
Table of Contents

简介	1.1
告警API	1.2
增加告警	1.2.1
更新告警	1.2.2
删除告警	1.2.3
指标数据API	1.3
推送指标数据	1.3.1
资产API	1.4
增加资产	1.4.1
更新资产	1.4.2
删除资产	1.4.3
附录一	1.5

简介

文档中代码是Javascript, Ajax访问API, \$代表JQuery, demo中的代码是放在3D机房系统发布包的resource/public下访问

3D机房系统提供了标准的（RESTfulwebservice）开放式的应用程序接口（API），供第三方调用 RESTfulwebservice：通过统一资源标示符（URL）来识别和定位资源，并且针对这些资源而执行的操作是通过HTTP规范定义。其核心操作只有GET、PUT、POST、DELETE。

接口原则：

1. 允许数据使用者，通过标准的接口访问3D机房系统数据
2. 允许数据提供者，通过标准的接口增加数据到3D机房系统

3D机房系统包含资产管理模块，动力环境监控模块功能模块。

告警API

告警API包含：增加、更新、删除、查询。模拟需求场景，对4种API进行说明。

需求场景：编号为rack84_E01设备，可能产生三种告警：温度告警，网络告警，物理告警

告警字段列表

字段	类型	必需	描述
realId	string	是	告警的唯一标识，与接口中alarmID对应
dataId	string	是	发生告警的设备编号，与接口中deviceID对应
alarmTypeId	string	是	对应告警类型中的编号，参考告警类型模块，与接口中alarmType对应
level	string	是	对应告警级别中的编号，参考告警级别模块
devIp	string		发生告警的IP
time	date		告警发生时间
ackTime	date		告警确认时间
ackNotice	string		确认告警时备注
description	string		描述
client	Object		扩展的字段

增加告警

POST `http://{serverhost:serverport}/api/monitor/alarms`

Body

一个或多个告警实例

Accept: application/json

```
{  
    module: string, 模块名, 保留字段, 缺省值“PEMS”,  
    data: [  
        {  
            alarmId: string, required, 告警的唯一标识,  
            deviceId : string, required, 发生告警的设备,  
            alarmType: string, required, 对应告警类型中的编号, 参考告警类型模块,  
            level: string, required, 对应告警级别中的编号, 参考告警级别模块,  
            description: string, 告警描述,  
            time: date, 告警发生时间,  
            ackTime: date, 告警确认时间,  
            ackNotice: string, 确认告警时备注,  
            devIp: string, 发生告警的IP,  
            client: {} Object, 扩展的字段  
        }  
    ]  
}
```

Returns

异步操作, 返回结果仅表示推送到3D机房系统后端

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

```
var data = {  
    module: "PEMS",  
    data: [
```

增加告警

```
{  
    alarmId: '123',  
    deviceId : 'rack84_E01',  
    alarmType: '温度告警',  
    level: 'critical',  
    description: '湿度过高',  
    time: new Date(),  
    devIp:'127.0.0.1',  
    client: {'告警值':70}  
,  
{  
    alarmId: '234',  
    deviceId : 'rack84_E01',  
    alarmType: '网络告警',  
    level: 'critical',  
    description: '网络端口',  
    time: new Date(),  
    devIp:'127.0.0.1',  
    client: {告警值:'disconnect'}  
,  
}  
]  
};  
$.post('/api/monitor/alarms', data, function(data, textStatus, xhr) {  
    /*optional stuff to do after success */  
});
```

更新告警

PUT

http://{serverhost:serverport}/api/monitor/alarm/ [id]

Path parameters

name	type	description
id	string	推送方的告警的唯一标识

Body

要更新的告警属性，通常在确认告警或修改告警状态时，通过更新方法

Accept: application/json

```
{  
    deviceId : string, required, 发生告警的设备,  
    alarmType: string, required, 对应告警类型中的编号, 参考告警类型模块,  
    level: string, required, 对应告警级别中的编号, 参考告警级别模块,  
    description: string, 告警描述,  
    time: date, 告警发生时间,  
    ack_time: date, 告警确认时间,  
    ack_notice: string, 确认告警时备注,  
    devIp: string, 发生告警的IP,  
    client: {} Object, 扩展的字段  
}
```

Returns

异步操作，返回结果仅表示推送到3D机房系统后端

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

以增加告警章节demo中添加的编号为123的告警为例

```
var data = {
    "ackTime": "2016-12-30",
    "ackNotice": "已转交其他同事处理",
    "client": {
        "确认时间": "2016-12-30 9",
        "告警确认人": "admin",
        "确认内容": "已转交其他同事处理"
    },
    "status": "ack"
}
$.ajax({
    url: '/api/monitor/alarm/123',
    type: 'PUT',
    data: data,
    success: function(result) {
        // Do something with the result
    }
});
```

删除告警

POST

`http://{serverhost:serverport}/api/monitor/alarm/ [id]`

Path parameters

name	type	description
id	string	推送方的告警的唯一标识

Body

无内容

Returns

异步操作，返回结果仅表示推送到3D机房系统后端

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

以增加告警章节demo中添加的编号为123的告警为例

```
$.ajax({
  url: '/api/monitor/alarm/123',
  type: 'DELETE',
  success: function(result) {
    // Do something with the result
  }
});
```

指标数据API

指标数据是指传感器或设备上的监控数据，例如温湿度、电流电压、各种开关量等数据。3D机房系统用户可以在系统查看此类数据，并且实时变化。

与3D机房系统进行数据交互有两种方式，过程如下：

- 动环系统 -----推-----> 3D机房系统后端 -----推-----> 3D机房系统前端 ----->
动态刷新
- 动环系统 <-----拉----- 3D机房系统后端 -----推-----> 3D机房系统前端 ----->
动态刷新

推送指标数据

POST `http://{serverhost:serverport}/api/monitor/data`

Body

一个或多个业务对象指标数据， deviceId作为一个对象，由其指标属性和值组成键值对

Accept: application/json

```
{  
    module: string, 模块名, 保留字段, 缺省值“PEMS”,  
    data: [  
        'deviceId1': {  
            "property1": value1,  
            "property2": value2,  
            "property3": value3,  
            ... ...  
        },  
        'deviceId2': {  
            "property1": value1,  
            "property2": value2,  
            "property3": value3,  
            ... ...  
        }  
    ]  
}
```

- **deviceId**: 指推送的业务对象唯一标识或资产对象唯一标识，确切的来说都称为业务对象唯一标示，只是在特定的情况直接将资产对象看作为业务对象。更多细节参考业务对象模块
 - **property**: 指业务对象的属性，也就是指标属性，
 - **value**: 指对应的指标属性值

Returns

异步操作，返回结果仅表示推送到3D机房系统后端

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

```
var getData = function(){
    return {
        module: "PEMS",
        data: {
            'b8f7th02': {
                'humidity':Math.floor(Math.random()*10),
                'temperature':Math.floor(Math.random()*10)
            },
            'b8r701c06hr01': {
                'humidity':Math.floor(Math.random()*10),
                'temperature':Math.floor(Math.random()*10),
                "网络":{
                    "状态":"接通",
                    "速度":"100m/s"
                }
            }
        }
    };
}
var data = getData();
$.post('api/monitor/data', data, function(data, textStatus, xhr) {
    /*optional stuff to do after success */
});
```

资产API

对于资产的管理，大部分的资产数据在初始化的系统的时候，会将资产数据整理导入系统。对于系统上线后，对于少量的资产操作可以使用资产API，通过资产API对资产进行操作可以实时同步到前端浏览器

资产字段列表

字段	类型	必需	描述
id	string	是	编号
name	string	是	名称
description	string		描述
position	string		参考附录一：location、position、空间规划
position2d	string		2D中的物理坐标
rotation	string		表示资产对象的旋转值，{x:0,y:0,z:0}，x、y、z分别表示三个方向上的旋转角度
location	string		参考附录一：location、position、空间规划
parentId	string	是	父对象，例如设备的父对象为机柜
dataTypeld	string	是	资产模型编号
weight	int		资产重量

增加资产

POST `http://{serverhost:serverport}/api/monitor/asset`

Body

一个或多个资产实例

Accept: application/json

```
{  
  data: [  
    {  
      id: string, required, 编号,  
      name : string, required, 名称,  
      description: string, 描述,  
      position: string, 参考附录一: location、position、空间规划,  
      position2d: string, 2D中的物理坐标,  
      rotation: string, 表示资产对象的旋转值, {x:0,y:0,z:0}, x、y、z分别表示三个方向上的  
      旋转角度,  
      location: string, 参考附录一: location、position、空间规划,  
      parentId: string, required, 父对象, 例如设备的父对象为机柜,  
      dataTypeId: string, required, 资产模型编号,  
      weight: int, 资产重量  
    }  
  ]  
}
```

Returns

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

```
var data = {  
  module: "PEMS",  
  data: [  
    {  
      id: 'b8r201c04r23e01',  
      name: 'PEMS',  
      description: 'PEMS模块',  
      position: 'location',  
      position2d: '2D中的物理坐标',  
      rotation: '0,0,0',  
      location: 'location',  
      parentId: '0',  
      dataTypeId: '1',  
      weight: 100  
    }  
  ]  
}
```

增加资产

```
        name: 'b8r201c04r23e01',
        location: {"y":28,"z":"pos_pos"},
        parentId: 'b8r201c04r23',
        dataTypeId: 'equipment2'
    }
]
};

$.post('/api/monitor/alarms', data, function(data, textStatus, xhr) {
    /*optional stuff to do after success */
});
```

更新资产

PUT

http://{serverhost:serverport}/api/monitor/asset/ [id]

Path parameters

name	type	description
id	string	资产的唯一标识

Body

要更新的告警属性，通常在确认告警或修改告警状态时，通过更新方法

Accept: application/json

```
{  
    name : string, required, 名称,  
    description: string, 描述,  
    position: string, 参考附录一: location、position、空间规划,  
    position2d: string, 2D中的物理坐标,  
    rotation: string, 表示资产对象的旋转值, {x:0,y:0,z:0}, x、y、z分别表示三个方向上的旋转角度,  
    location: string, 参考附录一: location、position、空间规划,  
    parentId: string, required, 父对象, 例如设备的父对象为机柜,  
    dataTypeId: string, required, 资产模型编号,  
    weight: int, 资产重量  
}
```

Returns

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

以增加资产章节demo中添加的编号为b8r201c04r23e01的资产为例

```
var data = {
```

更新资产

```
name: 'b8r201c04r23e01-update',
description: 'b8r201c04r23e01-update',
location: {"y":10,"z":"pos_pos"},
parentId: 'b8r201c04r23',
dataTypeId: 'equipment2'
}
$.ajax({
  url: '/api/monitor/asset/b8r201c04r23e01',
  type: 'PUT',
  data: data,
  success: function(result) {
    // Do something with the result
  }
});
```

删除资产

POST

`http://{serverhost:serverport}/api/monitor/asset/ [id]`

Path parameters

name	type	description
id	string	资产的唯一标识

Body

无内容

Returns

Content-Type: application/json

```
{value:"success", error:''}
```

Demo

以增加资产章节demo中添加的编号为b8r201c04r23e01的资产为例

```
$.ajax({
  url: '/api/monitor/asset/b8r201c04r23e01',
  type: 'DELETE',
  success: function(result) {

  }
});
```

附录一

location、position、空间规划

物理坐标 – position

任意一个3D对象在三维直角坐标系中显示时，必定会有物理坐标信息，反应为投射到每个轴上的位置。在3D机房系统中，资产对象是一个3D对象，该对象有一个position属性（position有三个值，分别是X, Y, Z），代表在空间直角坐标系中的三个轴上的位置。

逻辑坐标 – location

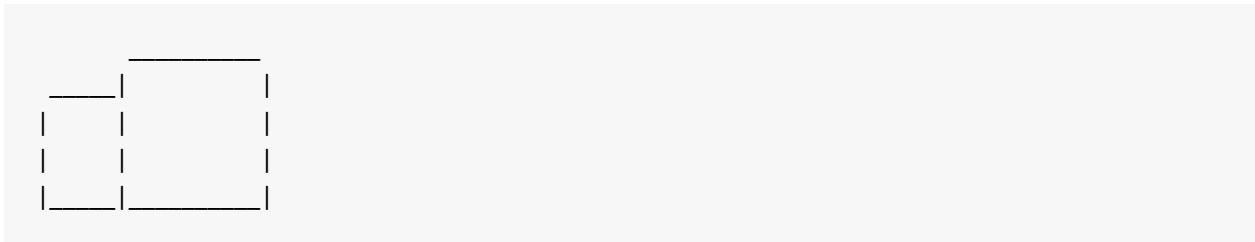
通过前面介绍的position属性，我们可以任意设定一个3D对象的位置信息，然而实际生活中，我们的被管对象往往是有规划的摆放。例如机房中可以摆4行10列机柜、机柜中可以摆下42个的1U的服务器。这些数据决定了空间坐标中的显示位置，然后又没有跟空间坐标的直接关联。在3D机房系统中这种逻辑坐标称为location。系统会将逻辑坐标location计算出一个物理坐标position，用来摆放显示。那3D机房系统是怎么计算出的position的呢？

我们还需要引入一个概念：空间规划，例如前面提到的机房里面可以摆放4行10列，机柜可以摆放42个1U服务器。在资产模型中的size和childrenSize属性表示空间规划概念，后面会详细介绍。例如上面的举例，机房的childrenSize值是{"x": 10, "z": 4}，42U的机柜的size值是{"x": 1, "z": 1}，它的childrenSize值是{y:42}。到此已经很清楚计算过程了，把机房外观尺寸（1000, 1000, 1000）的x方向分成10等份，每一份是100，z方向分成4等份，每一份是250来计算机柜的位置，例如机柜的location是（2, 0, 1），计算出的position值为（1002, 0, 1250）。那么设备的位置就是把机柜的y（高度）除了42，来计算单个设备的位置，例如机柜高度是H，一个1U的设备摆放在4U的位置，那么postion属性的y值等于H / 42 * 4。房间四周可能需要空留部分通道，不能用来摆放机柜、设备的x和z方向因为跟机柜对齐没有设置值，具体会在后面的内容介绍。

location属性介绍：x, y, z分别表示3个方位的布局，支持数字和对齐方式
(['posneg', 'center_neg', 'neg_neg', 'center', 'pos_pos',
, 'center_pos', 'neg_pos']) 两种方式。对齐方式说明：“前面对应的是自身，'_'后面对应的是父亲

如以x轴为例，即x:'pos_neg'...共七中情况

1. pos_neg:表示的是自己在x轴最大值处于父亲在x轴最小值处对齐；
2. center_neg:表示的是自己在x轴的中心点和父亲在x轴最小处值对齐；
3. neg_neg:表示的是自己在X轴最小处和父亲在x轴最小处对齐；
4. center:自己在x轴的中心点和父亲在x轴的中心点对齐；
5. pos_pos:自己在x轴的最大值和父亲在x轴的最大值处对齐；
6. center_pos:自己在x轴的中心点和父亲在x轴的最大值处对齐；
7. neg_pos:自己在x轴的最小处和父亲在x轴的最大值对齐； 例如机柜的location位置是(2, 0, 1) , x = 2 表示第二列, z = 1 表示在第一排, y 值"neg_pos" 表示一种对齐方式, 表示机柜的最下面和机房模型的最上面对齐。如下图：前面的矩形代表自己，后面的代表父亲，并且向右代表正方向，那么下面这种情况就是：“pos_neg”



空间规则

在上面的介绍中引入了空间规划的概念，空间规划的本质是父对象和子对象之间的物理位置重叠，本着更符合人机工程思维，将物理位置抽象出来，将物理单位转换成逻辑单位，简单理解为将空间或平面按最小单位划分成很多小格子，一个子对象最少要占用一个格子或整数倍。例如前面的房间尺寸是 (1000, 0, 1000) 柜子的尺寸是 (100, 100, 250) ，那么房间抽象出来的容积为 (10, 0, 4) ，即可以摆放4行10列柜子，容积是 $4 * 10 = 40$ ，如果有四个柜子的location是 (2, 0, 1) 、 (3, 0, 1) 、 (4, 0, 1) 、 (5, 0, 1) ，那么房间利用率就是 $4 / 40$ (房间容积) * 100% = 10%，剩余空间率为 90%。包括在计算整个机房内机柜的空间占用情况时，将会更加直观给出结果。

空间规划相关的两个属性：size和childrenSize。

size：表示自身所占的位置，属性包括x、y、z，分别表示三个方向的尺寸。例如2U的服务器，需要占用2个1U的高度。那么值为： {y: 2} 例如

```
var size = {"x": 1,"z" : 1};
```

childrenSize：属性包括：x、y、z、xPadding、yPadding、zPadding。其中x、y和z表示其中x、y和z表示可以容纳子对象的数量，xPadding、yPadding和zPadding的值是一个二维数组，默认是 [0, 0] ，表示三个方向相需要扣除的间隙。例如42U的机柜，有42个U的高度，如果装1U的设备可以容纳42个，如果装2U的设备则只能容纳21个。那么值为： {y: 42}

```
var childrenSize={  
    y:42,  
    yPadding:[5.545,5.545],  
    zPadding:[-0.5,-0.5],  
    xPadding:[5.545,5.545]  
};
```

空间规划在计算空间利用率等指标时，会根据size和childrenSize来计算剩余空间和已经使用的空间。