饼图配置简介

1，基本属性：

属性	功能
主标题	视图最上方显示的主标题文字
副标题	视图最上方显示的副标题文字

2，数据：

属性	功能
绑定数据	设置饼图是否需要绑定数据
专家模版	设置所使用的专家模版
资源类型	包括：模版，管理域，客户，项目，企业
选择资源方式	指定：指定一个资源，传递参数：通过resourceId来传递的
选择指标	选择对应指标
选择实例	选择对应的实例（默认为无）
取值规则	取区间末位，取区间首位，区间平均值
填充空值	是否自动填充空值补充曲线
统计周期	后台数据统计的周期
统计类型	当资源方式为企业时可用（默认为无）
图例	通过函数设置图例返回值
数据列	通过函数设置数据列返回值

图例的配置方法：

首先要注意饼图中我们用source.ci而非source.time来作为获取数据对象。

图例为一个函数：

```
(function (source){
        console.log(source)
        return source.ci.getLegend();
});
```

此函数需要返回一个数组

[第一个个扇区图例, 第二个个扇区图例 … 第n个个扇区图例]

source.ci.getLegend方法有一个参数formatter用来调整输出的图例取得那个数据。

例1如我们需要定义每组数组的名称只返回“设备名称_设备指标”，我们需要对表达式进行如下配置:

```
(function (source){
        var formatter = function(elem){
                return elem.ci + “_” + elem.kpi
        }
        return source.time.getLegend(formatter);
});
```

例2如我们需要定义每组数组的名称只返回“设备名称”，我们需要对表达式进行如下配置:

```
(function (source){
        var formatter = function(elem){
                return elem.ci;
        }
        return source.time.getLegend(formatter);
});
```

例3如我们需要定义每组数组的名称只返回“设备指标”，我们需要对表达式进行如下配置:

```
(function (source){
        var formatter = function(elem){
                return elem.kpi;
        }
        return source.time.getLegend(formatter);
});
```

数据列的格式：

数据列应符合如下格式。

```
[
        {
                name : “第一个饼(名称)”,
                data : [{
                        name : “第一个饼第一个扇形(名称)”,
                        value : 取值
                },{
                        name : “第一个饼第二个扇形(名称)”,
                        value : 取值
                }…{
                        name : “第一个饼第n个扇形(名称)”,
                        value : 取值
                }]
        },
        {
                name : “第二个饼(名称)”,
                data : [{
                        name : “第二个饼第一个扇形(名称)”,
                        value : 取值
                },{
```

```
                name : “第二个饼第二个扇形(名称)”,
                value : 取值
        }…{
                name : “第二个饼第n个扇形(名称)”,
                value : 取值
        }]
    }
    …
    {
        name : “第n个饼(名称)”,
        data : [{
                name : “第n个饼第一个扇形(名称)”,
                value : 取值
        },{
                name : “第n个饼第二个扇形(名称)”,
                value : 取值
        }…{
                name : “第n个饼第n个扇形(名称)”,
                value : 取值
        }]
    }

]
```

注意扇形的名称要与图例中的名称一致才能使图例中对应的地方正确显示出来。
当数据列为一个函数的时候，其返回值也必须符合上面的格式。

```
(function (source){
    console.log(source)
    return source.ci.getSeries();
});
```

同样getSeries也支持formatter参数

例1如我们需要定义每组数组的名称只返回“设备名称_设备指标”，我们需要对表达式进行如下配置:

```
 (function (source){
    var formatter = function(elem){
        return elem.ci + “_” + elem.kpi
    }
    return source.ci.getSeries(formatter);
});
```

例2如我们需要定义每组数组的名称只返回“设备名称”，我们需要对表达式进行如下配置:

```
 (function (source){
    var formatter = function(elem){
        return elem.ci;
    }
    return source.ci.getSeries(formatter);
});
```

例3如我们需要定义每组数组的名称只返回“设备指标”，我们需要对表达式进行如下配置:

```
 (function (source){
```

```
        var formatter = function(elem){
                return elem.kpi;
        }
        return source.ci.getSeries(formatter);
});
```

默认的返回函数source.ci.getSeries()会自动生成一个数据列。但其格式会将相同设备的不同种指标放到同一个饼图种去展示.但是如果我们要将相同指标的不同设备放倒同一个饼图中我们可以像如下的方法对函数进行改写：

```
(function (source){
  var data = [{data : []}];
  var arr = []
  var series = source.ci.getSeries();
  for(var i in  series){
    arr.push(series[i].data[0]);
  };
  data[0].data = arr;
  console.log("1", data);
  console.log("2", series);
  return data;
})
```