servasoftware

TWaver® Make

开发手册

Version 1.0

Jan 2016
Serva Software
info@servasoftware.com
http://www.servasoftware.com
PO Box 8143, Wichita Falls, Texas, USA 76307

产品定义

TWaver Make（以下简称make）是TWaver产品家族中的一个独立产品，是一个基于JavaScript的开发框架和模板库。具体包括：

- make开发框架：基于JavaScript的开发框架，用于把各类资源（代码片段、各种数据资源、2D/3D模型等）用统一的代码框架进行规范、定义、封装、调用，实现各种可视化数据资源的模版化，达到快速复用；
- make模板库：基于make框架预定义的海量资源模板库，包括基于TWaver 2D、3D技术的各行业模型数据，供开发者直接使用；

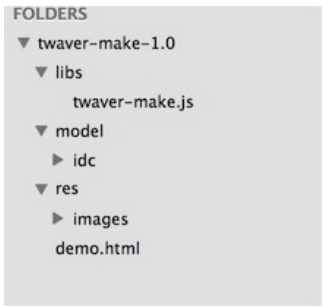
Make最典型的应用是基于TWaver HTML5的各类2D、3D对象库和模板库。Make通过定义模型库和使用内置模板库来大幅简化了对底层2D、3D引擎的使用难度，可以让开发者快速创建高质量、专业的行业可视化应用场景。

> **提示：**如果把搭建系统比喻成盖大楼，TWaver 2D和3D就像沙土和水泥，而TWaver Make的就像是各种标准的“砖头”。盖房子的过程，也就变成了“搭积木”的过程。盖大楼，也不再需要从挖土烧砖开始，让设计师更多的精力专注在建筑结构和功能设计上。

如何使用

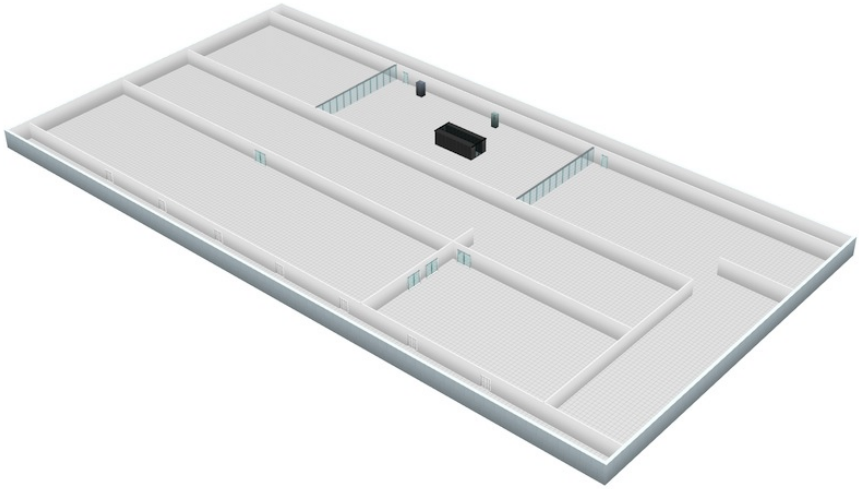
创建工程

首先需要到twaver官网申请下载TWaver Make的试用包，下载下来的是一个zip包，拿到这个包解压后的目录如下：



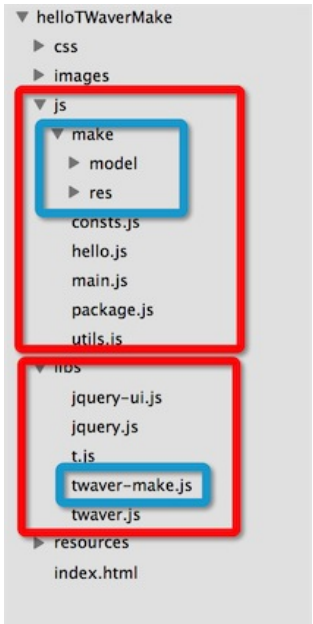
在libs目录下的twaver-make.js就是要引入到项目中的库文件，注意twaver-make.js是依赖于twaver.js和t.js，可以到这里下载这两个包；model目录下提供的就是模型库，目前给出了部分数据机房的模型库，若需要更多的模型库可联系我们；res目录下放置的是模型库依赖的资源文件；demo.html是运行的例子。

了解了TWaver Make包结构之后，就需要知道如何运行这个工程，在demo.html页面中给出了部分模型的例子，注意demo.html文件中还引入了twaver.js和t.js，这两个包需要另外申请，申请了这两个包之后需要放置到libs目录下，就可以将整个工程发布到服务器上，运行效果如下：



几行代码即可创建出房间，通道，机柜，空调，UPS这些模型。

这是直接运行TWaver Make，如果在自己的工程中该如何使用？首先创建新的工程，然后将twaver-make.js包引入到工程中，twaver-make.js文件可放置在任意目录下。引入了js文件之后，还需要将其他的文件(除了libs目录和demo.html文件)一并引入到工程中，需要注意的是要保持包的目录结构不变。下图给出了一个工程结构的示例：



这个项目中把twaver-make整个模型目录(这里重命名为make，目录名可任意更改)copy到js目录下，和工程中用到的其他js文件放在一起。twaver-make.js文件放在了libs目录下，和其他第三方的库文件放在一起。然后在html文件中需要指定这些路径。

```
<!DocType html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>3d模型</title>
    <script src="./libs/t.js"></script>
    <script src="./libs/twaver.js"></script>
    <script src="./libs/twaver-make.js"></script>
    <script>
        var network;

        function init(htmlElementId){
            make.Default.path = './js/make/';

            network= new mono.Network3D();
            var box = network.getDataBox();
            var camera = new mono.PerspectiveCamera(30, 1.5, 30, 30000);
            camera.setPosition(812,2373,1473);
            camera.lookAt(new mono.Vec3(147,997,-91));
            network.setCamera(camera);
```

第一个红框的代码大家都知道。第二个红框的代码make.Default.path = ‘./js/make/’;是指定模型库的相对目录，如果没有指定，那么内置的模型将找不到资源图片。

注册模型

接下来就来介绍代码中如何使用make：

```
//注册资源
var parameters = {
 category : "文本",
 type : "string",
 modelDefaultParameters : {},
 name : "",
 description:"",
};
make.Default.register('my_name', '赛瓦软件',parameters);

//....

//加载并使用资源
var myName = make.Default.load('my_name');
```

以上就是make最简单的产品定义和使用方法。我们只需要了解一个类make.Default和它的两个方法register和load就可以使用make框架了。

register和load的方法原型如下：

```
/**
 * id:注册资源的ID号
 * creator:注册资源的生成器
 * parameters:注册资源的参数
 */
make.Default.register: function(id, creator, parameters);

/**
 * ids:加载资源的ID或ID数组
 * parameters:加载资源的参数
 * callback:回掉方法
 */
make.Default.load: function(ids, parameters, callback);
```

注册2D对象模版

下面代码用于注册一个TWaver 2D节点：

```
make.Default.register('my_node', function(){
  var node = new twaver.Node();
  node.setSize(200, 100);
  node.setName('网元');
  node.setToolTip('网元');

  return node;
});

//....

var node = make.Default.load('my_node');
```

```
network.getDataBox().addByDescendant(node);
```

注册3D对象模版

下面代码用于注册一个mono（TWaver 3D）立方体对象：

```
make.Default.register('my_cube', function(){
  var cube = new mono.Cube(200, 100, 300);
  cube.setPosition(0, cube.getHeight()/2, 0);
  cube.s({
          'm.type': 'phong',
          'm.color': '#845527',
          'm.ambient': '#845527',
          'left.m.lightmap.image': 'inside_lightmap.jpg',
          'right.m.lightmap.image': 'outside_lightmap.jpg',
          'front.m.lightmap.image': 'outside_lightmap.jpg',
          'back.m.lightmap.image': 'inside_lightmap.jpg',
          'top.m.color': '#845527',
          'top.m.ambient': '#845527',
          'bottom.m.color': '#845527',
          'bottom.m.ambient': '#845527',
  });

  return node;
});

//....

var node = make.Default.load('my_cube');
network3d.getDataBox().addByDescendant(node);
```

使用parameters指定注册资源的参数

在注册资源的时候，可以指定注册的资源（或者资源creator）的一些参数，可以在register函数的第三个参数中指定。

```
make.Default.register('rack', function(json){
   ...
},{
  category : "机柜",
  type:"mono.Element",
  name : "机柜模型",
  description : "机柜模型，一般支持42U和47U，长宽高可以指定，默认高度42U，默认深度660",
  async : false,
  icon : 'icons/rack.png',
  modelDefaultParameters : {
     width : {name :"宽", value : 1000, type : "string"},
     height : {name : "高",value : "42U", type:"string"},
  },

  someOther:"someOther"
})
```

注册之后，可以通过make.Default.getParamters('rack')来获取相关模型的参数。 需要注意的是，category,type, modelDefaultParameters,async,name,description 是系统内置的一些标准（属性）。category 可以对模型进行分类，type指定了模型的类型，name和description可以指定模型描述，modelDefaultParameters是针对creator的，相当于creator的默认参数，icon指定了模型的缩略图。 如果指定了category，make将会对模型进行分类，而通过make.Default.getCategories() 获取所有分类，make.Default.getCreatorsForCategory 可以获取一个分类下面的所有模型；其他的内置属性，可以通过make.Default的getXXX方法获取，比如name 可以通过make.Default.getName('rack');来获取。 async指定了模型的是异步还是同步返回的，默认是同步，也就是该参数不指定就是false，用户可以通过 make.Default. isAsync('rack')来获取资源的异步同步属性，该属性的作用是，用户可以通过此属性来决定加载模型的是通过异步的加载还是同步的方式加载，代码片段如下：

```
if(make.Default.isAsync(id)){
 // 执行异步加载方法
}else{
 // 执行同步加载方法
}
```

而我们的另外一个产品，设备编辑器，将会直接使用到make指定的category,name,description,modelDefaultParameters等参数，也就是说设备编辑器将会和make无缝集成（具体可以参考以后的编辑器的说明）。

除此之外，用户还可以根据自己的需求，指定自己的属性，比如例子中的someOther属性；该属性可以通过make.Default.getParameters('rack').someOther来获取。

使用box.addByDescendant函数

可以注意到，上面的例子中，无论2D还是3D数据，都会使用DataBox的addByDescendant方法添加数据，而不是add。这是因为，大多数模型库返回的结果并不止是一个2D或3D对象，而是具有父子关系的一批对象组成的拓扑图或物体，而返回的对象往往是这些对象的root根对象。因此，使用addByDescendant函数可以一次性把所有的返回对象加载到DataBox中。

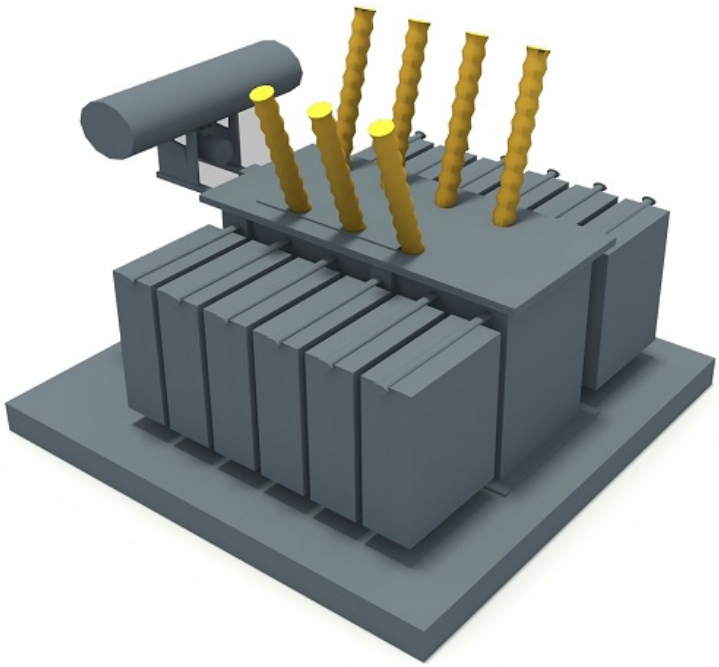
因此，使用addByDescendant总是比使用add更方便、稳妥。推荐总是使用addByDescendant来添加对象到DataBox中。

使用内置模型库

TWaver Make模型库提供了大量高精度、专业的行业预制2D、3D模型库，开发者可以直接使用。

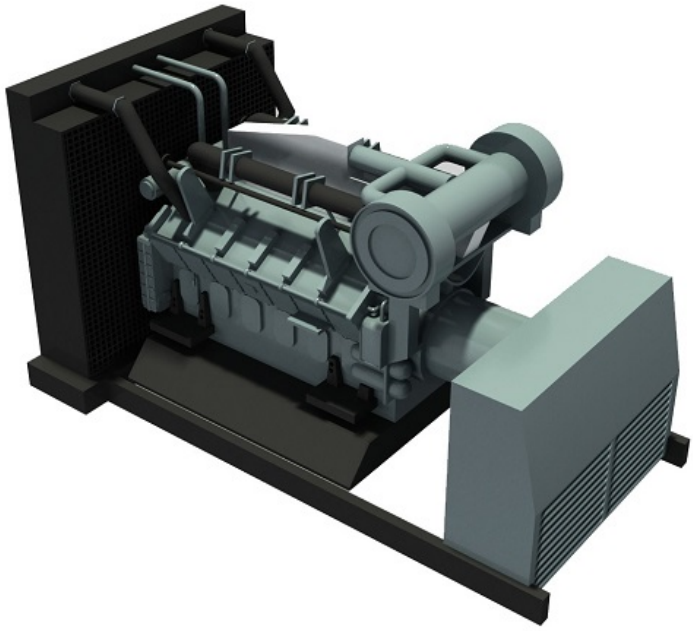
下面代码调用了make中预定义的电力变电站中的变压器模型。已知变压器模型的id为twaver.grid.transformer，则：

```
var transformer = make.Default.load('twaver.grid.transformer');
network3d.getDataBox().addByDescendant(transformer);
```

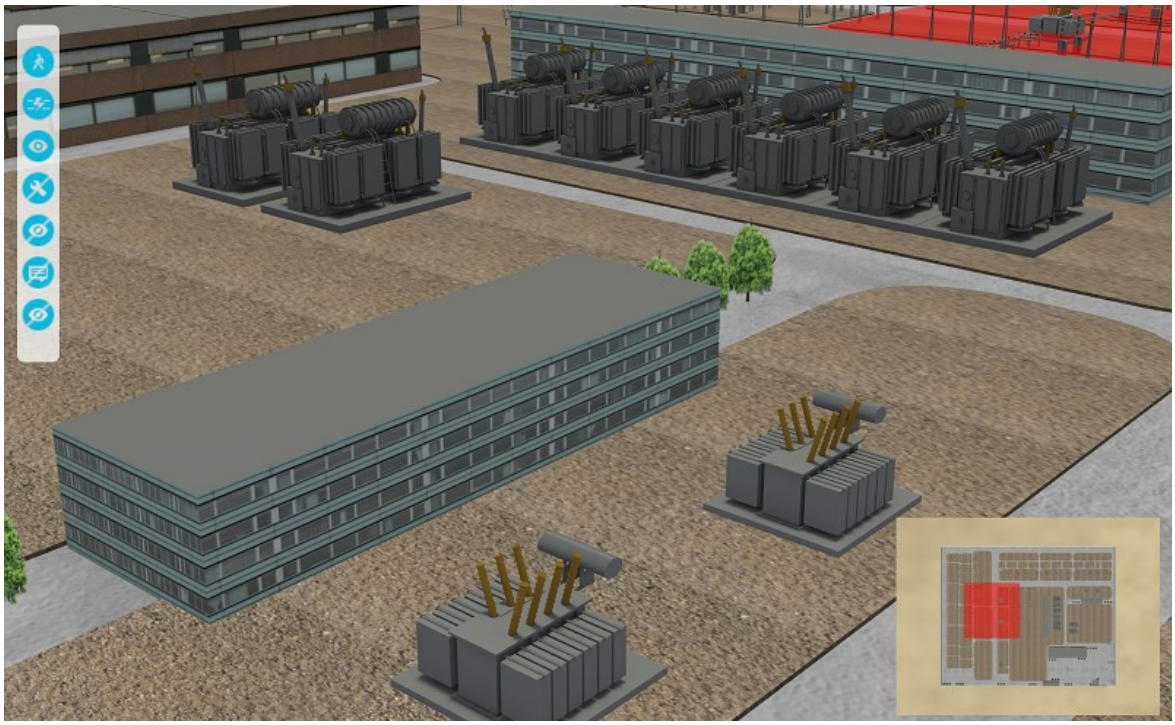


下面代码调用了make中预定义的数据中心发电机模型，其id为twaver.idc.generator，则：

```
var generator= make.Default.load('twaver.idc.generator');
network3d.getDataBox().addByDescendant(generator);
```



可见，使用make预制模板库可以非常快的创建各种复杂的3D模型，快速开发产品。下图中的场景，使用模型库创建，代码不过几百行。



预定义模型库

TWaver Make预定义了大量行业模型库，供开发者直接使用。包括：

- 电信行业
- 电力行业
- 数据中心
- 工业自动化
- 物流与仓储

可以根据具体需要来选择模型库，也可以自行添加自定义模型库。内置模版库参考模型库目录一章。如果您需要定义更多行业模型，请与TWaver团队联系。

基本概念

Make的模型分为定义和使用两部分：

- 通过make.Default.register函数来定义模型
- 通过make.Default.load函数来加载使用模型

每一个模型在定义时，要给一个全局唯一不会重复的id来标识。推荐格式举例：

- twaver.idc.rack
- mycompany.product.id001

只要id不重复即可。Make内置模型均以twaver开头，因此不要使用以twaver开头的id。

定义模型

make.Default.register函数定义如下：

make.Default.register(id, createFunction)

其中，id变量是要注册的模型的id，需保证全局唯一。如果id已经存在，则会覆盖旧的模型注册，并用console.log给出警告信息；createFunction是创建模型的函数，它是定义了如何创建模型，make会调用这个函数来生成具体的函数。createFunction最后必须return模型结果。

下面代码定义了一个最简单的模型：

```
make.Default.register('my_name', function(){
  var node = new twaver.Node();
  node.setSize(200, 100);
  return node;
});
```

直接返回基本类型

register函数的createFunction参数也可以直接使用基本类型，例如字符串、数字等。此时load模型时将直接返回这个数值。

```
//注册资源
make.Default.register('my_name', '赛瓦软件');

//....

//加载并使用资源
var myName = make.Default.load('my_name');
```

```
//myName将等于'赛瓦软件'
```

加载模型

make.Default.load(id, parameters, callback)

其中：

- id：必选。要加载的模型id
- parameters：可选。要传递给模型的额外控制参数
- callback：可选。模型返回时的回调函数，主要用于异步加载时使用

下面的代码直接加载my_node模型并放入DataBox中：

```
var node = make.Default.load('my_node');
network.getDataBox().addByDescendant(node);
```

批量调用多个模型

id参数也可以是多个id的数组，用来批量以此加载多个模型。返回的object则是一个“id-value”结构的对象。

下面代码直接返回了’my_node1’, ‘my_node2’, ‘my_node3’ 三个模型，返回值在object[‘my_node1’], object[‘my_node2’]，object[‘my_node3’]中。

```
var ids = ['my_node1', 'my_node2', 'my_node3'];
var object = make.Default.load(ids);

var object1 = object['my_node1'];
var object2 = object['my_node2'];
var object3 = object['my_node3'];
```

当使用parameters参数时，可以传入一个参数对象，也可以传入参数对象数组。其规则是：

- 如果传入一个参数对象，则这个参数对象会传递给每一个模型；
- 如果传入参数对象数组，则按ids数组的顺序依次传递给每一个模型；

下面的方法把param这个参数对象同时传递给了my_node1、my_node2、my_node3：

```
var param = {
  "width": 200,
  "height": 300,
};
var ids = ['my_node1', 'my_node2', 'my_node3'];
var object = make.Default.load(ids, param);
```

下面的方法则给my_node1、my_node2分别传递了不同的参数，而my_node3则没有传递参数：

```
```

```
var param1 = {
  "width": 200,
  "height": 300,
};
var param2 = {
  "width": 400,
  "height": 500,
};
var ids = ['my_node1', 'my_node2', 'my_node3'];
var object = make.Default.load(ids, [param1, param2]);
```

在批量加载模型的情况下，callback参数的意义依旧不变。批量返回的结果object对象将会被传递给callback回调。

异步返回模型

有时，模型需要从文件、网络等处进行异步加载并创建，需要异步方式完成，无法直接return结果对象。此时，load函数无法直接获得模型结果，只能通过callback来获得。

假设id为'twaver.idc.building_001'这个模型是异步加载，则可以通过下面的代码进行加载：

```
make.Default.load('twaver.idc.building_001', null, function(building){
  network3d.getDataBox().addByDescendant(building);
});
```

make是一个平台，除了拥有丰富的内置模型外，用户也可以扩展自己的模型库。

完全自定义的模型

利用自方法，用户可以完全更加自己的需求定制自己的模型，没有任何限制；缺点是无法利用已经有的模型库进行扩展；对于3D模型开发者，此种方法适用底层开发能力强的用户.如下：

```
make.Default.register('myRack',function(){
    var rack = new mono.Cube(60,220,60);
    rack.s({
       'm.image':'../rack.png'
       'm.front.image' : '../front_rack.png',
    });
},{
  category : "机柜模型"
  type : "mono.Element",
});
```

基于已有模型扩展

利用此方法，用户可以在原有的模型的基础上，定制出自己常用的模型，如下代码,在原本的“twaver.idc.rack”的模型基础上，通过修改参数width,扩展出自己的模型,此种方法定义模型简单，不需要精通太多的底层技术，缺陷是局限于已有的模型库，不过随着make的模型库不断丰富，这种局限性会变得越来越小.

```
make.Default.register('myRack',function(json){
  return make.Default.load('twaver.idc.rack',{width:700});
});
```

基于twaver.combo模型扩展

利用此方法，用户可以在原来模型库的基础上，通过组合的方式，形成一个新的组合体（注意，只是针对3D模型），比如一个楼层模型，其实是用过外墙，内墙，门，窗等组成的，所以一个楼层就可以通过twaver.combo来定义,此方法的参数是一个数组，而数组第一个元素定义的对象将作为组合体的主对象：

```
make.Default.register('room01',function(json){
   var jsonObject = [{"id":"twaver.idc.wall","data":[20,20,6444,20,6444,4488
{"id":"twaver.idc.floor","data":[20,20,6444,20,6444,4488,20,4488,20,20]},
{"id":"twaver.idc.innerWall","data":[616,1668,6086,1668,6086,2518,616,2518,6
];
var object3d = make.Default.load("twaver.combo",jsonObject);
return object3d;
```

```
8  });
```

开发文档 – TWaver Make　模型库目录

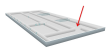
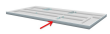
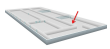
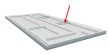
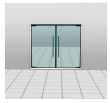
TWaver Make内置了各种行业的模型库，下面表格列出了常见的模型：

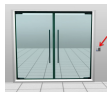
基本模型

ID	名称	图片	描述
twaver.cube	立方体		立方体是make里面提供的最基本的模型，可以用这种模型扩展出各种立方体形状的模型，比如箱子，柱子，盒子等。支持更改长，宽，高，贴图等基本参数
twaver.cylinder	圆柱体		圆柱体是make中提供的最基本模型，支持更改半径，高度的参数。可以更改这些参数扩展出各个圆柱体形状的模型，比如圆形柱子，桌子腿，瓶子，圆筒等

机房数据中心

ID	名称	图片	描述
twaver.idc.rack	机柜		机房最常见的模型。机柜是由机柜的主体和机柜门组成的，机柜门支持动画打开，关闭。 支持42U，47U两种高度。高度更改，模型上还支持常见的长，宽，机柜贴图的设置
twaver.idc.rack1	机柜		同上。区别在于这种类型的机柜加了环境贴图，是另外一种风格的机柜样式
twaver.idc.simpleRack	简单机柜		机柜的简单实现，一个cube加一个贴图，外观和twaver.idc.rack一样，但没有门和内部结构，可用于不关注内部细节时显示
twaver.idc.headerRack	列头柜		机房常见设备之一，列头柜支持修改长，宽，高等基本参数。
twaver.idc.equipment	1U设备		1U的服务器设备。默认是45cm宽度，支持修改贴图和宽，深。 1U=4.445cm
twaver.idc.equipment2	2U设备		同上，区别在于高度和贴图不同。2U= 2*4.445
twaver.idc.equipment3	3U设备		同上，区别在于高度和贴图不同。3U= 3*4.445
twaver.idc.equipment4	4U设备		同上，区别在于高度和贴图不同。4U= 4*4.445
twaver.idc.equipment5	5U设备		同上，区别在于高度和贴图不同。5U= 5*4.445
twaver.idc.equipment6	6U设备		同上，区别在于高度和贴图不同。6U= 6*4.445

			
twaver.idc.equipment7	7U设备		同上，区别在于高度和贴图不同。7U= 7*4.445
twaver.idc.equipment8	8U设备		同上，区别在于高度和贴图不同。8U= 8*4.445
twaver.idc.equipment8-1	8U设备机框		8U机框内部是挖空的，用于放置各种板卡。长，宽，高支持修改，默认是8U高度，可根据实际情况修改成合适的高度
twaver.idc.column	柱子		柱子一般是加在房间外墙上，有些机房会在通道内也会有柱子对象
twaver.idc.floor	地板		地板是和墙，内墙一起使用的。用来创建3d机房的房间轮廓
twaver.idc.wall	外墙		外墙是和内墙，地板一起使用的。用来创建3d机房的房间轮廓
twaver.idc.innerWall	内墙		内墙是和外墙，地板一起使用的。用来创建3d机房的房间轮廓
twaver.idc.glassWall	玻璃墙		玻璃墙是墙的一种，用来做出更美观的房间效果
twaver.idc.window	窗户		窗户一般不单独使用，是加在内墙或外墙模型上，作为这类模型的孩子对象
twaver.idc.door	双开门		门一般不单独使用，是加在内墙或外墙模型上，作为这类模型的孩子对象
twaver.idc.door1	单开门		门一般不单独使用，是加在内墙或外墙模型上，作为这类模型的孩子对象
twaver.idc.area	区域		区域是加在某个房间上的。用来对房间进一步细分。可以指定区域的高度，文字，颜色等属性
twaver.idc.aisle	通道		通道是机房中用于摆放机柜的节能模块，分为单通道和双通道，由通道门和天窗组成。可以设置通道内摆放的机柜数和通道的高度
twaver.idc.simpleAisle	简单通道		同上。区别在于 简单通道只是没有细节，是用几个cube加贴图组成，通道窗户不能打开。主要用于和真实通道之间的切换。
twaver.idc.airCondition	精密空调		精密空调是机房中用于散热，排风的模块，保证机房的恒温恒湿。支持修改长，宽高，贴图基本参数

			
twaver.idc.ups	UPS		用于给单台计算机、计算机网络系统或其它垫子设备如电磁阀、压力变送器等提供稳定、不间断的电力供应，保证机房在突发情况下也能正常运转
twaver.idc.battery	蓄电池		机房蓄电池用于给UPS或其他设备不间断供电，保证机房的稳定运行
twaver.idc.alternator	发电机		发电机也是保证机房正常工作的设备之一，在电停的时候发电机自动启动，电来时候，自动停止工作
twaver.idc.camera	摄像头		机房摄像头，这种类型的摄像头只在某个固定的角度监控视频。支持修改旋转角度
twaver.idc.camera1	摄像头1		360度无死角摄像头，这种类型的摄像头监控的范围更广一些
twaver.idc.ACS	门禁		门禁一般不单独创建，是和门一起使用的，放在门边上的一种控制系统

库房

ID	名称	图片	描述
twaver.wh.storageRack	货架		货架是仓库中用于放置货物的架子，可以设置货物的宽，高，深，层数，和每层之间的间距。
twaver.wh.ware	货箱		货箱是用于存放货物的容器，可以设置颜色和宽，高，深。

了解更多文档，请至 TWaver 在线文档中心：http://doc.servasoft.com/

关于 TWaver 和赛瓦软件

赛瓦软件（Serva Software）成立于 2004 年，是美国赛瓦集团（Serva Group）全资子公司、高新技术企业、“双软”认证企业，主要面向中国及亚洲地区提供产品和技术服务。赛瓦软件总部位于美国 Texas 州 Wichita Falls。

赛瓦软件 TWaver 产品是专业的行业 UI 可视化组件产品，提供丰富的 2D/3D 展示能力，支持各种技术平台。TWaver 广泛应用于电信、电力、金融等行业，可以 2D 或 3D 的方式呈现复杂的网络拓扑数据、地图、设备结构图等。爱立信、思科、惠普、华为、中兴、亿阳、浪潮、南瑞等几百家电信软件企业都是 TWaver 的长期用户。

联系我们

赛瓦软件（上海）有限公司

地址：上海市徐汇区中山西路 2025 号 1921 室
邮编：200235
电话：86-21-64398788
传真：86-21-64395374
官方网址：http://servasoft.com

技术热线

TWaver 技术支持邮箱：tw-service@servasoftware.com
TWaver 技术支持电话：86-21-64398788（转 TWaver 技术支持）
TWaver 技术紧急救援：135-6491-6007（Paul）