are as below: 001565121212 00 15 65 12 12 12 00-15-65-12-12-12 00:15:65:12:12:12
device.mac.needed	The MAC address is required.	Enter the MAC address.
device.mac.repeated	There are repeated items in the list of the MAC addresses.	Remove the repeated items.
device.remark.too.long	The length of the remark exceeds 256 characters.	The length of the remark cannot exceed 256 characters.
device.macs.contains.empty.item	There are empty items in the list of the MAC addresses.	Remove the empty items.
server.name.not.blank	The server name cannot be empty.	Enter the server name.
server.url.not.blank	The server URL cannot be empty.	Enter the server URL.
server.name.existed	The server name already exists.	Refer to Section 4.4.
server.not.found	The server cannot be found.	Check the server ID or the name you entered.
server.operate.forbidden	Limited server operational permission.	Use your own server.
server.name.too.long	The length of the server name exceeds 256 characters.	The length of the server name cannot exceed 256 characters.

6. Appendix

This part provides the example code (Java) about calculating the signature and calling the interface, only for reference.

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