



---

# Wave Analytics Dashboard JSON Reference

Salesforce, Spring '17





# CONTENTS

Wave Analytics Dashboard JSON Overview .....	1
View or Modify a Dashboard JSON File .....	2
JSON Example of a Wave Designer Dashboard .....	3
JSON Example of a Classic Designer Dashboard .....	17
<b>Dashboard JSON Properties .....</b>	<b>20</b>
dataSourceLinks (for Wave Designer Dashboards Only) .....	21
gridLayouts (for Wave Designer Dashboards Only) .....	23
widgetStyle Properties (for Wave Designer Dashboards Only) .....	32
layouts (for Classic Designer Dashboards Only) .....	34
steps .....	49
widgets .....	75



# WAVE ANALYTICS DASHBOARD JSON OVERVIEW

The easiest way to build dashboards in Wave Analytics is to use the designer. However, if needed, you can further customize dashboards by editing their JSON files. The JSON defines the components of the dashboard and how they interact.

Modify a dashboard's JSON file to perform advanced customization tasks that can't be accomplished in the designer's user interface, like:

- Manually set up bindings to override the default faceting behavior and specify the relationships between the steps that aren't bound by default.
- Set query limits.
- Specify columns for a values table.
- Specify a SAQL query.
- Populate a filter selector with a specified list of static values instead of from a query.
- Set up layouts for mobile devices for a dashboard.

The last three tasks only pertain to dashboards created with the classic designer. With the Wave dashboard designer, you can use the user interface to accomplish these tasks—no JSON editing required.



**Note:** This document specifies when information applies to only one of the dashboard designers.

# VIEW OR MODIFY A DASHBOARD JSON FILE

Use the Expert Editor Mode to modify the JSON for a dashboard or lens.

Expert Editor Mode displays the JSON of a lens or dashboard and lets you quickly see the effect of your edits in the running asset.

1. To access Expert Editor Mode, open the lens or dashboard you want to edit, and press CTRL+E for PCs or CMD+E for Macs.
2. Modify the JSON in the editor. You can use standard keyboard shortcuts for editing functions and search.
3. To go back to the explorer and see how edits to the JSON appear in the lens or dashboard, click **Done**.
4. To retain your edits, save the lens or dashboard. Changes made in the JSON editor are not saved until you explicitly save the lens or dashboard.

In Expert Editor Mode, the following shortcuts let you perform basic actions from your keyboard.

Expert Editor Mode Keyboard Shortcut	Description
CRTL+3 (Windows); CMD+3 (Mac)	Disregard changes and load the original JSON
CRTL+X (Windows); CMD+X (Mac)	Cut
CRTL+C (Windows); CMD+C (Mac)	Copy
CRTL+V (Windows); CMD+V (Mac)	Paste
CRTL+Z (Windows); CMD+Z (Mac)	Undo
SHIFT+CRTL+Z (Windows); SHIFT+CMD+Z (Mac)	Redo
CRTL+F (Windows); CMD+F (Mac)	Search (RegExp, case-sensitive, or whole word searches available)
CRTL+E (Windows); CMD+E (Mac)	View dashboard with changes to JSON

## EDITIONS

Available in Salesforce Classic and Lightning Experience.

Available for an extra cost in **Enterprise, Performance, and Unlimited** Editions. Also available in **Developer Edition**.

## USER PERMISSIONS

To modify the JSON file that defines a dashboard:

- "Create and Edit Wave Analytics Dashboards"

# JSON EXAMPLE OF A WAVE DESIGNER DASHBOARD

The JSON for each Wave designer dashboard contains multiple levels of properties. Review the structure of the JSON to help you understand where to configure properties.

 **Note:** The structure of the JSON varies based on when you use the Wave designer or classic designer to build the dashboard.

 **Example:**

```
{
  "label": "Opportunity Overview",
  "description": "Sample Wave designer dashboard with multiple layouts.",
  "state": {
    "gridLayouts": [
      {
        "name": "Default",
        "numColumns": 12,
        "pages": [
          {
            "widgets": [
              {
                "colspan": 3,
                "column": 3,
                "name": "container_1",
                "row": 1,
                "rowspan": 6,
                "widgetStyle": {
                  "backgroundColor": "#FFFFFF",
                  "borderColor": "#9687BB",
                  "borderEdges": [
                    "top"
                  ],
                  "borderRadius": 0,
                  "borderWidth": 8
                }
              },
              {
                "colspan": 12,
                "column": 0,
                "name": "text_2",
                "row": 0,
                "rowspan": 1,
                "widgetStyle": {
                  "backgroundColor": "#FFFFFF",
                  "borderColor": "#77B0AD",
                  "borderEdges": [],
                  "borderRadius": 0,
                  "borderWidth": 8
                }
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```
"colspan": 3,
"column": 0,
"name": "container_4",
"row": 1,
"rowspan": 6,
"widgetStyle": {
  "backgroundColor": "#FFFFFF",
  "borderColor": "#77B0AD",
  "borderEdges": [
    "top"
  ],
  "borderRadius": 0,
  "borderWidth": 8
}
},
{
  "colspan": 3,
  "column": 0,
  "name": "text_1",
  "row": 1,
  "rowspan": 1,
  "widgetStyle": {
    "borderEdges": []
  }
},
{
  "colspan": 3,
  "column": 3,
  "name": "text_3",
  "row": 1,
  "rowspan": 1,
  "widgetStyle": {
    "borderEdges": []
  }
},
{
  "colspan": 3,
  "column": 0,
  "name": "number_1",
  "row": 2,
  "rowspan": 2,
  "widgetStyle": {
    "borderEdges": []
  }
},
{
  "colspan": 3,
  "column": 3,
  "name": "number_2",
  "row": 2,
  "rowspan": 2,
  "widgetStyle": {
    "borderEdges": []
  }
}
```



```

        },
        {
            "colspan": 3,
            "column": 0,
            "name": "chart_1",
            "row": 4,
            "rowspan": 3,
            "widgetStyle": {
                "borderEdges": []
            }
        },
        {
            "colspan": 3,
            "column": 3,
            "name": "chart_2",
            "row": 4,
            "rowspan": 3,
            "widgetStyle": {
                "borderEdges": []
            }
        },
        {
            "colspan": 6,
            "column": 6,
            "name": "chart_5",
            "row": 1,
            "rowspan": 6,
            "widgetStyle": {
                "borderEdges": []
            }
        }
    ]
}
],
"selectors": [],
"style": {
    "alignmentX": "left",
    "alignmentY": "top",
    "backgroundColor": "#F2F6FA",
    "cellSpacingX": 8,
    "cellSpacingY": 8,
    "documentId": "",
    "fit": "original"
},
"version": 1
},
{
    "name": "Tablet",
    "numColumns": 8,
    "selectors": [
        "minWidth(600)",
        "maxWidth(900)",
        "orientation(portrait)"
    ],
}

```

```
"pages": [
  {
    "widgets": [
      {
        "colspan": 7,
        "column": 0,
        "name": "text_2",
        "row": 0,
        "rowspan": 1,
        "widgetStyle": {
          "backgroundColor": "#FFFFFF",
          "borderColor": "#77B0AD",
          "borderEdges": [],
          "borderRadius": 0,
          "borderWidth": 8
        }
      },
      {
        "colspan": 3,
        "column": 0,
        "name": "text_1",
        "row": 1,
        "rowspan": 1,
        "widgetStyle": {
          "borderEdges": []
        }
      },
      {
        "colspan": 3,
        "column": 0,
        "name": "number_1",
        "row": 2,
        "rowspan": 2,
        "widgetStyle": {
          "borderEdges": []
        }
      },
      {
        "colspan": 3,
        "column": 0,
        "name": "chart_1",
        "row": 4,
        "rowspan": 3,
        "widgetStyle": {
          "borderEdges": []
        }
      },
      {
        "colspan": 4,
        "column": 0,
        "name": "container_4",
        "row": 1,
        "rowspan": 6,
        "widgetStyle": {
```

```

        "backgroundColor": "#FFFFFF",
        "borderColor": "#77B0AD",
        "borderEdges": [
            "top"
        ],
        "borderRadius": 0,
        "borderWidth": 8
    }
},
{
    "colspan": 3,
    "column": 4,
    "name": "text_3",
    "row": 1,
    "rowspan": 1,
    "widgetStyle": {
        "borderEdges": []
    }
},
{
    "colspan": 3,
    "column": 4,
    "name": "number_2",
    "row": 2,
    "rowspan": 2,
    "widgetStyle": {
        "borderEdges": []
    }
},
{
    "colspan": 3,
    "column": 4,
    "name": "chart_2",
    "row": 4,
    "rowspan": 3,
    "widgetStyle": {
        "borderEdges": []
    }
},
{
    "colspan": 4,
    "column": 4,
    "name": "container_1",
    "row": 1,
    "rowspan": 6,
    "widgetStyle": {
        "backgroundColor": "#FFFFFF",
        "borderColor": "#9687BB",
        "borderEdges": [
            "top"
        ],
        "borderRadius": 0,
        "borderWidth": 8
    }
}

```

## JSON Example of a Wave Designer Dashboard

```
        },
        {
            "colspan": 8,
            "column": 0,
            "name": "chart_5",
            "row": 7,
            "rowspan": 6,
            "widgetStyle": {
                "borderEdges": []
            }
        }
    ]
}
],
"version": 1,
"style": {
    "backgroundColor": "#C5D3E0",
    "cellSpacingX": 4,
    "cellSpacingY": 4,
    "documentId": "",
    "fit": "original",
    "alignmentX": "left",
    "alignmentY": "top"
},
"maxWidth": 500
}
],
"layouts": [],
"steps": {
    "LeadSource_6": {
        "datasets": [
            {
                "id": "0Fbx000000000LzCAI",
                "label": "Opportunities",
                "name": "opportunity1",
                "url": "/services/data/v38.0/wave/datasets/0Fbx000000000LzCAI"
            }
        ],
        "isFacet": true,
        "isGlobal": false,
        "query": {
            "measures": [
                [
                    "sum",
                    "Amount"
                ]
            ],
            "groups": [
                "LeadSource"
            ]
        },
        "type": "aggregate",
        "useGlobal": true,
    }
}
```

## JSON Example of a Wave Designer Dashboard

```
    "visualizationParameters": {
      "visualizationType": "hbar",
      "options": {}
    }
  },
  "LeadSource_7": {
    "datasets": [
      {
        "id": "0Fbx00000000LzCAI",
        "label": "Opportunities",
        "name": "opportunity1",
        "url": "/services/data/v38.0/wave/datasets/0Fbx00000000LzCAI"
      }
    ],
    "isFacet": true,
    "isGlobal": false,
    "query": {
      "measures": [
        [
          "sum",
          "Amount"
        ]
      ],
      "groups": [
        "LeadSource"
      ]
    },
    "type": "aggregate",
    "useGlobal": true,
    "visualizationParameters": {
      "visualizationType": "hbar",
      "options": {}
    }
  },
  "CreatedDate_Year_CreatedDate_Month_9": {
    "datasets": [
      {
        "id": "0Fbx00000000LzCAI",
        "label": "Opportunities",
        "name": "opportunity1",
        "url": "/services/data/v38.0/wave/datasets/0Fbx00000000LzCAI"
      }
    ],
    "isFacet": true,
    "isGlobal": false,
    "query": {
      "measures": [
        [
          "sum",
          "Amount",
          "A",
          {

```

## JSON Example of a Wave Designer Dashboard

```
        "display": "Sum of Amount"
      }
    ],
    [
      "sum",
      "Amount",
      "B",
      {
        "display": "Running Total",
        "format": "currencydollars"
      }
    ]
  ],
  "columns": [
    {
      "query": {
        "measures": [
          [
            "sum",
            "Amount"
          ]
        ],
        "groups": [
          [
            "CreatedDate_Year",
            "CreatedDate_Month"
          ]
        ]
      }
    },
    {
      "query": {
        "measures": [
          [
            "sum",
            "Amount"
          ]
        ],
        "groups": [
          [
            "CreatedDate_Year",
            "CreatedDate_Month"
          ]
        ],
        "formula": "avg(A) over ([-1..0] partition by all order
by ('CreatedDate_Year~~~CreatedDate_Month'))"
      },
      "format": "currencydollars",
      "header": "Running Total"
    }
  ],
  "groups": [
    [
      "CreatedDate_Year",
```

## JSON Example of a Wave Designer Dashboard

```
        "CreateDate_Month"
      ]
    ],
    "selectMode": "single",
    "type": "aggregate",
    "useGlobal": true,
    "visualizationParameters": {
      "visualizationType": "hbar",
      "options": {}
    }
  },
  "Account_Industry_4": {
    "datasets": [
      {
        "id": "0Fbx000000000LzCAI",
        "label": "Opportunities",
        "name": "opportunity1",
        "url": "/services/data/v38.0/wave/datasets/0Fbx000000000LzCAI"
      }
    ],
    "isFacet": true,
    "isGlobal": false,
    "query": {
      "measures": [
        [
          "sum",
          "Amount"
        ]
      ],
      "groups": [
        "Account.Industry"
      ]
    },
    "type": "aggregate",
    "useGlobal": true,
    "visualizationParameters": {
      "visualizationType": "pie",
      "options": {}
    }
  },
  "Amount_3": {
    "datasets": [
      {
        "id": "0Fbx000000000LzCAI",
        "label": "Opportunities",
        "name": "opportunity1",
        "url": "/services/data/v38.0/wave/datasets/0Fbx000000000LzCAI"
      }
    ],
    "isFacet": true,
    "isGlobal": false,
```

```

        "query": {
          "measures": [
            [
              "sum",
              "Amount"
            ]
          ]
        },
        "type": "aggregate",
        "useGlobal": true,
        "visualizationParameters": {
          "visualizationType": "hbar",
          "options": {}
        }
      }
    },
    "widgetStyle": {
      "backgroundColor": "#FFFFFF",
      "borderColor": "#77B0AD",
      "borderEdges": [
        "top"
      ],
      "borderRadius": 0,
      "borderWidth": 8
    },
    "widgets": {
      "container_1": {
        "parameters": {
          "alignmentX": "left",
          "alignmentY": "top",
          "documentId": "",
          "fit": "original"
        },
        "type": "container"
      },
      "number_2": {
        "parameters": {
          "compact": true,
          "exploreLink": true,
          "measureField": "sum_Amount",
          "numberColor": "#335779",
          "numberSize": 32,
          "step": "LeadSource_6",
          "textAlignment": "center",
          "titleColor": "#335779",
          "titleSize": 16
        },
        "type": "number"
      },
      "number_1": {
        "parameters": {
          "compact": true,
          "exploreLink": true,
          "measureField": "sum_Amount",

```



## JSON Example of a Wave Designer Dashboard

```
        "numberColor": "#335779",
        "numberSize": 32,
        "step": "Amount_3",
        "textAlignment": "center",
        "titleColor": "#335779",
        "titleSize": 16
    },
    "type": "number"
},
"text_1": {
    "parameters": {
        "fontSize": 20,
        "text": "Industry",
        "textAlignment": "center",
        "textColor": "#000000"
    },
    "type": "text"
},
"container_4": {
    "parameters": {
        "alignmentX": "left",
        "alignmentY": "top",
        "documentId": "",
        "fit": "original"
    },
    "type": "container"
},
"text_3": {
    "parameters": {
        "fontSize": 20,
        "text": "Lead Source",
        "textAlignment": "center",
        "textColor": "#000000"
    },
    "type": "text"
},
"text_2": {
    "parameters": {
        "fontSize": 20,
        "text": "Opportunity Overview Dashboard",
        "textAlignment": "center",
        "textColor": "#000000"
    },
    "type": "text"
},
"chart_5": {
    "parameters": {
        "autoFitMode": "fit",
        "showValues": true,
        "barSize": 25,
        "legend": {
            "showHeader": true,
            "show": true,
            "position": "bottom-center",
```

```

        "inside": false
    },
    "axisMode": "multi",
    "visualizationType": "stackhbar",
    "exploreLink": true,
    "title": {
        "label": "",
        "align": "center",
        "subtitleLabel": ""
    },
    "trellis": {
        "enable": false,
        "type": "x",
        "chartsPerLine": 4
    },
    "measureAxis2": {
        "showTitle": true,
        "showAxis": true,
        "title": ""
    },
    "measureAxis1": {
        "showTitle": true,
        "showAxis": true,
        "title": ""
    },
    "normalize": false,
    "step": "CreatedDate_Year_CreatedDate_Month_9",
    "theme": "wave",
    "autoFit": false,
    "dimensionAxis": {
        "showTitle": true,
        "showAxis": true,
        "title": ""
    }
    },
    "type": "chart"
},
"chart_2": {
    "parameters": {
        "legend": {
            "showHeader": true,
            "show": true,
            "position": "right-top",
            "inside": false
        },
        "showMeasureTitle": false,
        "showTotal": true,
        "visualizationType": "pie",
        "step": "LeadSource_7",
        "theme": "wave",
        "exploreLink": true,
        "title": {
            "label": "",
            "align": "center",

```

```

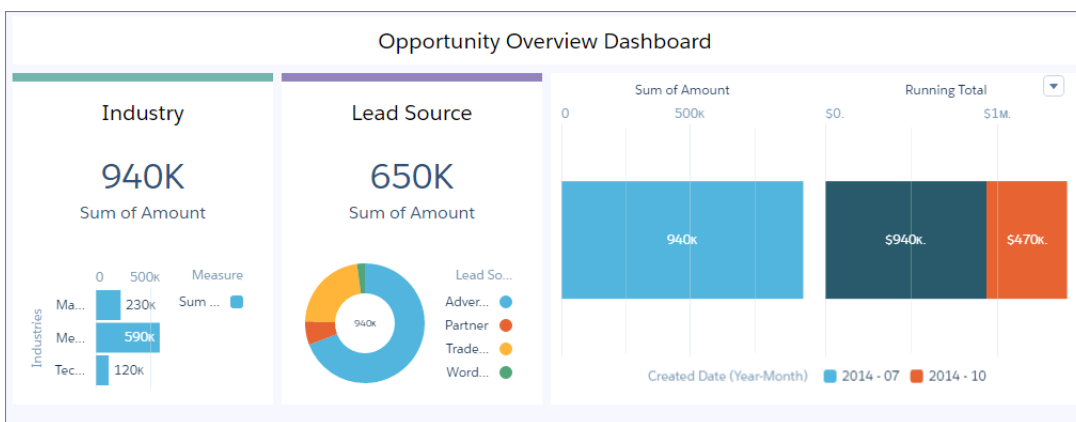
        "subtitleLabel": ""
    },
    "trellis": {
        "enable": false,
        "type": "x",
        "chartsPerLine": 4
    },
    "inner": 50
},
"type": "chart"
},
"chart_1": {
    "parameters": {
        "autoFitMode": "fit",
        "showValues": true,
        "barSize": 25,
        "legend": {
            "showHeader": true,
            "show": true,
            "position": "right-top",
            "inside": false
        },
        "axisMode": "multi",
        "visualizationType": "hbar",
        "exploreLink": true,
        "title": {
            "label": "",
            "align": "center",
            "subtitleLabel": ""
        },
        "trellis": {
            "enable": false,
            "type": "x",
            "chartsPerLine": 4
        },
        "measureAxis2": {
            "showTitle": true,
            "showAxis": true,
            "title": ""
        },
        "measureAxis1": {
            "showTitle": false,
            "showAxis": true,
            "title": ""
        },
        "step": "Account_Industry_4",
        "theme": "wave",
        "autoFit": false,
        "dimensionAxis": {
            "showTitle": true,
            "showAxis": true,
            "title": "Industries"
        }
    }
},

```

## JSON Example of a Wave Designer Dashboard

```
        "type": "chart"
      }
    },
    "datasets": [
      {
        "id": "0Fbx00000000LzCAI",
        "label": "Opportunities",
        "name": "opportunity1",
        "url": "/services/data/v38.0/wave/datasets/0Fbx00000000LzCAI"
      }
    ]
  }
}
```

The JSON file defines the following dashboard created in the Wave designer.



This dashboard displays the following widgets:

- Two faceted number widgets: `number_1` (based on step `Amount_3`) and `number_2` (based on step `LeadSource_6`).
- Three faceted chart widgets: one bar chart `chart_5` (based on step `Account_Industry_4`), one pie chart `chart_2` (based on step `LeadSource_7`), and one stacked bar chart `chart_5` (based on step `CreatedDate_Year_CreatedDate_Month_9`). The steps reference the same `0Fbx00000000LzCAI` dataset.
- Two container widgets (`container_1` and `container_4`) that each group a number widget and a chart.
- Two text widgets (`text_1` and `text_3`) that provide the labels for the containers.

# JSON EXAMPLE OF A CLASSIC DESIGNER DASHBOARD

The JSON for each classic designer dashboard contains multiple levels of properties. Review the structure of the JSON to help you understand where to configure properties.

 **Note:** The structure of the JSON varies based on when you use the Wave designer or classic designer to build the dashboard.

 **Example:**

```
{
  "description": "Shows opportunities by industry.",
  "label" : "Opportunities",
  "folder" : {
    "id" : "00136000000SpXiAAK"
  },
  "state": {
    "steps": {
      "AccountIndustryPieChart": {
        "type": "aggregate",
        "isGlobal": false,
        "isFacet": true,
        "useGlobal": true,
        "selectMode": "single",
        "start": null,
        "visualizationParameters": {
          "visualizationType": "pie"
        },
        "query": {
          "query":
            "{\\"groups\\": [\\"AccountId.Industry\\"], \\"measures\\": [\\"count\\", \\"*\\"]}",
          "version": -1
        },
        "datasets": [
          {
            "name": "Opps"
          }
        ]
      },
      "AccountIndustryBarChart": {
        "type": "aggregate",
        "isGlobal": false,
        "isFacet": true,
        "useGlobal": true,
        "selectMode": "single",
        "start": null,
        "visualizationParameters": {
          "options": {
            "sqrt": true
          },
          "visualizationType": "hbar"
        }
      }
    }
  }
}
```

## JSON Example of a Classic Designer Dashboard

```
    "query": {
      "query":
        "{\\\"measures\\\":[[\\\"sum\\\",\\\"Amount\\\"]],\\\"groups\\\":[\\\"AccountId.Industry\\\"],\\\"order\\\":[[-1,{\\\"ascending\\\":false}]]}",

      "version": -1
    },
    "datasets": [
      {
        "name": "Opps"
      }
    ]
  },
  "widgets": {
    "chart_1": {
      "type": "chart",
      "position": {
        "zIndex": 3,
        "x": 10,
        "y": 80,
        "w": 500,
        "h": 300
      },
      "parameters": {
        "step": "AccountIndustryPieChart",
        "legend": true,
        "visualizationType": "pie"
      }
    },
    "text_1": {
      "type": "text",
      "position": {
        "zIndex": 4,
        "x": 0,
        "y": 10
      },
      "parameters": {
        "text": "Account Industries",
        "textAlignment": "left"
      }
    },
    "text_4": {
      "type": "text",
      "position": {
        "zIndex": 11,
        "x": 500,
        "y": 10
      },
      "parameters": {
        "text": "Amount by Industries"
      }
    },
    "chart_4": {
      "type": "chart",
```

## JSON Example of a Classic Designer Dashboard

```
    "position": {
      "zIndex": 12,
      "x": 530,
      "y": 70,
      "w": 500,
      "h": 300
    },
    "parameters": {
      "step": "AccountIndustryBarChart",
      "sqrt": true,
      "visualizationType": "vbar"
    }
  }
}
```

The dashboard JSON file defines a simple dashboard created in the classic designer. This dashboard displays two faceted widgets: one pie chart `chart_1` (based on step `AccountIndustryPieChart`) and one bar chart `chart_4` (based on step `AccountIndustryBarChart`). Both steps reference the same `Opps` dataset.

# DASHBOARD JSON PROPERTIES

The dashboard JSON consists of properties that define layouts, widgets, and steps. key defines all layouts for the Wave designer dashboard. It contains a separate node for each layout. Each layout has properties that provide information about the devices that can use the layout as well as the placement of each widget in the layout. It also contains dashboard properties, like cell spacing in the grid and the dashboard's background color or image.

Some properties are exposed and editable in the dashboard designer user interface. Others are only editable via JSON.

In each dashboard JSON, you'll find the following high-level properties.

Property Name	Details
label	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Name of the dashboard.</p>
description	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Description of the dashboard.</p>
state	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> No</p> <p><b>Description</b> Specifies properties for all layouts, widgets, and steps defined in the dashboard. The state properties vary based on the designer used to save the dashboard. For example, if the dashboard is saved in Wave dashboard designer, the state also contains a <code>gridLayouts</code> section. When you save a dashboard using the dashboard designer, the state of the dashboard is persisted in the JSON.</p>
datasets	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> No</p>



Property Name	Details
	<p><b>Description</b></p> <p>Specifies all datasets used by steps in the dashboard.</p>

The following sections describe the different properties nested under `state`.

#### [dataSourceLinks \(for Wave Designer Dashboards Only\)](#)

The `dataSourceLinks` section defines all data sources configured for the dashboard.

#### [gridLayouts \(for Wave Designer Dashboards Only\)](#)

The `gridLayouts` section defines all layouts built for the dashboard.

#### [widgetStyle Properties \(for Wave Designer Dashboards Only\)](#)

The `widgetStyle` key contains the default widget properties that can be applied to each widget. This section only applies to dashboards that are created using the Wave dashboard designer.

#### [layouts \(for Classic Designer Dashboards Only\)](#)

Add a `layouts` section to the JSON to customize the appearance of a classic designer dashboard on mobile devices.

#### [steps](#)

The `steps` section defines all steps created in and clipped to the dashboard. The properties vary based on whether the dashboard is built using the Wave dashboard designer or classic designer.

#### [widgets](#)

The `widgets` section defines the widgets that appear in the dashboard. Each widget has a name.

## dataSourceLinks (for Wave Designer Dashboards Only)

---

The `dataSourceLinks` section defines all data sources configured for the dashboard.

For more information about connected data sources, see [Configure Cross-Dataset Faceting with Connected Data Sources](#).

### Example:

```
"dataSourceLinks": [
  {
    "fields": [
      {
        "dataSourceName": "ServiceOpportunity3",
        "dataSourceType": "saql",
        "fieldName": "AccountId"
      },
      {
        "dataSourceName": "account",
        "dataSourceType": "saql",
        "fieldName": "Id"
      }
    ],
    "label": "ServiceOpportunities Dataset to Account Dataset: Account ID",
    "name": "Link_970"
  }
]
```

```

    },
    {
      "fields": [
        {
          "dataSourceName": "ServiceOpportunity3",
          "dataSourceType": "saql",
          "fieldName": "StageName"
        },
        {
          "dataSourceName": "Static_Opp_Stage_1",
          "dataSourceType": "static",
          "fieldName": "value"
        }
      ],
      "label": "Static Opp Stage to ServiceOpportunities Dataset",
      "name": "Link_953"
    }
  ]
}

```

### dataSourceLinks Properties

The `dataSourceLinks` key defines all data source connections for the Wave designer dashboard. It contains a separate node for each connection. Each connection has properties about each data source.

## dataSourceLinks Properties

The `dataSourceLinks` key defines all data source connections for the Wave designer dashboard. It contains a separate node for each connection. Each connection has properties about each data source.

Property Name	Details
fields	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> List of data sources included in the connection. Each data source contains the following properties.</p> <p><b>dataSourceName</b> API name of the dataset or ID of the static step.</p> <p><b>dataSourceType</b> The type of data source: "saql" for a dataset or "static" for a static step.</p> <p><b>fieldName</b> Name of the field used to match records between the data sources.</p>
label	<p><b>Type</b> String</p>

Property Name	Details
	<p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> Display label for the data source connection.</p>
name	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> No.</p> <p><b>Description</b> API name of the data source connection.</p>

## gridLayouts (for Wave Designer Dashboards Only)

The `gridLayouts` section defines all layouts built for the dashboard.

For more information about layouts for Wave designer dashboards, see [Generate Unique Dashboard Layouts for Different Devices](#). For information about layouts for classic designer dashboards, see [layouts \(for Classic Designer Dashboards Only\)](#).

### Example:

```
"gridLayouts": [
  {
    "name": "Default",
    "numColumns": 14,
    "pages": [
      {
        "widgets": [
          {
            "colspan": 9,
            "column": 3,
            "name": "dateselector_1",
            "row": 0,
            "rowspan": 15,
            "widgetStyle": {}
          },
          {
            "row": 12,
            "column": 0,
            "rowspan": 9,
            "colspan": 3,
            "name": "image_1",
            "widgetStyle": {}
          },
          {
            "colspan": 12,
            "column": 1,
            "name": "table_1",
```

```

        "row": 29,
        "rowspan": 18,
        "widgetStyle": {
            "borderEdges": [ "all" ],
            "backgroundColor": "#2EC2BA",
            "borderColor": "#9271E8",
            "borderWidth": 4,
            "borderRadius": 4
        }
    }
]
}
],
"rowHeight": "fine",
"selectors": [
    "minWidth(400)",
    "orientation(portrait)",
    "platform(iOS)"
],
"style": {
    "alignmentX": "center",
    "alignmentY": "center",
    "backgroundColor": "#2EC2BA",
    "cellSpacingX": 4,
    "cellSpacingY": 0,
    "fit": "stretch",
    "gutterColor": "#AFA3CE",
    "image": {
        "name": "X1png",
        "namespace": ""
    }
},
"version": 1,
"maxWidth": 800
}
]

```

### [gridLayouts Properties](#)

The `gridLayouts` key defines all layouts for the Wave designer dashboard. It contains a separate node for each layout. Each layout has properties that provide information about the devices that can use the layout as well as the placement of each widget in the layout. It also contains dashboard properties, like cell spacing in the grid and the dashboard's background color or image.

## gridLayouts Properties

The `gridLayouts` key defines all layouts for the Wave designer dashboard. It contains a separate node for each layout. Each layout has properties that provide information about the devices that can use the layout as well as the placement of each widget in the layout. It also contains dashboard properties, like cell spacing in the grid and the dashboard's background color or image.

Property Name	Details
name	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> Name of the layout.</p>
maxWidth	Maximum width (in pixels) that the dashboard can use. If needed, Wave rearranges the existing dashboard widgets based on this setting in the layout.
numColumns	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> The number of columns in the designer grid for this layout.</p>
pages	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> No</p> <p><b>Description</b> Contains properties that determine the placement of each widget in the dashboard layout. Currently, Wave designer supports only one page for each layout.</p>
rowHeight	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> The height of each row in the designer grid for this layout. Valid values are <code>fine</code> and <code>normal</code> (default).</p>
selectors	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> Device requirements that help Wave choose the optimal layout for the device accessing the dashboard.</p>
style	<p><b>Type</b> Array</p>

Property Name	Details
	<p><b>Exposed in the Dashboard Designer's User Interface</b> Yes.</p> <p><b>Description</b> Properties about the designer grid, including columns, rows, cell spacing, and background.</p>

#### [pages Properties](#)

The `pages` key contains properties that determine the placement of each widget in the Wave designer dashboard layout. Currently, Wave designer supports only one page for each layout.

#### [selectors Properties](#)

The `selectors` key contains layout properties that specify the layout name, designer grid settings, background settings, and requirements for devices that can use this layout.

#### [style Properties](#)

The `style` key contains the dashboard properties, like cell spacing in the grid, as well as the dashboard's background color or image.

## pages Properties

The `pages` key contains properties that determine the placement of each widget in the Wave designer dashboard layout. Currently, Wave designer supports only one page for each layout.


Property Name	Details
<code>widgets</code>	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> No</p> <p><b>Description</b> Contains properties that determine the height and width of each widget, and where it's placed on the dashboard layout.</p>

#### [widgets Properties](#)

The `widgets` key contains properties that determine the height and width of each widget, and where it's placed on the dashboard layout. Because the Wave dashboard designer uses a grid, you specify the properties in terms of rows and columns. For example, you specify the number of columns to determine the width of a widget.

## widgets Properties

The `widgets` key contains properties that determine the height and width of each widget, and where it's placed on the dashboard layout. Because the Wave dashboard designer uses a grid, you specify the properties in terms of rows and columns. For example, you specify the number of columns to determine the width of a widget.

Property Name	Details
name	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> No</p> <p><b>Description</b> Internal name of the widget. This name is used to reference the widget in the dashboard JSON.</p>
column	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes. Value is determined based on the widget's placement.</p> <p><b>Description</b> The column number where the widget starts. <code>column</code> and <code>row</code> specify the top left corner of the widget.</p> <p> <b>Note:</b> If this widget is included in a container, these properties are relative to the container widget.</p>
row	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes. Value is determined based on the widget's placement.</p> <p><b>Description</b> The row number where the widget starts. <code>column</code> and <code>row</code> specify the top left corner of the widget.</p>
colspan	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes. Value is determined based on the widget's placement.</p> <p><b>Description</b> The number of columns that a widget spans—the width of the widget. If the dashboard doesn't have enough columns to accommodate the specified width, then columns are added to the dashboard.</p>
rowspan	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes. Value is determined based on the widget's placement.</p> <p><b>Description</b> The number of rows that a widget spans—the height of the widget. If the dashboard doesn't have enough rows to accommodate the specified height, then rows are added.</p>

Property Name	Details
<code>widgetStyle</code>	<p><b>Type</b> Array</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>All widgets</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> No</p> <p><b>Description</b> Contains properties that set the border type, border color, and background color.</p>

### [widgetStyle Properties](#)

The `widgetStyle` key contains properties that set the border type, border color, and background color of the widget. You can specify these attributes at two levels. To set the default for all dashboard widgets, use the `widgetStyle` field under `gridLayouts`. To set a specific widget, use the `widgetStyle` field under `widgets`. This setting overrides the default settings for all widgets.

### **widgetStyle Properties**

The `widgetStyle` key contains properties that set the border type, border color, and background color of the widget. You can specify these attributes at two levels. To set the default for all dashboard widgets, use the `widgetStyle` field under `gridLayouts`. To set a specific widget, use the `widgetStyle` field under `widgets`. This setting overrides the default settings for all widgets.

Property Name	Details
<code>backgroundColor</code>	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Background color of the widget. The default is #FFFFFF.</p>
<code>borderColor</code>	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Color of the widget's border. The default is #FFFFFF.</p>



Property Name	Details
<code>borderEdges</code>	<p><b>Type</b> List</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> A list of values that specify which edges of the widget have a border. Valid values are <code>left</code>, <code>right</code>, <code>top</code>, <code>bottom</code>, and <code>all</code>. Default is no border.</p>
<code>borderRadius</code>	<p><b>Type</b> Integer</p> <p><b>Available for This Widget</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The roundness of the border corners.  Valid values are: 0 (not rounded, default), 4, 8, and 16. The higher the value, the more rounded the corner.</p>
<code>borderWidth</code>	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Width of the widget's border. Valid values are 1, 2 (default), 4, and 8.</p>

## selectors Properties

The `selectors` key contains layout properties that specify the layout name, designer grid settings, background settings, and requirements for devices that can use this layout.

Property Name	Details
<code>minWidth (&lt;width&gt;)</code>	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>

Property Name	Details
	<p><b>Description</b></p> <p>Minimum width (in pixels) of the devices supported by this layout.</p>
<code>maxWidth (&lt;width&gt;)</code>	<p><b>Type</b></p> <p>Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Maximum width (in pixels) of the devices supported by this layout.</p>
<code>orientation (&lt;orientation&gt;)</code>	<p><b>Type</b></p> <p>String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Orientation of the devices supported by this layout. Valid values are: <code>portrait</code> or <code>landscape</code>. If this property is not specified, then the layout supports both orientations.</p>
<code>platform (&lt;platform&gt;)</code>	<p><b>Type</b></p> <p>String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Platform of the devices supported by this layout. Valid values are: <code>iOS</code> or <code>Android</code>. If this property is not specified, the layout supports both platforms.</p>

## style Properties

The `style` key contains the dashboard properties, like cell spacing in the grid, as well as the dashboard's background color or image.


Property Name	Details
<code>alignmentX</code>	<p><b>Type</b></p> <p>String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>The horizontal alignment of the background image applied to the dashboard.</p> <p>Valid values are: <code>left</code> (default), <code>center</code>, and <code>right</code>.</p>

Property Name	Details
alignmentY	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The vertical alignment of the background image applied to the dashboard. Valid values are: <code>top</code> (default), <code>center</code>, and <code>bottom</code>.</p>
backgroundColor	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Background color of the dashboard, specified in hex color code. The default is <code>#FFFFFF</code>.</p>
cellSpacingX	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Horizontal spacing (in pixels) between cells in the dashboard grid. Valid values are 0, 4, 8 (default), and 16.</p>
cellSpacingY	<p><b>Type</b> Integer</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Vertical spacing (in pixels) between cells in the dashboard grid. Valid values are 0, 4, 8 (default), and 16.</p>
documentId	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The 15-character document ID of the image to apply as the dashboard's background. To ensure security, upload the image file to Salesforce as a document, and select the <b>Externally Available Image</b> option. The image doesn't show up if this option is not selected or the referenced document is not an image.</p>

Property Name	Details
fit	<p><b>Type</b> String</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates how to scale the image. Valid values are: <code>original</code> (default), <code>stretch</code>, <code>tile</code>, <code>fitwidth</code>, and <code>fitheight</code>.</p>
image	<p><b>Type</b> Array</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) Identifies the image using the following properties.</p> <p><b>name</b> Name of the image.</p> <p><b>namespace</b> Optional. Namespace of the image. Default is null.</p> <p><b>Example</b></p> <pre>"image": {   "name": "My_Corporate_Logo",   "namespace": "" }</pre>

## widgetStyle Properties (for Wave Designer Dashboards Only)

The `widgetStyle` key contains the default widget properties that can be applied to each widget. This section only applies to dashboards that are created using the Wave dashboard designer.

 **Note:** You can specify these attributes at two levels. To set the default for all dashboard widgets, use the `widgetStyle` field under `gridLayouts`. To set a specific widget, use the `widgetStyle` field under `widgets`. Settings at the widget level override the default settings for all widgets.

Property Name	Details
backgroundColor	<p><b>Type</b> String</p> <p><b>Available for This Widget</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>

Property Name	Details
	<p><b>Description</b></p> <p>Color of the widget's background, specified in hex color code. The default is #FFFFFF.</p>
borderColor	<p><b>Type</b> String</p> <p><b>Available for This Widget</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>Color of the widget's border, specified in hex color code. The default is #FFFFFF. If no border is specified, the widget has no border.</p>
borderEdges	<p><b>Type</b> List</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>A list of values that specify which edges of the widget have a border. Valid values are <code>left</code>, <code>right</code>, <code>top</code>, <code>bottom</code>, and <code>all</code>. Default is no border.</p>
borderRadius	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>Roundness of the border corners.</p> <p>Valid values are: 0(not rounded, default), 4, 8, and 16. The higher the value, the more rounded the corner.</p>
borderWidth	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b> All widgets</p> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>

Property Name	Details
	<p><b>Description</b></p> <p>Thickness of the border.</p> <p>Valid values are: 1, 2 (default), 4, and 8. The higher the value, the thicker the border.</p>

## layouts (for Classic Designer Dashboards Only)

Add a `layouts` section to the JSON to customize the appearance of a classic designer dashboard on mobile devices.

 **Note:** For more information about layouts for Wave designer dashboards, see [Generate Unique Dashboard Layouts for Different Devices](#).

There are two types of classic designer dashboard layouts for mobile devices:

### Absolute (default)

If no `layouts` section is defined in your dashboard's JSON, then the dashboard's layout is absolute.

The absolute layout is optimized for display in a Web browser on a desktop or laptop computer.

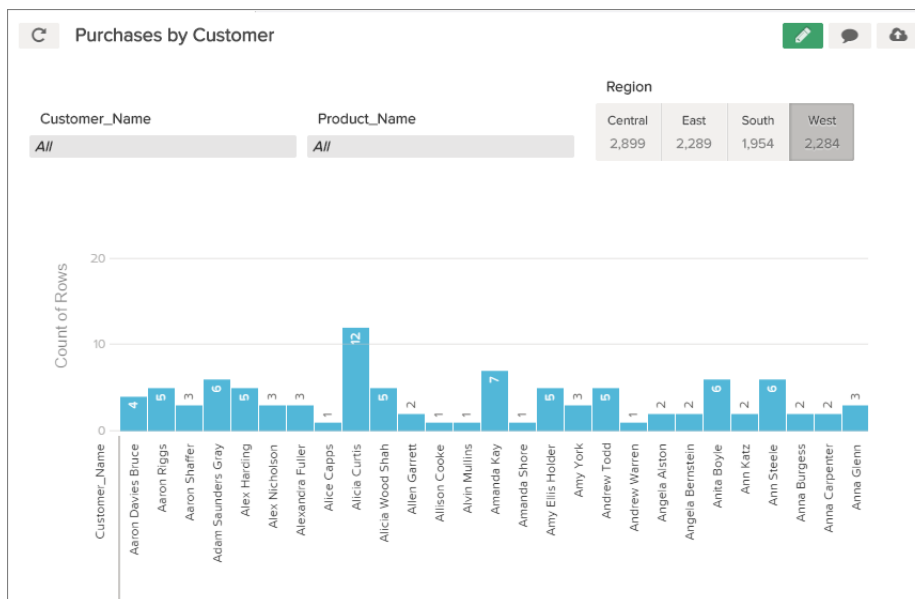
### Mobile

If a `layouts` section is present in your dashboard's JSON, then the dashboard's layout is mobile.

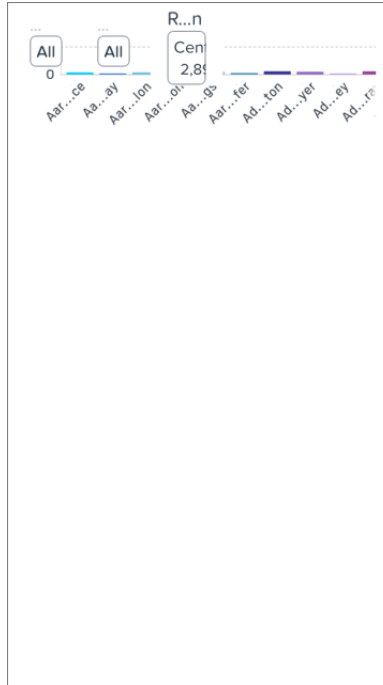
The mobile layout lets you optimize the position, order, and size of the widgets in your dashboard for display on mobile devices.

This layout is made up of rows, columns, cells, and pages. Each cell in the grid can contain zero or more widgets. The number of rows, columns, and cells in your mobile layout depend on the number of widgets and the number of pages.

A dashboard with an absolute layout looks great in a Web browser:



The same dashboard with an absolute layout might not render well on a smart phone:



By using a mobile layout with two pages, the dashboard renders perfectly on a smart phone:



[Use a Mobile Layout for Your Dashboard](#)

Use a mobile layout to customize your dashboard’s appearance on mobile devices.

[Understanding Column, Row, and Cell Sizing in Mobile Layouts](#)

Widgets size, row size, and the number of columns are determined dynamically, but can also be specified in the JSON.

### Layouts Specification

The `layouts` section is used to customize how dashboards display on mobile devices.

### layouts Properties

The `layouts` key specifies the position, order, and size of each widget in the mobile layout. This layout is made up of rows, columns, cells, and pages. Each cell in the grid can contain zero or more widgets. The number of rows, columns, and cells in your mobile layout depend on the number of widgets and the number of pages

## Use a Mobile Layout for Your Dashboard

Use a mobile layout to customize your dashboard's appearance on mobile devices.

In a dashboard's JSON file, the `layouts` section is a child of the `state` section and a sibling of the `widgets` and `steps` sections.

1. From the open dashboard, press CTRL+E for PC or CMD+E for Mac. This opens expert editor mode. For more information, see [View or Modify a Dashboard JSON File](#).
2. Add a `layouts` section to your dashboard's JSON.

For example, this `layouts` section defines a mobile layout with two pages, two rows of widgets on each page. The first page has 1 widget on each row. The second page has two widgets on the first row, and one widget on the second row.

```
"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "buttongroup_2",
          "chart_1"
        ]
      },
      {
        "rows": [
          "dimfilter_1 | dimfilter_3",
          "chart_1"
        ]
      }
    ]
  },
  "version": 1
}
```

3. Optionally, customize the layout of your dashboard by setting [attributes for each widget and row](#).

For example, the `layouts` from step two can be updated to include widget and row attributes. The first row on the first page has a row height of 300 pixels. The chart widget on the second page has a width of 2 columns.

```
"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "buttongroup_2 | row:{height=300}",
          "chart_1"
        ]
      }
    ]
  }
]
```



```

    },
    {
      "rows": [
        "dimfilter_1 | dimfilter_3",
        "chart_1 {colspan=2}"
      ]
    }
  ],
  "version": 1
}

```

- Optionally, set device-specific and orientation-specific layouts for your dashboard. For available device and orientation options, see [Layouts Options](#) in the [Layouts Specification](#) guide.

For example, the `layouts` from step three can be updated to use only one page when viewed on an iPad in landscape mode:

```

"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "buttongroup_2 | row:{height=300}",
          "chart_1"
        ]
      },
      {
        "rows": [
          "dimfilter_1 | dimfilter_3",
          "chart_1 {colspan=2}"
        ]
      }
    ]
  },
  {
    "device": "ipad",
    "orientation": "landscape",
    "pages": [
      {
        "rows": [
          "dimfilter_1 | dimfilter_3 | buttongroup_2",
          "chart_1 {colspan=3}"
        ]
      }
    ]
  },
  "version": 1
}

```

- Click **Switch to Runtime**, and then save your updated dashboard.

6. Test your dashboard's new mobile layout by viewing the dashboard on a mobile device.

SEE ALSO:

[layouts Properties](#)

[Layouts Specification](#)

## Understanding Column, Row, and Cell Sizing in Mobile Layouts

Widgets size, row size, and the number of columns are determined dynamically, but can also be specified in the JSON.

### How Column Number and Size Are Set

The number of columns in your mobile layout is equivalent to the number of widgets in your rows. If there are three widgets in each row, then the dashboard has three columns. If your mobile layout has two rows with four widgets in row one and five widgets in row two, then the dashboard has five columns. If the `colspan` attribute specifies a number of columns greater than the number of widgets in any row, then the dashboard adds columns to accommodate the `colspan` attribute.

For example, a dashboard with this `layouts` section has three columns on the first page and two columns on the second page:

```
"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "buttongroup_2",
          "chart_1 {colspan=3}"
        ]
      },
      {
        "rows": [
          "dimfilter_1 | dimfilter_3",
          "chart_1"
        ]
      }
    ]
  },
  {
    "version": 1
  }
]
```

Remember these tips when determining how many columns are in your mobile layout:

- All columns have the same width. If your dashboard has four columns, then each column is half the width of a dashboard with two columns.
- Each page of a dashboard independently determines how many columns appear. For example, a dashboard can have three columns on page one, and four columns on page two.
- Every dashboard has at least one column.
- There is no limit to the number of columns that a dashboard can have. If you add too many columns, then column width could become impractically small. Remember to test your layout for usability!

## How Row Number and Height Are Set

For each row, here's how height is calculated:

- If a row height is set using the `height` attribute, then the row's height is equal to the specified value.
- If one or more widgets in the row has a preferred height, then the row's height is equal to that of whichever preferred height is tallest.
- If there is no `height` attribute and none of the row's widgets have a preferred height, then the row's height dynamically grows to occupy the available space. If multiple rows grow dynamically, then their heights are equal to one another. For example, if there are 200 pixels of available space, and two rows with dynamically set heights, then each row has a height of 100 pixels.

## How Widgets Are Sized

Some widgets have absolute sizes, and some scale dynamically.

Widget	Has a Fixed Width?	Has a Fixed Height?	Width Scaling Behavior	Height Scaling Behavior
Link	Yes	Yes	Don't scale	Don't scale
Text	No	If one line long, yes. If more than one line long, no.	Scale to fit text	Scale to fit text
Pillbox	No	Yes	Scale	Don't scale
Box	No	No	Scale	Scale
Chart	No	No	Scale	Scale
List selector	No	Yes	Scale	Don't scale
Range selector	No	Yes	Scale	Don't scale
Number	No	Yes	Scale	Don't scale

## Layouts Specification

The `layouts` section is used to customize how dashboards display on mobile devices.

In a dashboard's JSON file, the `layouts` section is a child of the `state` section and a sibling of the `widgets` and `steps` sections. Here is an example of a typical `layouts` section:

```
"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "widget_name_1",
          "widget_name_2"
        ]
      }
    ]
  },
]
```

```

    {
      "rows": [
        "widget_name_3 | widget_name_4",
        "widget_name_2 {attribute=2}"
      ]
    }
  ],
  "version": 1
},
{
  "device": "ipad",
  "orientation": "landscape",
  "pages": [
    {
      "rows": [
        "widget_name_1 | widget_name_3 | widget_name_4 | row: {attribute=300}",
        "widget_name_2 {widget_name=3}"
      ]
    }
  ],
  "version": 1
}

```

In the prior example, *widget\_name* refers to a specific widget named in the *widgets* section of the JSON file. *Attribute* refers to one of the attributes listed in the [layouts Properties](#). The pipe character (|) is the delimiter for cells. A cell can contain multiple widgets separated by a comma (,). Rows are delimited by a comma (,) outside the quoted string (each quoted string is a single row).

## Simple Layouts Section

Here's a simple `layouts` section that has four widgets on four rows in a single column on a single page:

```

"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "buttongroup_1",
          "dimfilter_1",
          "dimfilter_2",
          "chart_1"
        ]
      }
    ]
  }
]
"version": 1
}

```

## Complex Layouts Section

A more complex `layouts` section can be used to set device-specific and orientation-specific display rules. The following `layouts` section lays out the dashboard's widgets on two pages. The first page's first row has a height of 300 pixels. The second page has two rows and two columns. One of the cells in the first row contains two widgets. One of the box widgets has three attributes set. The chart

widget spans two columns. If the dashboard is viewed on an iPad in landscape mode, then only one page with two rows is shown. The first row has three widgets and the second row has one widget that spans three columns.

```

"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "buttongroup_2 | row: {height=300}",
          "chart_1"
        ]
      },
      {
        "rows": [
          "dimfilter_1, box_1 {colspan=2, rowspan=2, zIndex=-1, vpad=5, hpad=5} |
          dimfilter_2", "chart_1 {colspan=2}"
        ]
      }
    ],
    "version": 1
  },
  {
    "device": "ipad",
    "orientation": "landscape",
    "pages": [
      {
        "rows": [
          "dimfilter_1, box_1 {colspan=2, rowspan=3, zIndex=-1, vpad=5, hpad=5} | dimfilter_2
          |
          buttongroup_2",
          "chart_1 {colspan=3}"
        ]
      }
    ],
    "version": 1
  }
]

```

## Layouts Options

The previous example shows a layout specifically for an iPad in landscape mode ("device:ipad, orientation:landscape"). Layout device and orientation choices are as follows:

- "device": "default": For layouts not targeted to any specific device or orientation.
- "device": "ipad", "orientation": "portrait": For Apple iPad in portrait mode.
- "device": "ipad", "orientation": "landscape": For Apple iPad in landscape mode.
- "device": "ipad": For Apple iPad in either portrait or landscape mode.
- "device": "iphone": For Apple iPhone; portrait mode is implied.
- "device": "external": For displaying on an external device, for example if device is connected via HDMI cable to a projector or display. To use external layout, select Presentation Mode in Settings.
- "device": "applewatch": For Apple Watch. Supports only a single, scrolling page.
- "orientation": "portrait": For either iPhone or iPad in portrait mode.

- `"orientation": "landscape"`: For iPad in landscape mode.
- 📌 **Note:** If the app is viewed on Apple Watch and `"device": "applewatch"` layout is not present, the app first tries to reformat the first page of the `"device": "iphone"` layout. If `"device": "iphone"` is not present, it then attempts to use the first page of the `"device": "default"` layout.
- 📌 **Note:** If the app is viewed on an external device and `"device": "external"` layout is not present, the app first tries to use the first page of the `"device": "ipad"` `"orientation": "landscape"`. If `"device": "ipad"` `"orientation": "landscape"` is not present, it then attempts to use the first page of the `"device": "default"` layout.

## Layout Autoformatting

If `layouts` is not specified, Wave uses autoformatting to present the dashboard, which takes a best guess about the appropriate layout to use. Note the following about layout autoformatting:

- With AppleWatch, autoformat uses the first page of the default layout and converts it to a single column.
- With an external device, autoformat supports only a single, unscrollable page and attempts to fit all the dashboard contents on the external display.
- Autoformat supports a limited number of columns on each device, as shown in the table.

Device	Maximum columns supported by autoformatting
Apple Watch	One
Apple iPhone	Two
Apple iPad	Four

Autoformatting is enabled by default. To disable autoformatting, for example for a carefully designed dashboard that cannot use a mobile layout, add an empty `pages` array under the `layouts` array, which looks like this:

```
"layouts": [
  {
    "pages": [
      {
      }
    ]
  }
]
```

SEE ALSO:

- [Use a Mobile Layout for Your Dashboard](#)
- [layouts Properties](#)

## layouts Properties

The `layouts` key specifies the position, order, and size of each widget in the mobile layout. This layout is made up of rows, columns, cells, and pages. Each cell in the grid can contain zero or more widgets. The number of rows, columns, and cells in your mobile layout depend on the number of widgets and the number of pages

## Widget Attributes

These attributes can be set on widgets. Each widget can have zero or more attributes.

Property Name	Details
colspan	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>All widgets</li> </ul> <p><b>Description</b> The number of columns that a widget spans—the width of the widget. If the dashboard doesn't have enough columns to accommodate the specified width, then columns are added to the dashboard.</p> <p><b>Example</b> In this example, the widget named "chart_1" spans 3 columns:</p> <pre>"layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "dimfilter_1   dimfilter_2   dimfilter_3",           "chart_1 {colspan=3}"         ]       }     ]   } ] "version": 1 }</pre>
rowspan	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>All widgets</li> </ul> <p><b>Description</b> The number of rows that a widget spans—the height of the widget. If the dashboard doesn't have enough rows to accommodate the specified height, then rows are added.</p> <p><b>Example</b> In this example, the widget named "dimfilter1_1" spans 2 rows:</p> <pre>"layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "dimfilter_1 {rowspan=2}   dimfilter_2",           "chart_1"         ]       }     ]   } ]</pre>

Property Name	Details
	<pre>     }   ]   "version": 1 } </pre>
zIndex	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>All widgets</li> </ul> <p><b>Description</b> The position of a widget relative to other widgets in the dashboard. <code>zIndex</code> specifies whether a widget is in front of or behind another widget. A smaller <code>zIndex</code> means that a widget appears further behind other widgets with larger <code>zIndex</code> values.</p> <p>The default value of <code>zIndex</code> is 0.</p> <p><b>Example</b> In this example, the widget named <code>"box_1"</code> appears behind the widget named <code>"number_1"</code>:</p> <pre> "layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "box_1 {zIndex=1}, number_1 {zIndex=2}   chart_1"         ]       }     ]   } ] "version": 1 } </pre>
vpad	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>All widgets</li> </ul> <p><b>Description</b> The padding added to the top and bottom sides of the widget's cell in pixels. If <code>vpad</code> equals 10, then 10 pixels are added to the top of the cell and 10 pixels are added to the bottom.</p> <p>The default value of <code>vpad</code> is 0.</p> <p><b>Example</b> In this example, the cell containing widget named <code>"dimfilter_1"</code> has 5 pixels of padding on its top and bottom sides:</p> <pre> "layouts": [   {     "device": "default", </pre>



Property Name	Details
hpad	<pre data-bbox="456 254 1445 552"> "pages": [   {     "rows": [       "dimfilter_1 {vpad=5}"     ]   } ] "version": 1 } </pre> <p data-bbox="415 596 526 659"><b>Type</b> Integer</p> <p data-bbox="415 674 714 707"><b>Available for These Widgets</b></p> <ul data-bbox="456 722 602 756" style="list-style-type: none"> <li>• All widgets</li> </ul> <p data-bbox="415 770 542 804"><b>Description</b></p> <p data-bbox="456 806 1445 905">The padding added to the left and right sides of the widget's cell in pixels. If hpad equals 10, then 10 pixels are added to the left side of the cell and 10 pixels are added to the right side. A negative value can be assigned to</p> <p data-bbox="456 919 760 953">The default value of hpad is 0.</p> <p data-bbox="415 968 509 1001"><b>Example</b></p> <p data-bbox="456 1003 1430 1066">In this example, the cell containing widget named "dimfilter_1" has 5 pixels of padding on its top and bottom sides:</p> <pre data-bbox="456 1081 1445 1470"> "layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "dimfilter_1 {hpad=5}"         ]       }     ]   } ] "version": 1 } </pre>
vAxisWidth	<p data-bbox="415 1518 526 1581"><b>Type</b> Integer</p> <p data-bbox="415 1596 714 1629"><b>Available for These Widgets</b></p> <ul data-bbox="456 1644 570 1677" style="list-style-type: none"> <li>• chart</li> </ul> <p data-bbox="415 1692 542 1726"><b>Description</b></p> <p data-bbox="456 1728 1373 1761">The size of a chart widget's x-axis in pixels. Use vAxisWidth to align multiple chart widgets.</p>

Property Name	Details
	<p><b>Example</b></p> <p>In this example, the widget named "chart_1" has an x-axis that is 250 pixels wide:</p> <pre>"layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "chart_1 {vAxisWidth=250}"         ]       }     ]   } ] "version": 1 }</pre>
hAxisHeight	<p><b>Type</b></p> <p>Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart</li> </ul> <p><b>Description</b></p> <p>The size of a chart widget's y-axis in pixels. Use <code>hAxisHeight</code> to align multiple chart widgets.</p> <p><b>Example</b></p> <p>In this example, the widget named "chart_1" has a y-axis that is 250 pixels tall:</p> <pre>"layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "chart_1 {hAxisHeight=250}"         ]       }     ]   } ] "version": 1 }</pre>

## Row Attributes

These attributes can be set on rows.

Property Name	Details
height	<p><b>Description</b></p> <p>If <code>height</code> is set to a number, then <code>height</code> is the height of a row in pixels.</p> <p>If <code>height</code> is set to <code>preferred</code>, then the row's height is equal to the largest height</p> <p><b>Example</b></p> <p>In this example, the first row's height is 300 pixels. The second row's height is equal to the height of its tallest widget:</p> <pre> "layouts": [   {     "device": "default",     "pages": [       {         "rows": [           "chart_1 {colspan=3}   row:{height=300}",           "dimfilter_1   buttongroup_1   number_1   row:{height=preferred}"         ]       }     ]     "version": 1   } </pre>

## Override Widget Attributes

These attributes can be passed to a widget to override the default behavior of the widget. Note that all color properties take a string, such as "#022B54".

Widget Type	Attributes
Any widget	<ul style="list-style-type: none"> <li>compact</li> <li>placeholder</li> </ul>
ChartWidget	<ul style="list-style-type: none"> <li>legend</li> <li>miniBars</li> <li>fit</li> <li>normalize</li> <li>multiMetrics</li> <li>splitAxis</li> <li>backgroundColor</li> <li>measureAxis (true/false)</li> <li>categoryLabels (true/false)</li> <li>textColor</li> <li>limitBarThickness (true/false)</li> </ul>

Widget Type	Attributes
BoxWidget	<ul style="list-style-type: none"> <li>• backgroundColor</li> <li>• borderColor</li> <li>• imageUrl</li> <li>• stretch</li> </ul>
LinkWidget	<ul style="list-style-type: none"> <li>• destinationType</li> <li>• destination</li> <li>• visualizationType</li> <li>• includeState</li> </ul>
ListSelectorWidget	<ul style="list-style-type: none"> <li>• instant</li> <li>• expanded</li> </ul>
TextWidget	<ul style="list-style-type: none"> <li>• title or text</li> <li>• textColor</li> <li>• textAlignment</li> <li>• fontSize</li> </ul>
PillBoxWidget	<ul style="list-style-type: none"> <li>• backgroundColor</li> <li>• textColor</li> <li>• borderColor</li> <li>• selectedColor</li> </ul>

The following example shows passing attributes to change the background color, text color, and visibility of the axes and labels on chart\_1.

```
"layouts": [
  {
    "device": "default",
    "pages": [
      {
        "rows": [
          "chart_1 {backgroundColor=\"#022B54\", measureAxis=false, categoryLabels=false,
textColor=\"#FFFFFF\"}"
        ]
      }
    ],
    "version": 1
  }
]
```

```
}
]
```

**SEE ALSO:**

[Use a Mobile Layout for Your Dashboard](#)  
[Layouts Specification](#)

**steps**

The `steps` section defines all steps created in and clipped to the dashboard. The properties vary based on whether the dashboard is built using the Wave dashboard designer or classic designer.

[steps Properties for Wave Designer Dashboards](#)

The `steps` key defines all steps available in a Wave designer dashboard. It contains a separate node for each step. Each step node has properties that define the query or list of static values. It also contains properties that control the behavior of the step, like whether to facet the step. The properties and JSON syntax vary based on the step type and whether the step is in compact form or SAQL form.

[steps Properties for Classic Designer Dashboards](#)

The `steps` key defines all steps available in a classic designer dashboard. It contains a separate node for each step. Each step node has properties that define the query or list of static values. It also contains properties that control the behavior of the step, like whether to facet the step.

**steps Properties for Wave Designer Dashboards**

The `steps` key defines all steps available in a Wave designer dashboard. It contains a separate node for each step. Each step node has properties that define the query or list of static values. It also contains properties that control the behavior of the step, like whether to facet the step. The properties and JSON syntax vary based on the step type and whether the step is in compact form or SAQL form.

[steps Properties for Compact Form and SAQL Form](#)

The properties and JSON syntax in the `query` node of the step vary based on whether the step is in compact form or SAQL form.

[aggregateflex Step Type Properties](#)

Use the `aggregateflex` step type to query a Wave dataset. It's the most common step type used to power widgets.

[grain Step Type Properties](#)

Use the `grain` step type for a values table. Values tables have no groupings, just a list of dataset fields to display as columns in the table.

[saql Step Type Properties](#)

Use the `saql` step type for special cases when querying a Wave dataset. With this step type, you can write a custom SAQL query to create derived fields in a values table. You can specify dimensions without groupings. Also, you can bind the dataset name or entire query. For example, you can bind this step type to a static step that provides different SAQL queries or datasets based on a selection.

### [soql Step Type Properties](#)

Use to directly query Salesforce objects—both standard and custom—to get Salesforce data that’s not available in datasets. You can also query external objects created with an OData adapter for Salesforce Connect. To view the results in the dashboard, the user viewing the dashboard must have access to the object and fields queried by the `soql` step.

### [staticflex Step Type Properties](#)

Use the `staticflex` step type to manually define your own set of data. For example, you can use this step to populate a list of static values in a toggle or list widget. It can also be used to provide values to a binding. For example, it can provide possible filters, groups, measures, sort order, and limits.

### [visualizationParameters Properties](#)

The `visualizationParameters` key contains chart properties defined for the step. When you associate the step with a widget, the widget properties override these settings.

### [filters Properties](#)

Use the `filters` property to add a filter to a step query. Although you can create filters for query steps in the user interface, you have to manually define filters for static steps in the dashboard JSON.

## steps Properties for Compact Form and SAQL Form

The properties and JSON syntax in the `query` node of the step vary based on whether the step is in compact form or SAQL form.

### Example: Compact-Form Step for a Wave Designer Dashboard

```

"steps": {
  "Product_StageName_1": {
    "type": "aggregateflex",
    "visualizationParameters": {
      "visualizationType": "hbar",
      "options": {}
    },
    "query": {
      "measures": [
        [
          "sum",
          "Amount"
        ],
        [
          "sum",
          "quantity"
        ]
      ],
      "groups": [
        "Product",
        "StageName"
      ],
      "order": [
        [-1, { "ascending": false }]
      ],
      "aggregateFilters": [[
        [
          "sum",
          "Amount"
        ]
      ]
    ]
  }
}

```

```

        [
          [
            14550720,
            58807698
          ]
        ],
        ">=<="
      ]
    ],
    "isFacet": true,
    "useGlobal": true,
    "isGlobal": false,
    "datasets": [{
      "name": "Flexy_Sales",
      "url": "/services/data/v38.0/wave/datasets/0FbB00000000q5gKAA",
      "id": "0FbB00000000q5gKAA"
    }]
  }
}

```

### Example: SAQL-Form Step for a Wave Designer Dashboard

When the step is in SAQL form, notice how each group and measure are defined in the `groups` and `measures` properties, respectively, and also in the `pigql` property. Other parts of the query—like filters, limits, and order—only need to be defined once in the `pigql` property. You specify the compact form elements of `"groups"` and `"measures"` so that the associated chart widget can render the correct projections.

In the sample step below, notice that the `'sum_Amount'` and `'sum_quantity'` projections in the `pigql` property are referenced in `"measures"` as `[[ "count", "*", "sum_Amount" ], [ "count", "*", "sum_quantity" ]]`. Measure projections in the `pigql` property always include the aggregation, underscore (`_`), and the name of the measure (`'sum_Amount'`) so that they can be referenced in the compact form, as shown here. `"measures": [[ "count", "*", "sum_Amount" ], [ "count", "*", "sum_quantity" ]]`.

```

"steps": {
  "Product_StageName_2": {
    "type": "aggregateflex",
    "visualizationParameters": {
      "options": {}
    },
    "query": {
      "pigql": "q = load \"Flexy_Sales\";\n
              q = group q by ('Product', 'StageName');\n
              q = foreach q generate 'Product' as 'Product',\n
                                   'StageName' as 'StageName',\n
                                   sum('Amount') as 'sum_Amount',\n
                                   sum('quantity') as 'sum_quantity';\n
              q = filter q by 'sum_Amount' >= 14550720 && 'sum_Amount' <=
58807698;\n
              q = order q by 'sum_Amount' desc;\nq = limit q 10000;";
      "measures": [
        [
          "count",
          "*",
          "sum_Amount"
        ],
        [
          "count",
          "*",
          "sum_quantity"
        ]
      ]
    }
  }
}

```

```

        ],
        [
            "count",
            "*",
            "sum_quantity"
        ]
    ],
    "groups": [
        "Product",
        "StageName"
    ]
},
"isFacet": true,
"useGlobal": true,
"isGlobal": false,
"datasets": [{
    "name": "Flexy_Sales",
    "url": "/services/data/v38.0/wave/datasets/0FbB00000000q5gKAA",
    "id": "0FbB00000000q5gKAA"
}]
}
}


```

## aggregateflex Step Type Properties

Use the `aggregateflex` step type to query a Wave dataset. It's the most common step type used to power widgets.

Field Name	Description
<code>datasets</code>	An array of datasets used by this step. Specify the alias of each dataset. If the <code>piqql</code> attribute references a dataset that's not specified here, the dashboard doesn't render.
<code>isFacet</code>	<p>Enables this step to facet and be faceted by other steps. Faceting is when a selection in a widget filters other steps in the dashboard.</p> <p>To enable SAQL-form queries to receive facets from another step, also set the <code>autoFilter</code> option to <code>true</code> in the dashboard JSON. A SAQL-form step without the <code>autoFilter</code> option still emits facets. By default, steps from the same dataset are faceted to each other. To facet steps from different datasets, connect the data sources.</p> <p>To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code>.</p>
<code>isGlobal</code>	<p>Indicates whether the filter that's specified in the query is used as a global filter (<code>true</code>) or not (<code>false</code>). Default is <code>false</code>. You can only apply this property on steps that are connected to a global filter widget—all other steps ignore this property.</p> <p>A global filter widget filters other steps in the dashboard that have <code>useGlobal</code> set to <code>true</code> and reference the same dataset.</p> <p>To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code>.</p>
<code>label</code>	Step label, which is primarily used for display in the designer user interface.



Field Name	Description
query	<p>The query used to retrieve results from a dataset. It must contain at least one grouping and can be in SAQL or compact form. Use a query in SAQL form to customize the query in a way that can't be done using the compact form.</p> <p>For compact form, the query can contain the following properties.</p> <p><b>filters</b> The filters to apply to the data. For more information, see <a href="#">filters Properties</a> on page 65.</p> <p><b>groups</b> The dimension to group by.</p> <p><b>limit</b> The maximum number of results that the step can return. When you create an <code>aggregateflex</code> step, by default, Wave sets <code>limit</code> to 2,000. To return more results, set the <code>limit</code> attribute accordingly. The higher you increase the limit, the longer the query might take. When a limit isn't set, Wave returns up to 10,000 results.</p> <p> <b>Note:</b> The returned results aren't automatically ordered—use this statement only with ordered data.</p> <p><b>measures</b> The measures returned by the query.</p> <p><b>order</b> Sort order (ascending or descending) of the first specified measure. To order the results in ascending order, set <code>ascending</code> to <code>true</code>. To order the results in descending order, set <code>ascending</code> to <code>false</code>. If you don't want to impose a specific order, specify empty brackets this way: <code>"order": []</code>.</p>

#### Compact-form Query Example

```
"query": {
  "filters": [
    [
      "Account.Industry",
      [
        "Agriculture",
        "Apparel",
        "Banking",
        "Biotechnology",
        "Consulting",
        "Education",
        "Electronics",
        "Energy",
        "Engineering",
        "Finance",
        "Healthcare",
        "Insurance",
        "Manufacturing",
        "Media",
        "Retail",
        "Technology",
        "Telecommunications",

```

Field Name	Description
------------	-------------

```

        "Transportation",
        "Utilities"
    ],
    "in"
  ]
},
"groups": [ "Account.Industry" ],
"measures": [
  [
    "avg",
    "Amount"
  ]
],
"order": [
  [
    -1,
    { "ascending": false }
  ]
]
}

```

For SAQL form, the query can contain the following properties

**pigql**

Specify the SAQL query to retrieve data from a dataset. When you specify a SAQL query, you must specify the filters, limits, and ordering inside the `pigql` attribute—Wave ignores the following attributes if they are set under the `query` attribute: `filters`, `limit`, and `order`.

**measures**

Defines the fields included as measures. When using a SAQL-form query, you must include each measure in this parameter and in the `pigql` parameter. You can change the UI label of a measure by setting the `display` option.

```

"count", "*", null, {
  "display": "% of total flights"
}

```

**groups**



Defines the dimension fields to group by. When using a SAQL-form query, you must specify the group-by dimension in this parameter and in the `group` property in the `pigql` parameter.

**SAQL-form Query Example**

```

"query": {
  "pigql": "q = load \"ServiceOpportunity3\";\n
          q = filter q by 'Account.Industry' in
              [\"Agriculture\", \"Apparel\", \"Banking\",
\"Biotechnology\",
              \"Consulting\", \"Education\", \"Electronics\",
              \"Energy\",
              \"Engineering\", \"Finance\", \"Healthcare\",
              \"Insurance\",
              \"Manufacturing\", \"Media\", \"Retail\",
              \"Technology\",

```

Field Name	Description
	<pre>                 \"Telecommunications\", \"Transportation\", \"Utilities\"];\\n                 q = group q by 'Account.Industry';\\n                 q = foreach q generate 'Account.Industry' as 'Account.Industry',                                 count() as 'count';\\n                 q = order q by 'count' desc;\\n                 q = limit q 1000;\";     \"measures\": [         [             \"count\",             \"*\",             \"count\"         ]     ],     \"groups\": [ \"Account.Industry\" ],     \"measuresMap\": {} } </pre> <p>For more information about SAQL queries, see the <a href="#">Wave Analytics SAQL Reference</a></p>
selectMode	<p>Determines the selection interaction. The options for charts are: <code>none</code>, <code>single</code>, and <code>singlerequired</code>. The options for list, range, and toggle selectors are: <code>single</code>, <code>singlerequired</code>, <code>multi</code>, and <code>multirequired</code>.</p> <p> <b>Note:</b> <code>selectMode</code> doesn't apply to number, values table, compare table, date, and global filter widgets.</p>
start	The initial selections that are applied to the step when the dashboard first opens.
type	<p>Step type. Set to <code>aggregateflex</code>. This step type applies to Wave designer dashboards only.</p> <p> <b>Note:</b> If you bind a step property for an <code>aggregateflex</code> step, you must use the bindings syntax for Wave designer. For more information about bindings, see the <a href="#">Wave Analytics Bindings Developer Guide</a>.</p>
useGlobal	<p>Indicates whether to apply global filters to this step (<code>true</code>) or not (<code>false</code>). If the step is in SAQL form, you must also set <code>autoFilter</code> to <code>true</code> to apply the global filters. By default, the global filter widget filters compact-form steps only.</p>
visualizationParameters	Visualization details about the step. For more information, see <a href="#">visualizationParameters Properties</a> .

### Example: `aggregateflex` Step for a Wave Designer Dashboard

```

"Account_Industry_1": {
  "label": "Account Industry",
  "type": "aggregateflex",
  "visualizationParameters": {
    "visualizationType": "hbar",
    "options": {}
  }
}

```

```


},
"query": {
  "filters": [
    [
      "Account.Industry",
      [
        "Agriculture",
        "Apparel",
        "Banking",
        "Biotechnology",
        "Consulting",
        "Education",
        "Electronics",
        "Energy",
        "Engineering",
        "Finance",
        "Healthcare",
        "Insurance",
        "Manufacturing",
        "Media",
        "Retail",
        "Technology",
        "Telecommunications",
        "Transportation",
        "Utilities"
      ],
      "in"
    ]
  ],
  "groups": [ "Account.Industry" ],
  "measures": [
    [
      "avg",
      "Amount"
    ]
  ],
  "order": [
    [
      -1,
      { "ascending": false }
    ]
  ]
},
"isFacet": true,
"useGlobal": true,
"isGlobal": false,
"selectMode": "single"
"datasets": [
  {
    "name": "ServiceOpportunity3",
    "url": "/services/data/v39.0/wave/datasets/0FbR0000000012uKAA",
    "id": "0FbR0000000012uKAA"
  }
]

```

```
  ]
}
```

## grain Step Type Properties

Use the `grain` step type for a values table. Values tables have no groupings, just a list of dataset fields to display as columns in the table.

Field Name	Description
<code>datasets</code>	<p>An array of datasets used by this step. Specify the alias of each dataset. If the <code>piqql</code> attribute references a dataset that's not specified here, the dashboard doesn't render.</p> <p> <b>Note:</b> A <code>grain</code> step can only have one dataset.</p>
<code>isFacet</code>	<p>Enables this step to facet and be faceted by other steps. Faceting is when a selection in a widget filters other steps in the dashboard.</p> <p>To enable SAQL-form queries to receive facets from another step, also set the <code>autoFilter</code> option to <code>true</code> in the dashboard JSON. A SAQL-form step without the <code>autoFilter</code> option still emits facets. By default, steps from the same dataset are faceted to each other. To facet steps from different datasets, connect the data sources.</p> <p>To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code>.</p>
<code>isGlobal</code>	<p>Indicates whether the filter that's specified in the query is used as a global filter (<code>true</code>) or not (<code>false</code>). Default is <code>false</code>. You can only apply this property on steps that are connected to a global filter widget—all other steps ignore this property.</p> <p>A global filter widget filters other steps in the dashboard that have <code>useGlobal</code> set to <code>true</code> and reference the same dataset.</p> <p>To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code>.</p>
<code>label</code>	Step label, which is primarily used for display in the designer user interface.
<code>query</code>	<p>The query used to retrieve results from a dataset. The query can be in compact form only and can contain the following properties:</p> <p><b>filters</b> The filter conditions to apply to the data.</p> <p><b>values</b> List of dataset fields to show as table columns.</p> <p><b>limit</b> The maximum number of results that the step can return. When you create a <code>grain</code> step, by default, Wave sets <code>limit</code> to 100. To return more results, set the <code>limit</code> attribute accordingly. The higher you increase the limit, the longer the query might take. When a limit isn't set, Wave returns up to 10,000 results.</p>

```
"steps": {
  "lens_1": {
```

Field Name	Description
------------	-------------

```

    "type": "grain",
    "visualizationParameters": {
      "visualizationType": "valuestable",
      "options": {}
    },
    "query": {
      "values": [
        "Case.IsEscalated",
        "AccountId",
        "StageName",
        "ForecastCategory",
        "IsClosed",
        "Amount"
      ],
      "limit": 500
    },
    "isFacet": true,
    "useGlobal": true,
    "isGlobal": false,
    "label": "",
    "datasets": [
      {
        "name": "ServiceOpportunity16",
        "url":
"/services/data/v39.0/wave/datasets/0FbB00000000kOSKAY",
        "id": "0FbB00000000kOSKAY"
      }
    ]
  },
},

```

For more information SAQL queries, see the [Wave Analytics SAQL Reference](#).

type	Step type. Set to <code>grain</code> .
useGlobal	Indicates whether to apply global filters to this step ( <code>true</code> ) or not ( <code>false</code> ). If the step is in SAQL form, you must also set <code>autoFilter</code> to <code>true</code> to apply the global filters. By default, the global filter widget filters compact-form steps only.
visualizationParameters	Visualization details about the step. For more information, see <a href="#">visualizationParameters Properties</a> .

### Example: grain Step for a Wave Designer Dashboard

```

"lens_1": {
  "type": "grain",
  "visualizationParameters": {
    "visualizationType": "valuestable",
    "options": {
      "totals": true
    }
  }
}

```

```



    },
    "query": {
      "filters": [
        [
          "Amount",
          [
            [
              1000000,
              7780844
            ],
            ">=<="
          ]
        ]
      ],
      "values": [
        "AccountId",
        "ForecastCategory",
        "CloseDate",
        "Amount",
        "Account.Name",
        "StageName"
      ]
    },
    "isFacet": true,
    "useGlobal": true,
    "isGlobal": false,
    "label": "",
    "datasets": [
      {
        "name": "ServiceOpportunity3",
        "url": "/services/data/v39.0/wave/datasets/0FbR0000000012uKAA",
        "id": "0FbR0000000012uKAA"
      }
    ]
  }
}

```

## saq1 Step Type Properties

Use the `saq1` step type for special cases when querying a Wave dataset. With this step type, you can write a custom SAQL query to create derived fields in a values table. You can specify dimensions without groupings. Also, you can bind the dataset name or entire query. For example, you can bind this step type to a static step that provides different SAQL queries or datasets based on a selection.

Field Name	Description
<code>type</code>	Step type. Set to <code>saq1</code> . This step type applies to Wave designer dashboards only.
<code>label</code>	Step label, which is primarily used for display in the designer user interface.
<code>query</code>	<p>The SAQL query used to retrieve results. When you create a <code>saq1</code>-type step, by default, no <code>limit</code> is set in the query. When a limit isn't set, a step can return up to 10,000 results. To return more results, set the <code>limit</code> attribute accordingly. The higher you increase the limit, the longer the query might take. For more information about SAQL queries, see <a href="#">Wave Analytics SAQL Reference</a>.</p> <p>You can bind a <code>saq1</code> step type to dynamically set the dataset used in the query or change the entire query based on a selection in another step. For example, you can create a toggle based on a static step</p>

Field Name	Description
	that allows the dashboard viewer to select a query. Each toggle option contains a valid SAQL query. Each query can be based on different datasets. (See the example below.)
<code>isFacet</code>	Enables this step to facet and be faceted by other steps. Faceting is when a selection in a widget filters other steps in the dashboard.
<code>useGlobal</code>	Indicates whether to apply global filters to this step ( <code>true</code> ) or not ( <code>false</code> ). If the step is in SAQL form, you must also set <code>autoFilter</code> to <code>true</code> to apply the global filters. By default, the global filter widget filters compact-form steps only.
<code>selectMode</code>	Determines the selection interaction. The options for charts are: <code>none</code> , <code>single</code> , and <code>singlerequired</code> . The options for list, range, and toggle selectors are: <code>single</code> , <code>singlerequired</code> , <code>multi</code> , and <code>multirequired</code> .   <b>Note:</b> <code>selectMode</code> doesn't apply to number, values table, compare table, date, and global filter widgets.
<code>start</code>	The initial selections that are applied to the step when the dashboard first opens.   <b>Note:</b> A widget with a <code>saql</code> -type step can return up to 10,000 results, by default. If Wave doesn't find the initial value in those results, it ignores this setting.

### Example: `saql` Step for a Wave Designer Dashboard

```
"Oppty2ProdSAQL": {
  "type": "saql",
  "label": "Oppty2ProdSAQL",
  "query": "q = load \"Goppty\";\nq = group q by id;\nq = foreach q generate id as
'id', first('Account') as 'Account',s sum(Oppty_Amount) as 'sum_Oppty_Amount';",
  "isFacet": true,
  "useGlobal": true,
  "selectMode": "single"
}
```

To show the Account dimension, which isn't a grouping, the example uses the `first()` function. The function retrieves the first account for each grouping, where the grouping is based on the opportunity ID. Because there's a unique `id` for each record, there's only one record in each grouping. As a result, the function always retrieves the right account for each opportunity. If there were duplicate records for an ID, the query might assign the wrong account to all subsequent opportunities after the first record in the group.


### Example: `saql` Step with a Bound Query for a Wave Designer Dashboard

```
"query": "{{cell(static_1.selection, 0, \"query\")}.asString()"
```

The `static_1` step looks like this:

```
values: [
  {query: "q = load \"opp\"; ..."}
  {query: "q = load \"account\"; ..."}
]
```





 **Tip:** Every dataset referenced in a binding of a `soql` step must be referenced by another step in the dashboard. If not, Wave removes the dataset from the `datasets` attribute in the dashboard JSON. As a result, widgets based on the `soql` step display an error because it can't find the dataset.

## soql Step Type Properties

Use to directly query Salesforce objects—both standard and custom—to get Salesforce data that's not available in datasets. You can also query external objects created with an OData adapter for Salesforce Connect. To view the results in the dashboard, the user viewing the dashboard must have access to the object and fields queried by the `soql` step.

For more information about using Salesforce Connect to access external data, see the [Salesforce Connect](#) online help.

Field Name	Description
<code>type</code>	Step type. Set to <code>soql</code> . This step type applies to Wave designer dashboards only.
<code>label</code>	Step label, which is primarily used for display in the designer user interface.
<code>query</code>	The SOQL query used to retrieve results from a Salesforce object. Because Salesforce—not Wave—executes the query, the maximum number of returned results depends on the SOQL query limit. For more information about SOQL queries, see <a href="#">Force.com SOQL and SOSL Reference</a> .   <b>Note:</b> Every field listed in your SOQL query must be listed in one of the metadata properties: <code>strings</code> , <code>numbers</code> , or <code>groups</code> .
<code>strings</code>	Flags the specified fields as non-grouping dimensions. For example, you can flag a field as a dimension for a values table in which no groupings are allowed.
<code>numbers</code>	Flags the specified fields as measures.
<code>groups</code>	Flags the specified fields as groupings. For example, you can flag a field as a grouping for a pivot table or chart.

 **Note:** The `isFacet` and `useGlobal` step properties don't apply to this step type. You can use a binding to filter other steps based on a selection in a `soql` step.

 **Example:** `soql` Step for a Wave Designer Dashboard

```
"soql": {
  "type": "soql",
  "query": "SELECT Name from ACCOUNT",
  "strings": ["Name"],
  "numbers": [],
  "groups": [],
  "selectMode": "single"
}
```

## staticflex Step Type Properties

Use the `staticflex` step type to manually define your own set of data. For example, you can use this step to populate a list of static values in a toggle or list widget. It can also be used to provide values to a binding. For example, it can provide possible filters, groups, measures, sort order, and limits.

Field Name	Description
<code>type</code>	Step type. Set to <code>staticflex</code> . This step type applies to Wave designer dashboards only.
<code>label</code>	Step label, which is primarily used for display in the designer user interface.
<code>values</code>	Values for the static step. You can have multiple fields for each static value, where each field provides different information about the value, like a label, measurement, or range. When the static step is associated with a widget, the widget uses the first specified field as the display label. You can use other fields to specify values or ranges that you can use to facet or bind steps. For more information about binding a static step to another step, see the <a href="#">Wave Analytics Bindings Developer Guide</a> . For more information about faceting a static step with another data source, see <a href="#">Configure Cross-Dataset Faceting with Connected Data Sources</a> .


Values for the static step. You can have multiple fields for each static value, where each field provides different information about the value, like a label, measurement, or range. When the static step is associated with a widget, the widget uses the first specified field as the display label. You can use other fields to specify values or ranges that you can use to facet or bind steps. For more information about binding a static step to another step, see the [Wave Analytics Bindings Developer Guide](#). For more information about faceting a static step with another data source, see [Configure Cross-Dataset Faceting with Connected Data Sources](#).

If you use the static step wizard to create the step, the step contains the following default fields: `display` and `value`, as shown here. You can change these arbitrary field names and add more fields.

```
"step_static_date_with_start": {
  "type": "staticflex",
  "values": [
    {
      "display": "-6 years",
      "value": [[["year", -6], ["year", 0]]]
    },
    {
      "display": "-5 years",
      "value": [[["year", -5], ["year", 0]]]
    },
    {
      "display": "-4 years",
      "value": [[["year", -4], ["year", 0]]]
    }
  ],
  "isFacet": false,
  "useGlobal": false,
  "isGlobal": false,
  "selectMode": "single",
  "label": "Static Step - Time Periods",
  "start": [[["year", -5], ["year", 0]]]
}
```

Note that the values in each field must have the same datatype, like numbers, strings, or arrays. For instance, if one row has `"value": "123"`, another row can't have `"value": [123]`.

<code>isFacet</code>	Enables this step to facet other steps—it can't be faceted. Faceting is when a selection in a widget filters other steps in the dashboard. To enable faceting, set this property to <code>true</code> and connect this step with another data source.
----------------------	---

Field Name	Description
	To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code> .
<code>isGlobal</code>	Not applicable. You can only apply this property on steps that are connected to a global filter widget—all other steps ignore this property.
<code>useGlobal</code>	Indicates whether to apply global filters to this step ( <code>true</code> ) or not ( <code>false</code> ).
<code>selectMode</code>	Determines the selection interaction. The options for charts are: <code>none</code> , <code>single</code> , and <code>singlerequired</code> . The options for list, range, and toggle selectors are: <code>single</code> , <code>singlerequired</code> , <code>multi</code> , and <code>multirequired</code> .   <b>Note:</b> <code>selectMode</code> doesn't apply to number, values table, compare table, date, and global filter widgets.
<code>start</code>	The initial selections that are applied to the step when the dashboard first opens.

### Example: `staticflex` Step for a Wave Designer Dashboard


```

"static_1": {
  "type": "staticflex",
  "values": [
    {
      "display": "Open",
      "value": "false",
      "won": "false"
    },
    {
      "display": "Won",
      "value": "true",
      "won": "true"
    },
    {
      "display": "Lost",
      "value": "true",
      "won": "false"
    }
  ],
  "isFacet": false,
  "isGlobal": false,
  "selectMode": "single",
  "start": {
    "display": [
      "Open",
      "Won"
    ]
  },
  "useGlobal": false
}

```

## visualizationParameters Properties

The `visualizationParameters` key contains chart properties defined for the step. When you associate the step with a widget, the widget properties override these settings.

Field Name	Description
<code>options</code>	Specifies chart properties for steps clipped to the designer. Wave overrides these options when they are defined in the widget parameters. For more information about chart properties, see <a href="#">Visualizing Data With Charts</a> .
<code>visualizationType</code>	<p>Specifies the chart type. You can override the chart type at the widget level.</p> <p>Valid values for <code>visualizationType</code> are:</p> <ul style="list-style-type: none"> <li>• <code>calheatmap</code>— calendar heat map</li> <li>• <code>choropleth</code> — choropleth (map)</li> <li>• <code>combo</code> — lines and bars to show multiple metrics</li> <li>• <code>comparisontable</code> — comparison table in the classic designer only</li> <li>• <code>flatgauge</code> — flat gauge in the Wave dashboard designer only</li> <li>• <code>funnel</code> — funnel</li> <li>• <code>hbar</code> — horizontal bar</li> <li>• <code>hdot*</code> — horizontal dot plot</li> <li>• <code>heatmap</code>— heat map</li> <li>• <code>matrix</code>— matrix</li> <li>• <code>parallelcoords*</code> — parallel coordinates</li> <li>• <code>pie</code> — donut</li> <li>• <code>pivottable*</code> — pivot table</li> <li>• <code>polargauge</code> — polar gauge in the Wave dashboard designer only</li> <li>• <code>pyramid</code> — pyramid in the Wave dashboard designer only</li> <li>• <code>rating</code> — rating in the Wave dashboard designer only</li> <li>• <code>scatter</code> — scatter plot</li> <li>• <code>stackhbar</code> — stacked horizontal bar</li> <li>• <code>stackpyramid</code> — stacked pyramid in the Wave dashboard designer only</li> <li>• <code>stackvbar</code> — stacked vertical bar</li> <li>• <code>stackwaterfall</code> — stacked waterfall</li> <li>• <code>time</code> — timeline</li> <li>• <code>valuestable</code> — raw data table in the classic designer only</li> <li>• <code>vbar</code> — vertical bar</li> <li>• <code>vdot*</code> — vertical dot plot</li> <li>• <code>waterfall</code> — waterfall</li> </ul> <p> <b>Note:</b> The Wave dashboard designer doesn't support chart types with an asterisk (*). If you specify an unsupported type, the designer replaces it with <code>hbar</code> in the dashboard.</p>

## filters Properties

Use the `filters` property to add a filter to a step query. Although you can create filters for query steps in the user interface, you have to manually define filters for static steps in the dashboard JSON.

The syntax for a filter in the step definition varies based on whether the step is in compact or SAQL form. This section describes the filter syntax for compact-form steps, including a description and example of every operator. For information about the filters for SAQL-form steps, see the [Wave Analytics SAQL Reference](#).

### Filter Syntax

Filters defined in compact-form steps have the following syntax.

```
"filters": [[
  "field",
  [value],
  "operator"
]]
```

For example, the following filter shows only records with a "Customer" account type.

```
"filters": [[
  "Account_Type",
  ["Customer"],
  "in"
]]
```

To compare against multiple values, include the values in an array, like this.


```
"filters": [[
  "field",
  [[value1, value2, value3]],
  "operator"
]]
```

To specify an absolute date value for a date filter, specify the value in epoch format, where the value is the number of milliseconds since January 1, 1970 midnight UTC (1970-01-01 00:00:00). The following example shows dataset rows with a close date on or before January 1, 2016.

```
"filters": [[
  "Close Date",
  [[
    1451606400000,
    null
  ]],
  ">="
]]
```



### Operators

You can use different operators in a filter. The supported operators depend on the field type. If you don't specify the operator, Wave applies the `==` operator.

Operator	Description	Compact-Form Example
in	Value of dataset field equals one of the specified values. Applies to dimensions only.	<code>["Dimension", ["Value1", "Value2"], "in"]</code>
not in	Value of dataset field is not in the specified list of values. Applies to dimensions only.	<code>["Dimension", ["Value1", "Value2"], "not in"]</code>
matches	Value of dataset field contains the specified value. This operator is not case-sensitive. Applies to dimensions only.	<code>["Dimension", ["Val"], "matches"]</code>
is null	Value of dataset field is null. Applies to measures only.	<code>["Amount", [], "isnull"]</code>
is not null	Value of dataset field is not null. Applies to measures only.	<code>["Amount", [], "isnotnull"]</code>
==	Value of dataset field equals the specified value. Applies to measures only.	<code>["Measure", [1], "=="]</code>
!=	Value of dataset field does not equal the specified value. Applies to measures only.	<code>["Measure", [1], "!="]</code>
<	Value of dataset field is less than the specified value. Applies to measures only.	<code>["Measure", [10], "&lt;"]</code>
>	Value of dataset field is greater than the specified value. Applies to measures only.	<code>["Measure", [1], "&gt;"]</code>
<=	Value of dataset field is less than or equal to the specified value. Applies to measures and absolute dates only.	<ul style="list-style-type: none"> <li><code>["Measure", [10], "&lt;="]</code></li> <li><code>["Date", [1229040000000], "&lt;="]</code></li> </ul>
>=	Value of dataset field is greater than or equal to the specified value. Applies to measures and absolute dates only.	<ul style="list-style-type: none"> <li><code>["Measure", [1], "&gt;="]</code></li> <li><code>["Date", [-374457600000], "&gt;="]</code></li> </ul>
>=<= (between)	<p>Value of dataset field is between the specified values, inclusive. For relative dates, you can specify the following time periods: "year", "quarter", "month", "week", and "day". Applies to measures and dates only.</p> <p> <b>Note:</b> You can also use:</p> <p><code>&gt;=&lt;</code> Greater than or equal to one value, but less than another.</p> <p><code>&gt;&lt;=</code> Greater than one value, but less than or equal to another.</p> <p><code>&gt;&lt;</code> Between two values, exclusive.</p>	<ul style="list-style-type: none"> <li><code>["Measure", [1, 10], "&gt;=&lt;="]</code></li> <li><code>["Date", [-374457600000, 122904000000], "&gt;&lt;="]</code></li> <li><code>["Date", [{"month", -1}, {"month", 1}], "&gt;&lt;="]</code></li> <li><code>["Close Date", [null, 1451606400000], "&lt;="]</code></li> </ul>

## steps Properties for Classic Designer Dashboards

The `steps` key defines all steps available in a classic designer dashboard. It contains a separate node for each step. Each step node has properties that define the query or list of static values. It also contains properties that control the behavior of the step, like whether to facet the step.

Field Name	Description
<code>datasets</code>	<p>An array of datasets used by this step. Specify the alias of each dataset. If the <code>pgsql</code> attribute references a dataset that's not specified here, the dashboard doesn't render.</p> <p> <b>Note:</b> Faceted steps are filtered based on only the first dataset specified in this array.</p>
<code>visualizationParameters</code>	<p>Visualization details about the step. For more information, see <a href="#">visualizationParameters Properties</a>.</p>
<code>isFacet</code>	<p>Enables this step to facet and be faceted by other steps. Faceting is when a selection in a widget filters other steps in the dashboard. To enable SAQL-form queries to receive facets from another step, also set the <code>autoFilter</code> option to <code>true</code>. A SAQL-form step without the <code>autoFilter</code> option still emits facets. By default, steps from the same dataset are faceted to each other. To facet steps from different datasets, connect the data sources.</p> <p>Steps with a cogrouping that query multiple datasets only facet steps based on the first dataset specified in the <code>datasets</code> field.</p> <p>To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code>.</p>
<code>isGlobal</code>	<p>Indicates whether the filter that's specified in the query is used as a global filter (<code>true</code>) or not (<code>false</code>). Default is <code>false</code>. You can only apply this property on steps that are connected to a scope widget—all other steps ignore this property.</p> <p>A scope widget filters other steps in the dashboard that have <code>useGlobal</code> set to <code>true</code> and reference the same dataset. By default, it filters compact-form steps only. To filter a SAQL step, set <code>autoFilter</code> to <code>true</code> in the SAQL step.</p> <p>To avoid unexpected behavior, do not set both <code>isFacet</code> and <code>isGlobal</code> to <code>true</code>.</p>
<code>query</code>	<p>The query that the step uses. It can be in SAQL or compact form.</p>
<code>selectMode</code>	<p>Determines the selection interaction. The options for charts are: <code>none</code>, <code>single</code>, and <code>singlerequired</code>. The options for list and toggle selectors are: <code>single</code>, <code>singlerequired</code>, <code>multi</code>, and <code>multirequired</code>.</p> <p> <b>Note:</b> <code>selectMode</code> isn't used by the number, values table, compare table, range, date, and global filter widgets.</p>
<code>start</code>	<p>The initial selections that are applied to the step when the dashboard first opens.</p>
<code>type</code>	<p>The type can be set to:</p> <ul style="list-style-type: none"> <li>• <code>grain</code>—Use to build a values table.</li> <li>• <code>multi</code>—Use to build a compare table in a classic designer dashboard.</li> <li>• <code>static</code>—Use to build a static step.</li> <li>• <code>aggregate</code>—Use for all other steps.</li> </ul>

Field Name	Description
useGlobal	Indicates whether the step uses the dashboard's scope widget ( <code>true</code> ) or not ( <code>false</code> ).
dimensions	<p>The dimension used to facet other steps. Wave facets other steps based on the value selected for this dimension in the user interface. Specify the <code>dimensions</code> attribute only if <code>isFacet</code> is set to <code>true</code>.</p> <p>Example:</p> <pre> "step_filter_dim": {   "type": "static",   "dimensions": [ "Product" ],   "datasets": [{"name": "opportunity"}],   "selectMode": "single",   "values": [     {"value": ["EKG Machine"]},     {"value": ["Mammography Machine"]},     {"value": ["Ultrasound Machine"]}   ],   "isFacet": true }, </pre>
values	<p>Values used to filter the results of a static step. For example, you can use these values to populate a date selector.</p> <pre> "step_date_static_with_start": {   "type": "static",   "values": [     {       "display": "-6 years",       "value": [[["year", -6], ["year", 0]]]     },     {       "display": "-5 years",       "value": [[["year", -5], ["year", 0]]]     },     {       "display": "-4 years",       "value": [[["year", -4], ["year", 0]]]     }   ],   "selectMode": "singlerequired",   "start": [[["year", -5], ["year", 0]]] } </pre> <p>When you define a static step, you can create any fields and values under <code>values</code>. To bind the static values to another step, the binding can reference any of the fields to retrieve the values. For more information about binding a static step, see the <a href="#">Wave Analytics Bindings Developer Guide</a>.</p>



### Example: **aggregate Step with Bindings for a Classic Designer Dashboard**

If you convert a classic designer dashboard to Wave designer, `aggregate` steps are not converted to `aggregateflex`. They continue to use the bindings syntax from the classic designer, which is different from the Wave designer syntax. The Wave designer supports both syntaxes.







```

"steps": {
  "step_Account_Name_1": {
    "isFacet": false,
    "query": {
      "pigql": "q = load \"opp\";\n
              q = filter q by 'Account-Name' in {{
selection(step_Account_Owner_Name_2) }};\n
              q = group q by {{ single_quote(value(selection(step_StageName_3)))
}};\n
              q = foreach q generate
                  {{ single_quote(value(selection(step_StageName_3))) }}
as {{ value(selection(step_StageName_3)) }},
                  sum('Amount') as 'sum_Amount',
                  count() as 'count'",
      "groups": "{{ selection(step_StageName_3) }}",
      "measures": [
        ["sum", "Amount"]
      ]
    },
    "visualizationParameters": {
      "visualizationType": "hbar"
    },
    "selectMode": "none",
    "useGlobal": true,
    "datasets": [{
      "name": "opp"
    }],
    "type": "aggregate",
    "isGlobal": false
  }
}

```

### Example: **SAQL Query Step for a Compare Table for a Classic Designer Dashboard**

This example shows a compare table step for a classic designer dashboard for a mobile client. The `globalQuery` definition under `globalQuery` contains a single, unified SAQL query for creating this simple, two-column compare table.

Industry	Sum of LeadScore	Avg of LeadScore
High Tech		3.0679
Fin Svcs		3.1796
Mfg		2.8361
Healthcare		3.5238
Prof Svcs		3.4258
Consumer		2.3604

```

"compare_2": {
  "isFacet": true,
  "isGlobal": false,
  "selectMode": "single",
  "type": "multi",
  "useGlobal": true,
  "start": null,
  "datasets": [
    {
      "name": "Honeywell_Recent_Deals1"
    }
  ],
  "visualizationParameters": {
    "visualizationType": "comparisontable"
  },
  "columns": [
    {
      "header": "Sum of LeadScore",
      "query": {
        "measures": [
          [
            "max",
            "LeadScore"
          ]
        ],
        "groups": [
          "Industry"
        ]
      },
      "showBars": true
    },
    {
      "header": "Avg of LeadScore",
      "query": {
        "measures": [
          [
            "avg",
            "LeadScore"
          ]
        ]
      }
    }
  ]
}

```

```

    ],
    "groups": [
      "Industry"
    ]
  },
  "showBars": false
}
],
"globalQuery": {
  "pigql": "q = load \"Honeywell_Recent_Deals1\"; q = group q by 'Industry'; q
= filter q by 'Industry' in [\"Consumer\", \"Fin Svcs\", \"Mfg\", \"High
Tech\", \"Healthcare\", \"Prof Svcs\"]; q = foreach q generate 'Industry' as 'Industry',
avg('LeadScore') as 'avg_LeadScore', sum('LeadScore') as 'sum_LeadScore'; q = limit
q 2000;"
}
},

```

 **Note:** The compare table has the following limitations.

- Only these functions can be included: +, -, \*, /, ().
- On mobile devices, do not use SAQL at the column level. You can use a global SAQL query or use the compact form in each column.
- On mobile devices, the Compare Table is read-only.

For more information about SAQL, see the [SAQL Reference](#).


### query Properties

The `query` key contains the query parameters for compact form and SAQL form steps. Steps created in the explorer or dashboard designer are created in compact form. Steps created in the SAQL editor are created in SAQL form.

## query Properties


The `query` key contains the query parameters for compact form and SAQL form steps. Steps created in the explorer or dashboard designer are created in compact form. Steps created in the SAQL editor are created in SAQL form.

Non-static steps retrieve data based on a query. The `query` key under the step node defines the query parameters, like measures, filters, groupings, limits, and sort order. The structure of the query and the properties vary based on whether the step is created in compact form and SAQL form. For more information about steps, see [Create Steps in the Wave Dashboard Designer](#).

 **Note:** You can dynamically set query properties based on the selection or results of another step. For example, you can change a grouping based on a selection in a toggle widget. For more information, see the [Wave Analytics Bindings Developer Guide](#).

The properties of the `query` section of a dashboard JSON file are:

Field Name	Description
<code>autoFilter</code>	Enables filters from compact-form query steps and scope/global filter widgets to be applied to the faceted SAQL query step. To apply filters from compact-form query steps to the SAQL query step, set <code>autoFilter</code> and <code>isFacet</code> to <code>true</code> . To apply filters from Scope widgets to the SAQL query step, set <code>autoFilter</code> and <code>useGlobal</code> to <code>true</code> . If <code>autoFilter</code> is set to <code>false</code> or not specified, filters from compact-form query steps and Scope widgets are not applied to the SAQL query step.

Field Name	Description
dimensions	<p>The dimensions to use are specified this way:</p> <pre>"dimensions": [ "Department" ]</pre>
filters	<p>The filter conditions to apply to the data. Here's an example of a simple filter condition to include only rows that have the destination "SFO", "LAX", "ORD", or "DFW":</p> <pre>"filters": [{"dest", ["SFO", "LAX", "ORD", "DFW"]}]</pre> <p> <b>Note:</b> Applies to steps with compact form queries only. To specify a filter for a step based on a SAQL query, include a <code>filter</code> statement in the SAQL query.</p>
formula	<p>Formula is used with the <i>multi</i> step type in a step for a compare table. A <i>multi</i> type step includes multiple subqueries. You can use the basic mathematical operators <code>*</code>, <code>/</code>, <code>-</code>, <code>+</code>, <code>(</code>, and <code>)</code> to create a formula to reference other subqueries in the step. To reference other subqueries, use the automatically assigned names: "A" is the first query, "B" is the second query, and so on.</p> <pre>"step_comptable": {   "type": "multi",   "datasets": [{"name": "opp"}],   "isFacet": true,   "useGlobal": true,   "query": {     "columns": [       {         "header": "Opptys Won",         "query": {           "pigql": null,           "filters": [{"StageName", ["5 - Closed-Won"]}, ["Close Date", [{"year", -1}, {"year", 0}]]],           "measures": [{"count", "*"}],           "values": [],           "groups": ["Owner-Name"],           "formula": null,           "order": []         }       }, {         "header": "Opptys Won (\$)",         "query": {           "pigql": null,           "filters": [{"StageName", ["5 - Closed-Won"]}],           "measures": [{"sum", "Amount"}],           "values": [],           "groups": ["Owner-Name"],           "formula": null,           "order": []         }       }     ], {       "sort": {</pre>

Field Name	Description
------------	-------------

```

      "asc": false,
      "inner": false
    },
    "header": "Opptys Won ($)",
    "showBars": true,
    "query": {
      "pigql": null,
      "filters": [{"StageName", ["5 - Closed-Won"]}],


      "measures": [{"sum", "Amount"}],
      "values": [],
      "groups": ["Owner-Name"],
      "formula": null,
      "order": []
    }
  }, {
    "header": "Opptys Lost ($)",
    "query": {
      "pigql": null,
      "filters": [{"StageName", ["5 - Closed-Lost"]}],



      "measures": [{"sum", "Amount"}],
      "values": [],
      "groups": ["Owner-Name"],
      "formula": null,
      "order": []
    }
  }, {
    "header": "Opptys Lost ($)",
    "showBars": true,
    "query": {
      "pigql": null,
      "filters": [{"StageName", ["5 - Closed-Lost"]}],

      "measures": [{"sum", "Amount"}],
      "values": [],
      "groups": ["Owner-Name"],
      "formula": null,
      "order": []
    }
  }, {
    "header": "Win-Loss (%)",
    "query": {
      "groups": ["Owner-Name"],
      "filters": [{"StageName", ["5 - Closed-Lost"]}],

      "measures": [{"sum", "Amount"}],
      "values": [],
      "pigql": null,
      "formula": "B/(B+D)*100",
      "order": []
    }
  }

```

Field Name	Description
	<pre>       }     ]   } } }, </pre>
groups	<p>The dimension to group by. For example, "groups": ["carrier"]. Specify groups for both compact form and SAQL form queries. To group by a dimension when using a SAQL form query, you must specify the group-by dimension in this parameter and in the SAQL query in the <code>piqq1</code> parameter.</p>
limit	<p>The number of results to return. For example, "limit": 10. The results that the limit statement returns aren't automatically ordered, so use this statement only with data that has been ordered.</p> <p> <b>Note:</b> Applies to steps with compact form queries only. To specify a limit for a step based on a SAQL query, include a <code>limit</code> statement in the SAQL query.</p>
measures	<p>The measures to use are specified this way:</p> <pre> "count", "*", null, {   "display": "% of total flights" } </pre> <p>Specify for both compact form and SAQL form queries. Specify for SAQL queries so that the associated chart widget can render the correct projections. You can change the UI label of a measure by setting the <code>display</code> option.</p> <p>To add a measure when using a SAQL form query, specify the measure in this parameter and in the SAQL query in the <code>piqq1</code> parameter.</p>
order	<p>Sorts the first specified measure in ascending or descending order. To order the results in ascending order, set <code>ascending</code> to <code>true</code>. To order the results in descending order, set <code>ascending</code> to <code>false</code>. If you don't want to impose a specific order, specify empty brackets this way: "order": [].</p> <p>Example:</p> <pre> "step1": {   "type": "aggregate",   "datasets": [{"name": "airline"}],   "query": {     "groups": ["dest"],     "filters": [       ["carrier", "{{ selection(step1) }}"],       ["dest", "{{ filter(step1, 'dest') }}"],       ["origin", "{{ filter(step1, 'origin') }}"]     ],     "measures": [{"sum", "miles"}, {"count", "*"}],     "order": [[-1, {"ascending": false}]]   } } </pre>

Field Name	Description
	<p> <b>Note:</b> Applies to steps with compact form queries only. To specify order for a step based on a SAQL query, include an <code>order</code> statement in the SAQL query.</p>
<code>piqq1</code>	<p>The query in SAQL form. Use a query in SAQL form when you need to customize the query in a way that can't be done using the compact form.</p> <p>When you specify a SAQL query, you must specify the filters, limits, and ordering inside the <code>piqq1</code> attribute—Wave ignores the following attributes if they are set under the <code>query</code> attribute: <code>filters</code>, <code>limit</code>, and <code>order</code>. You must include each measure in the SAQL query and also specify it in the <code>measures</code> attribute. To specify a grouping, include a group by statement in the SAQL query and specify the same dimension in the <code>groups</code> attribute.</p> <p> <b>Note:</b> You can enable faceting on a step created from a SAQL query. However, if the SAQL query is based on multiple datasets, only the first dataset specified in the <code>datasets</code> field is faceted.</p>
<code>values</code>	<p>Values are used with the <code>grain</code> steps for a values table widget or in <code>static</code> steps. Values in a <code>grain</code> step list the columns to include in the values table. For example:</p> <pre> "step_grain": {   "type": "grain",   "datasets": [{"name": "opp"}],   "query": {     "values": ["Amount", "Owner-Name", "Name", "Account-Name", "StageName", "ForecastCategory", "Current Age", "Time to Win"],   } } </pre> <p>You manually define the values in a <code>static</code> step. You can include <code>values</code> in both compact form and SAQL form queries.</p>

## widgets

The `widgets` section defines the widgets that appear in the dashboard. Each widget has a name.

### Example: Widgets in a Wave Designer Dashboard

```

"widgets": {
  "text_1": {
    "parameters": {
      "fontSize": 20,
      "text": "Grouping",
      "textAlignment": "center",
      "textColor": "#091A3E"
    },
    "type": "text"
  },
  "pillbox_1": {
    "parameters": {

```

```
"compact": false,
"exploreLink": false,
"step": "StaticSQLMinRanges"
},
"type": "pillbox"
},
"chart_1": {
  "parameters": {
    "autoFitMode": "fit",
    "showValues": true,
    "legend": {
      "showHeader": true,
      "show": true,
      "position": "right-top",
      "inside": false
    },
    "axisMode": "multi",
    "visualizationType": "hbar",
    "exploreLink": true,
    "title": {
      "label": "",
      "align": "center",
      "subtitleLabel": ""
    },
    "trellis": {
      "enable": false,
      "type": "x",
      "chartsPerLine": 4
    },
    "measureAxis2": {
      "showTitle": true,
      "showAxis": true,
      "title": ""
    },
    "measureAxis1": {
      "showTitle": true,
      "showAxis": true,
      "title": ""
    },
    "theme": "wave",
    "step": "Account_BillingCount_1",
    "dimensionAxis": {
      "showTitle": true,
      "showAxis": true,
      "title": ""
    }
  },
  "type": "chart"
}
}
```



 **Example: Widgets in a Classic Designer Dashboard**

```
"widgets": {
  "chart_1": {
    "type": "chart",
    "position": {
      "zIndex": 3,
      "x": 10,
      "y": 80,
      "w": 500,
      "h": 300
    },
    "parameters": {
      "step": "Match_Status_3",
      "legend": true,
      "visualizationType": "pie"
    }
  },
  "text_1": {
    "type": "text",
    "position": {
      "zIndex": 4,
      "x": 0,
      "y": 10
    },
    "parameters": {
      "text": "We took a look at your current accounts and \ncontacts and found the following matches below",
      "textAlignment": "left"
    }
  },
  "number_3": {
    "type": "number",
    "position": {
      "zIndex": 5,
      "x": 510,
      "y": 90,
      "w": 550
    },
    "parameters": {
      "step": "all_4",
      "measureField": "count",
      "title": "Accounts and Contacts from your org were analyzed",
      "fontSize": 42,
      "textAlignment": "left"
    }
  },
  "chart_2": {
    "type": "chart",
    "position": {
      "zIndex": 7,
      "x": 10,
      "y": 440,
      "w": 500,
      "h": 490
    }
  }
}
```

```
    },
    "parameters": {
      "step": "SIC_Desc_Match_Status_5",
      "visualizationType": "hbar"
    }
  },
  "text_3": {
    "type": "text",
    "position": {
      "zIndex": 8,
      "x": 10,
      "y": 390
    },
    "parameters": {
      "text": "What does this look like by SIC ?"
    }
  },
  "chart_3": {
    "type": "chart",
    "position": {
      "zIndex": 10,
      "x": 520,
      "y": 240,
      "w": 500,
      "h": 300
    },
    "parameters": {
      "step": "Match_Status_6",
      "visualizationType": "hbar"
    }
  },
  "text_4": {
    "type": "text",
    "position": {
      "zIndex": 11,
      "x": 500,
      "y": 200
    },
    "parameters": {
      "text": "What's the value of these to you?"
    }
  },
  "chart_4": {
    "type": "chart",
    "position": {
      "zIndex": 12,
      "x": 530,
      "y": 440,
      "w": 500,
      "h": 300
    },
    "parameters": {
      "step": "SIC_Desc_7",
      "sqrt": true,

```

```

    "visualizationType": "vbar"
  }
},
"text_5": {
  "type": "text",
  "position": {
    "zIndex": 13,
    "x": 520,
    "y": 390
  },
  "parameters": {
    "text": "Where do you make your money today?"
  }
},
"text_6": {
  "type": "text",
  "position": {
    "zIndex": 16,
    "x": 1050,
    "y": 20
  },
  "parameters": {
    "text": "The future of data..."
  }
}
}
}

```

### widget Properties

The `widgets` key defines all widgets that are available in the dashboard. It contains a separate node for each widget. Each widget appears in all layouts to which it's added. The properties available for each widget depend on the widget type. For example, a chart widget has the `legend` property, but a text widget doesn't.

## widget Properties

The `widgets` key defines all widgets that are available in the dashboard. It contains a separate node for each widget. Each widget appears in all layouts to which it's added. The properties available for each widget depend on the widget type. For example, a chart widget has the `legend` property, but a text widget doesn't.

Field Name	Description
<code>parameters</code>	Widget parameters vary depending on the type of widget and, if applicable, type of chart. The <code>step</code> element defines the step attached to a widget. For detailed information about different widget parameters, see <a href="#">parameters Properties</a> .
<code>position</code>	(For classic designer dashboards only.) Specifies the position of the widget in the dashboard. Position can consist of the following properties: <ul style="list-style-type: none"> <li><b>x and y</b> Specifies the top left corner of the widget. The values of these fields must be integers.</li> </ul>

Field Name	Description
------------	-------------

**w and h**

Specifies the width and height, respectively. You can enter "auto," percentages ("36%"), and integers ("20") as a string value.

**zIndex**

Determines the position of a widget relative to other widgets in the dashboard. `zIndex` specifies whether a widget is in front of or behind another widget. A smaller `zIndex` means that a widget appears further behind other widgets with larger `zIndex` values. The value must be an integer.

Example:

```
"position": {
  "x": 40,
  "y": 40,
  "w": "300",
  "h": "auto"}
```

Measurements are in pixels.



**Note:** The Wave dashboard designer ignores these settings and uses the `position` attribute specified under the `gridLayouts` section of the dashboard JSON.

<code>type</code>	The widget type specifies one of the other supported widget types. The value of this field must be a string.
-------------------	--

- `box`—available in the classic designer only
- `chart`
- `comparabletable`
- `container`—available in the Wave dashboard designer only
- `dateselector`
- `globalfilters`
- `image`—available in the Wave dashboard designer only
- `link`
- `listselector`
- `number`
- `pillbox`
- `rangeselector`
- `table`—available in the Wave dashboard designer only
- `text`
- `url`—available in the classic designer only
- `valuabletable`



**Note:** The Wave dashboard designer doesn't support `box` and `url` widgets. The designer removes these widget types when you open the dashboard. Also, the classic designer doesn't support the `container` widget—use a `box` widget instead.

### parameters Properties

The `parameters` key contains a list of properties that control the appearance of the widget. Each widget type, including each chart type, contains a unique set of properties.

## parameters Properties

The `parameters` key contains a list of properties that control the appearance of the widget. Each widget type, including each chart type, contains a unique set of properties.


 **Note:** You can dynamically set properties for number and chart widgets in Wave designer dashboards based on the selection or results of another step. For example, you can change the map type in a chart based on a selection in a toggle widget. For more information, see the [Wave Analytics Bindings Developer Guide](#).

Chart widgets have many properties that vary based on the chart type. For a list of properties for each chart in a classic designer dashboard, see the following table. Wave designer doesn't support some of the chart properties listed below. For chart-specific properties for Wave designer dashboards, see [Visualizing Data with Charts](#)—this section doesn't cover chart properties for Wave designer dashboards.

Visualization Type	Valid Properties
Bar	<code>legend</code> , <code>legendHideHeader</code> , <code>legendWidth</code> , <code>maxColumnWidth</code> , <code>minColumnWidth</code> , <code>miniBars</code> , <code>multiMetrics</code> , <code>splitAxis</code> , <code>sqrt</code> , and <code>trellis</code>
Compare Table	<code>maxColumnWidth</code> and <code>minColumnWidth</code>
Donut	<code>legend</code> , <code>legendHideHeader</code> , and <code>legendWidth</code>
Dot Plot	<code>fit</code> , <code>legend</code> , <code>legendHideHeader</code> , <code>legendWidth</code> , and <code>sqrt</code>
Heat Map	<code>legend</code> , <code>legendHideHeader</code> , and <code>legendWidth</code>
Matrix	<code>legend</code> , <code>legendHideHeader</code> , and <code>legendWidth</code>
Parallel Coordinates	<code>fit</code> , <code>legend</code> , <code>legendHideHeader</code> , <code>legendWidth</code> , and <code>sqrt</code>
Pivot Table	<code>maxColumnWidth</code> , <code>minColumnWidth</code> , and <code>totals</code>
Scatter Plot	<code>fit</code> , <code>legend</code> , <code>legendHideHeader</code> , <code>legendWidth</code> , and <code>sqrt</code>
Stacked Bar/Column	<code>legend</code> , <code>legendHideHeader</code> , <code>legendWidth</code> , <code>maxColumnWidth</code> , <code>minColumnWidth</code> , <code>miniBars</code> , <code>normalize</code> , and <code>sqrt</code>
Timeline	<code>fit</code> , <code>legend</code> , <code>legendHideHeader</code> , <code>legendWidth</code> , and <code>sqrt</code>
Values Table	<code>hideHeaderColumn</code> , <code>maxColumnWidth</code> , and <code>minColumnWidth</code>

The widget properties set by the `parameters` property are:

Property Name	Details
alignmentX	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• image</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates the horizontal alignment of the image in the widget. Valid values are: left (default), center, and right.</p>
alignmentY	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• image</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates the vertical alignment of the image in the widget. Valid values are: top (default), center, and bottom.</p>
calendarTypeSwitchingAllowed	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• dateselector</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) Indicates whether the dashboard viewer can switch between the fiscal and calendar year in the date widget. Default is false.</p>
compact	<p><b>Type</b> Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• listselector</li> <li>• number</li> <li>• pillbox</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>



Property Name	Details
	<p><b>Description</b></p> <p>Indicates whether displayed numbers are abbreviated (<code>true</code>) or not (<code>false</code>).</p> <p>For example, if <code>true</code>, the number 48,081 appears as 48k. Although the number appears to be rounded, it is not. The value 48,081 is preserved in charts and when performing calculations. If <code>false</code>, then 48,081 appears as 48,081.</p> <p>Default is <code>false</code>.</p>
computeTotal	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code> (only when <code>visualizationType</code> is <code>stackwaterfall</code> and <code>waterfall</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Indicates whether to include the total measure column (<code>true</code>) or not (<code>false</code>).</p> <p>Default is <code>true</code>.</p>
containedWidgets	<p><b>Type</b></p> <p>List</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>container</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>A list of all widgets inside the container widget.</p> <p><b>Example</b></p> <p>This example shows 2 widgets (<code>meafilter_1</code> and <code>chart_1</code>) included in the container widget (<code>container_1</code>).</p> <pre data-bbox="646 1451 1445 1860"> "container_1": {   "type": "container",   "position": {     "x": 0,     "y": 0   },   "parameters": {     "containedWidgets": [       "meafilter_1",       "chart_1"     ]   } } </pre>

Property Name	Details
customBulkActions	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• table</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) Specifies the following details about custom bulk action.</p> <p><b>label</b> Display label for the button in the table widget's action menu. The dashboard viewer clicks the button to execute the action.</p> <p><b>visualforce</b> The name and namespace prefix of the Visualforce page that defines the bulk action. Namespace prefix is optional.</p> <p><b>Example</b></p> <pre> "customBulkActions":   [     {       "label": "Generate Opportunities",       "visualforce":         {           "name": "VF_Create_Opportunities",           "namespace": "Prefix"         }     }   ] </pre>
defaultAbsoluteMode	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• dateselector</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) Indicates whether the date widget displays absolute dates, by default. If not, then it shows relative dates. Default is <code>true</code>.</p>
defaultFiscalMode	<p><b>Type</b> Boolean</p>





Property Name	Details
	<p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>dateselector</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>(For Wave designer dashboards only.) Indicates whether the date widget displays dates using the fiscal year, by default. If not, then it uses the calendar year.</p> <p>Default is <code>false</code>.</p>
<code>destination</code>	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>link</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>The ID of the dashboard, lens, or step.</p> <p>Default is <code>null</code>.</p>
<code>destinationType</code>	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>link</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>The destination type of a link. Possible values are:</p> <ul style="list-style-type: none"> <li>• <code>dashboard</code> — a saved dashboard</li> <li>• <code>explore</code> — an unsaved, active exploration session of the lens</li> <li>• <code>lens</code> — a saved lens</li> </ul> <p>Default is <code>lens</code>.</p>
<code>displayTemplate</code>	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>listselector</code></li> <li>• <code>pillbox</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>

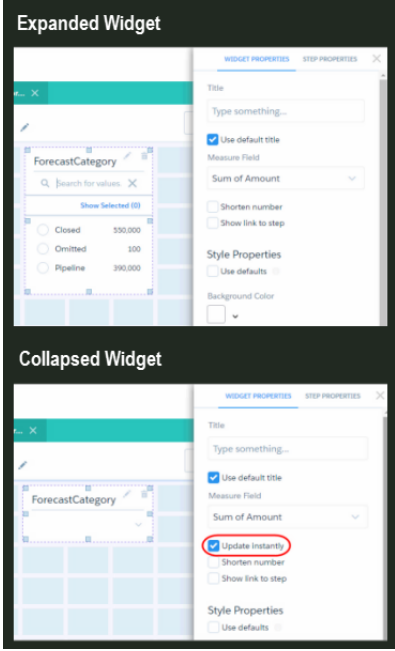

Property Name	Details
	<p><b>Description</b></p> <p>(For Wave designer dashboards only.) Specifies the string of grouping and measure fields to display in the widget. Fields must be enclosed in square brackets. By default, all groupings are included.</p> <p><b>Example</b></p> <p>This example displays the value for the Account.Type dimension, Account.BillingCountry dimension, and Amount measure.</p> <pre data-bbox="651 548 1446 625">"displayTemplate": "[Account.Type] - [Account.BillingCountry] ([avg_Amount])"</pre>
documentId	<p><b>Type</b></p> <p>String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• image</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>The 15-character document Id of the image file that you want to apply as the background. To ensure security, the image file must be uploaded to Salesforce as a document and the <b>Externally Available Image</b> option must be selected. If this option is not selected or the referenced document is not an image, the image doesn't show up in the widget. Default is null.</p> <p><b>Example</b></p> <p>This example image widget (image_1) displays an image with ID 015R0000000DC1P.</p> <pre data-bbox="651 1272 1446 1560">"image_1": {   "type": "image",   "parameters": {     "documentId": "015R0000000DC1P",     "fit": "stretch",     "alignmentX": "center",     "alignmentY": "center"   } }</pre>
dualAxis	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart (only when visualizationType is combo)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p>




Property Name	Details
	<p><b>Description</b> Indicates whether to include an axis for each of the two measures (<code>true</code>) or not (<code>false</code>). Default is <code>true</code>.</p>
expanded	<p><b>Type</b> Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• <code>dateselector</code></li> <li>• <code>listselector</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For classic designer dashboards only.) Indicates whether items in widget are displayed (<code>true</code>) or hidden (<code>false</code>). If hidden (<code>false</code>), dashboard viewers can click the widget to view and change items. Default is <code>true</code>.</p> <p> <b>Note:</b> Mobile devices display items in a list, regardless of this setting.</p>
exploreLink	<p><b>Type</b> Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code></li> <li>• <code>comparetable</code></li> <li>• <code>listselector</code></li> <li>• <code>number</code></li> <li>• <code>pillbox</code></li> <li>• <code>valuestable</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates whether the widget shows the explore icon that dashboard viewers can click to explore the widget as a lens (<code>true</code>) or not (<code>false</code>). This option only applies to widgets based on steps in compact form. You can't explore widgets that are built on SAQL form steps. Default is <code>true</code>.</p> <p> <b>Note:</b> Mobile devices display the icon, regardless of this setting.</p>

Property Name	Details
fit (for chart widgets)	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• chart (only when <code>visualizationType</code> is <code>scatter</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates whether the axis of a chart is in the center of the data (<code>true</code>) or at (0, 0) (<code>false</code>). Default is <code>false</code>.</p>
fit (for image widgets)	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• image</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates how to scale the image. Valid values are: <code>original</code> (default), <code>stretch</code>, <code>tile</code>, <code>fitwidth</code>, and <code>fitheight</code>.</p>
fontSize	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• link</li> <li>• number</li> <li>• text</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The font size of a number or of text. Defaults are:</p> <ul style="list-style-type: none"> <li>• number: 36</li> <li>• text: 26</li> </ul>
hideHeaderColumn	<p><b>Type</b> Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart</li> </ul>


Property Name	Details
	<ul style="list-style-type: none"> <li>• <code>valuable</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> No. Only editable via JSON.</p> <p><b>Description</b> Indicates whether the first column in a raw data table—which is simply a count of rows—is hidden (<code>true</code>) or not (<code>false</code>). Default is <code>false</code>.</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
image	<p><b>Type</b> Array</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>container</code></li> <li>• <code>image</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) Identifies the image using the following properties.</p> <p><b>name</b> Name of the image.</p> <p><b>namespace</b> Optional. Namespace of the image. Default is null.</p> <p><b>Example</b></p> <pre data-bbox="646 1283 1446 1419">"image": {   "name": "My_Corporate_Logo",   "namespace": "" }</pre>
imageUrl	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>box</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For classic designer dashboards only.) The document Id of the image file that you want to apply as the background. To ensure security, the image file must be uploaded to Salesforce as a document and the <b>Externally Available Image</b> option must be</p>


Property Name	Details
	<p>selected. If this option is not selected or the referenced document is not an image, the image doesn't show up in the widget. Default is null.</p>
<code>includeState</code>	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>link</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Apply selections in chart, list, toggle, range, and date widgets in the source dashboard as selections in the linked asset. For example, you select North America in a list widget based on the Region dataset field. Wave applies that same selection to each step in the linked dashboard that has faceting enabled and has a grouping based on the Region field.  Default is <code>false</code>.</p>
<code>instant</code>	<p><b>Type</b> Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• <code>dateselector</code></li> <li>• <code>listselector</code></li> <li>• <code>rangeselector</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates whether other faceted widgets immediately update (<code>true</code>) or not (<code>false</code>) when a dashboard viewer makes a selection in this widget.  When <code>false</code>, dashboard viewers must click <b>Update</b> for their changes to cascade to faceted widgets. When <code>true</code>, the <b>Update</b> button is hidden.  Defaults are:</p> <ul style="list-style-type: none"> <li>• <code>dateselector: false</code></li> <li>• <code>listselector: true</code></li> <li>• <code>rangeselector: false</code></li> </ul> <p> <b>Note:</b> For list, range, or date widgets that are expanded in the Wave dashboard designer, this widget property is always enabled—meaning that selections in this widget instantly update other widgets. While these widgets are expanded, you can't change this setting.</p>

Property Name	Details
	
<p>legend</p>	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• chart (only when visualizationType is hbar, vbar, stackhbar, stackvbar, pie, scatter, time, hdot, vdot, matrix, calheatmap, heatmap, parallelcoords, stackwaterfall, funnel, or choropleth)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates whether to display a legend (true), or not (false). Default is false for all chart types, except pivottable.</p> <p> <b>Note:</b> Mobile devices can only display legends for pie widgets.</p>
<p>legendHideHeader</p>	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• chart (only when visualizationType is hbar, vbar, stackhbar, stackvbar, pie, scatter, time, hdot, vdot, matrix, calheatmap, heatmap, stackwaterfall, combo, combo, or parallelcoords)</li> </ul>

Property Name	Details
	<p><b>Exposed in the Dashboard Designer's User Interface</b> No. Only editable via JSON.</p> <p><b>Description</b> Indicates whether the legend has a title (<code>true</code>) or not (<code>false</code>). The title is always the name of the dimension that the legend describes.  Default is <code>false</code> for all chart types except <code>pivottable</code>.</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
legendWidth	<p><b>Type</b> Integer</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code> (only when <code>visualizationType</code> is <code>hbar</code>, <code>vbar</code>, <code>stackhbar</code>, <code>stackvbar</code>, <code>pie</code>, <code>scatter</code>, <code>time</code>, <code>hdot</code>, <code>vdot</code>, <code>matrix</code>, <code>calheatmap</code>, <code>heatmap</code>, <code>stackwaterfall</code>, <code>combo</code>, or <code>parallelcoords</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> No. Only editable via JSON.</p> <p><b>Description</b> The width of the legend area in pixels.  Default is <code>145</code> for all chart types except <code>pivottable</code>.</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
maxColumnWidth	<p><b>Type</b> Integer</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code> (only when <code>visualizationType</code> is <code>comparisontable</code>, <code>pivottable</code>, <code>stackhbar</code>, <code>stackvbar</code>, <code>hbar</code>, <code>stackwaterfall</code>, or <code>vbar</code>)</li> <li>• <code>comparisontable</code></li> <li>• <code>valuestable</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> No. Only editable via JSON.</p> <p><b>Description</b> (For classic designer dashboards only.) The maximum display size (in pixels) of a dimension field on a web browser of a desktop or laptop.  Default is <code>200</code>, minimum value is <code>20</code>, and maximum value is <code>200</code>.</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices. This setting doesn't apply to compare table columns that show bars in a classic designer dashboard if you specify a value less than <code>100</code>.</p>





Property Name	Details
measureField	<p data-bbox="609 262 657 294"><b>Type</b></p> <p data-bbox="641 304 706 336">String</p> <p data-bbox="609 346 901 378"><b>Available for These Widgets</b></p> <ul data-bbox="641 388 868 514" style="list-style-type: none"> <li>• listselector</li> <li>• number</li> <li>• pillbox</li> </ul> <p data-bbox="609 525 1153 556"><b>Exposed in the Dashboard Designer’s User Interface</b></p> <p data-bbox="641 556 673 588">Yes</p> <p data-bbox="609 598 730 630"><b>Description</b></p> <p data-bbox="641 640 1096 672">The mathematical function performed on data.</p> <p data-bbox="641 682 1356 714">Specify the <code>measureField</code> in this format: <code>&lt;formula&gt;_&lt;field&gt;</code>.</p> <p data-bbox="641 724 1429 798"><code>&lt;formula&gt;</code> must match one of the formulas specified in the <code>measures</code> step property. Possible values for <code>&lt;formula&gt;</code> are:</p> <ul data-bbox="641 808 1445 1050" style="list-style-type: none"> <li>• <code>avg</code> — calculate the mathematical average (mean)</li> <li>• <code>max</code> — the maximum value</li> <li>• <code>min</code> — the minimum value</li> <li>• <code>sum</code> — add all the values</li> <li>• <code>unique</code> — count the number of unique values. For example, use to count the number of unique dimensions.</li> </ul> <p data-bbox="641 1060 1429 1134">The <code>&lt;field&gt;</code> paired with the <code>&lt;formula&gt;</code> must match the field name that is specified in <code>measures</code>.</p> <p data-bbox="641 1144 1112 1176">For example, if the <code>measures</code> step property is:</p> <pre data-bbox="649 1197 1437 1680"> "measures": [   [     "sum",     "Profit"   ],   [     "avg",     "Discount"   ],   [     "count",     "ModelNumber"   ] ] </pre> <p data-bbox="641 1690 1445 1795">Then <code>measureField</code> must be <code>sum_Profit</code>, <code>avg_Discount</code>, or <code>unique_ModelNumber</code>. The <code>measureField</code> can't be <code>avg_Profit</code> because <code>avg</code> and <code>Profit</code> aren't paired together in the <code>measures</code> step property.</p> <p data-bbox="641 1806 1396 1879"> <b>Note:</b> Unlike for <code>measures</code>, a count on a dimension in the user interface calculates the number of unique dimension values. As a result,</p>

Property Name	Details
	<p>measureField in the underlying JSON shows the unique formula, like <code>unique_&lt;dimension_field_name&gt;</code>.</p> <p>Default is null.</p>
minColumnWidth	<p><b>Type</b> Integer</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• chart (only when visualizationType is comparisontable, pivottable, stackhbar, stackvbar, hbar, stackwaterfall, or vbar)</li> <li>• comparisontable</li> <li>• valuestable</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> No. Only editable via JSON.</p> <p><b>Description</b> (For classic designer dashboards only.) The minimum display size of a dimension field in pixels. Default is 30.</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
miniBars	<p><b>Type</b> Integer</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• chart (only when visualizationType is stackhbar, stackvbar, hbar, or vbar)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For classic designer dashboards only.) The display size in pixels of bars in bar charts. Default is 0 (available only for bar charts and column charts).</p>
modeSwitchingAllowed	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• dateselector</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>

Property Name	Details
	<p><b>Description</b></p> <p>(For Wave designer dashboards only.) Indicates whether the dashboard viewer can switch between absolute and relative dates in the date widget.</p> <p>Default is <code>true</code>.</p>
<code>multiMetrics</code>	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li><code>chart</code> (only when <code>visualizationType</code> is <code>hbar</code> or <code>vbar</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Indicates whether two or more measures are displayed as adjacent bars under each grouping (<code>true</code>) or as individual, adjacent graphs (<code>false</code>).</p> <p>Default is <code>false</code> (available only for bar charts and column charts).</p>
<code>negativeColor</code>	<p><b>Type</b></p> <p>String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li><code>chart</code> (only when <code>visualizationType</code> is <code>waterfall</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>The color of the measure columns that have decreased in value in the chart.</p> <p>Specify the color in this format: <code>rgb (a, b, c, d)</code>.</p> <p>Using a number between zero and 255, <b>a</b> indicates how much red is in the color, <b>b</b> how much green, and <b>c</b> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color.</p> <p>Using a number between zero and one, <b>d</b> indicates the level of transparency. A value of 0 is invisible and 1 is opaque.</p> <p>For example, <code>rgb (0, 0, 0, 0.93)</code> sets the color to a nearly opaque black. <code>rgb (255, 0, 0, 0.14)</code> sets the color to a nearly invisible red.</p> <p>Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red.</p>
<code>normalize</code>	<p><b>Type</b></p> <p>Boolean</p>

Property Name	Details
	<p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code> (only when <code>visualizationType</code> is <code>stackhbar</code> or <code>stackvbar</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates whether charts are displayed using a logarithmic scale (<code>true</code>) or a linear scale (<code>false</code>).  Default is <code>false</code> (available only for <code>stackhbar</code> and <code>stackvbar</code>).</p>
numberColor	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>number</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) The font color of the number.  Specify the color in this format: <code>rgb ( a , b , c , d )</code>.  Using a number between zero and 255, <b>a</b> indicates how much red is in the color, <b>b</b> how much green, and <b>c</b> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color.  Using a number between zero and one, <b>d</b> indicates the level of transparency. A value of 0 is invisible and 1 is opaque.  For example, <code>rgb ( 0 , 0 , 0 , 0 . 93 )</code> sets the color to a nearly opaque black. <code>rgb ( 255 , 0 , 0 , 0 . 14 )</code> sets the color to a nearly invisible red.  Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red.  Default is <code>#000</code>.</p>
numberSize	<p><b>Type</b> Integer</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>number</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) The font size of the number. Default is 26.</p>

Property Name	Details
pivoted	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• table</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) Indicates whether the table is pivoted. A pivot table requires the underlying step to have at least one grouping. Wave pivots the table on the last defined grouping. Default is <code>false</code>.</p>
positiveColor	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart (only when <code>visualizationType</code> is <code>waterfall</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The color of the measure columns that have increased in value or remained the same in the chart.</p> <p>Specify the color in this format: <code>rgb (a, b, c, d)</code>.</p> <p>Using a number between zero and 255, <b>a</b> indicates how much red is in the color, <b>b</b> how much green, and <b>c</b> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color.</p> <p>Using a number between zero and one, <b>d</b> indicates the level of transparency. A value of 0 is invisible and 1 is opaque.</p> <p>For example, <code>rgb (0, 0, 0, 0.93)</code> sets the color to a nearly opaque black. <code>rgb (255, 0, 0, 0.14)</code> sets the color to a nearly invisible red.</p> <p>Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red.</p>
showValues	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• chart (only when <code>visualizationType</code> is <code>stackwaterfall</code> or <code>waterfall</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p>

Property Name	Details
	<p><b>Description</b></p> <p>Indicates whether to display the values of each measure column (<code>true</code>) or not (<code>false</code>).</p> <p>Default is <code>true</code>.</p>
<code>splitAxis</code>	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li><code>chart</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Indicates whether each dimension in a chart is measured on its own axis (<code>true</code>) or a shared axis (<code>false</code>).</p> <p>Only applicable when <code>multiMetrics</code> is <code>true</code>.</p> <p>Default is <code>false</code> (available only for bar charts and column charts).</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>sqrt</code>	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li><code>chart</code> (only when <code>visualizationType</code> is <code>parallelcoords</code>, <code>hdot</code>, <code>vdot</code>, <code>time</code>, <code>scatter</code>, <code>stackhbar</code>, <code>stackvbar</code>, <code>hbar</code>, <code>stackwaterfall</code>, or <code>vbar</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Indicates whether charts are displayed using a logarithmic scale (<code>true</code>) or a linear scale (<code>false</code>).</p> <p>Default is <code>false</code> (available only for bar charts, column charts, line charts, and time series).</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>startColor</code>	<p><b>Type</b></p> <p>String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li><code>chart</code> (only when <code>visualizationType</code> is <code>waterfall</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p>



Property Name	Details
	<p><b>Description</b></p> <p>The color of the first measure column in the chart.</p> <p>Specify the color in this format: <code>rgb ( <b>a</b>, <b>b</b>, <b>c</b>, <b>d</b> )</code>.</p> <p>Using a number between zero and 255, <b>a</b> indicates how much red is in the color, <b>b</b> how much green, and <b>c</b> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color.</p> <p>Using a number between zero and one, <b>d</b> indicates the level of transparency. A value of 0 is invisible and 1 is opaque.</p> <p>For example, <code>rgb ( 0, 0, 0, 0.93 )</code> sets the color to a nearly opaque black. <code>rgb ( 255, 0, 0, 0.14 )</code> sets the color to a nearly invisible red.</p> <p>Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red.</p>
step	<p><b>Type</b></p> <p>String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart</li> <li>• comparetable</li> <li>• dateselector</li> <li>• globalfilters</li> <li>• listselector</li> <li>• number</li> <li>• pillbox</li> <li>• rangeselector</li> <li>• valuestable</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>The name of the step that supplies data for the widget.</p> <p>Default is null.</p>
stretch	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• box</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p>


Property Name	Details
	<p><b>Description</b></p> <p>Indicates whether an image's width and height are set to the same values of the widget's width and height (<code>true</code>) or not (<code>false</code>).</p> <p>Default is <code>false</code>.</p>
<code>stretchImage</code>	<p><b>Type</b></p> <p>Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>container</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>Indicates whether an image's width and height are set to the same values of the widget's width and height (<code>true</code>) or not (<code>false</code>).</p> <p>Default is <code>false</code>.</p>
<code>text</code>	<p><b>Type</b></p> <p>String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>link</code></li> <li>• <code>text</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>The message rendered in a text widget. For example, if <code>text</code> is assigned the value <code>"Hello, world!"</code>, then "Hello, World!" appears in the text widget.</p> <p>Default is <code>null</code>.</p>
<code>textAlignment</code>	<p><b>Type</b></p> <p>String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>link</code></li> <li>• <code>number</code></li> <li>• <code>text</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b></p> <p>Yes</p> <p><b>Description</b></p> <p>The alignment of text. Possible values include <code>left</code>, <code>center</code>, and <code>right</code>. If no value is specified, text alignment defaults to center.</p> <p>Defaults are:</p>





Property Name	Details
	<ul style="list-style-type: none"> <li>• number: right</li> <li>• text: center</li> </ul>
textColor	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• link</li> <li>• number</li> <li>• text</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>The font color of text.</p> <p>Specify the color in this format: <code>rgb ( <i>a</i>, <i>b</i>, <i>c</i>, <i>d</i> )</code>.</p> <p>Using a number between zero and 255, <i>a</i> indicates how much red is in the color, <i>b</i> how much green, and <i>c</i> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color.</p> <p>Using a number between zero and one, <i>d</i> indicates the level of transparency. A value of 0 is invisible and 1 is opaque.</p> <p>For example, <code>rgb (0, 0, 0, 0.93)</code> sets the color to a nearly opaque black. <code>rgb (255, 0, 0, 0.14)</code> sets the color to a nearly invisible red.</p> <p>Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red.</p> <p>Default is <code>#000</code>.</p>
title	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• dateselector</li> <li>• listselector</li> <li>• number</li> <li>• pillbox</li> <li>• rangeselector</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b></p> <p>The title of a widget.</p> <p>Default is null.</p>

Property Name	Details
titleColor	<p><b>Type</b> String</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• number</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) The font color of the title. Specify the color in this format: <code>rgb (a, b, c, d)</code>. Using a number between zero and 255, <b>a</b> indicates how much red is in the color, <b>b</b> how much green, and <b>c</b> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color. Using a number between zero and one, <b>d</b> indicates the level of transparency. A value of 0 is invisible and 1 is opaque. For example, <code>rgb (0, 0, 0, 0.93)</code> sets the color to a nearly opaque black. <code>rgb (255, 0, 0, 0.14)</code> sets the color to a nearly invisible red. Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red. Default is <code>#000</code>.</p>
titleSize	<p><b>Type</b> Integer</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• number</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> (For Wave designer dashboards only.) The font size of the title. Default is 26.</p>
totalColor	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart (only when <code>visualizationType</code> is <code>waterfall</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The color of the total measure column in the chart. Specify the color in this format: <code>rgb (a, b, c, d)</code>.</p>

Property Name	Details
	<p>Using a number between zero and 255, <b>a</b> indicates how much red is in the color, <b>b</b> how much green, and <b>c</b> how much blue. A value of 0 indicates the absence of a color, and a value of 255 indicates the full expression of a color.</p> <p>Using a number between zero and one, <b>d</b> indicates the level of transparency. A value of 0 is invisible and 1 is opaque.</p> <p>For example, <code>rgb(0, 0, 0, 0.93)</code> sets the color to a nearly opaque black. <code>rgb(255, 0, 0, 0.14)</code> sets the color to a nearly invisible red.</p> <p>Alternatively, the color can be set using hexadecimal notation. When using hexadecimal notation, transparency can't be set. All hexadecimal colors default to opaque. <code>#000000</code> indicates black in hexadecimal. <code>#ff0000</code> indicates red.</p>
totals	<p><b>Type</b> Boolean</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code> (only when <code>visualizationType</code> is <code>pivottable</code>)</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> Indicates whether to include a row that displays the sum of all the values in each measure column (<code>true</code>) or not (<code>false</code>). Default for <code>chart</code> is <code>false</code> (available only for <code>pivottable</code>).</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
trellis	<p><b>Type</b> Boolean</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>chart</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> When a step has two or more groupings and one measure, indicates whether the last grouping displays on its own axis (<code>true</code>) or on the same axis as other groupings (<code>false</code>). Default for <code>chart</code> is <code>false</code> (available only for bar charts and column charts).</p> <p> <b>Note:</b> This setting doesn't apply when viewing the widget on mobile devices.</p>
videoSize	<p><b>Type</b> String</p>

Property Name	Details
	<p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• url</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The dimensions of a YouTube video. Possible values are:</p> <ul style="list-style-type: none"> <li>• (4/3) 240 x 180</li> <li>• (4/3) 420 x 315</li> <li>• (4/3) 480 x 360</li> <li>• (4/3) 640 x 480</li> <li>• (4/3) 960 x 720</li> <li>• (16/9) 320 x 180</li> <li>• (16/9) 560 x 315</li> <li>• (16/9) 640 x 360</li> <li>• (16/9) 853 x 480</li> <li>• (16/9) 1280 x 720</li> </ul> <p>Default is (4/3) 240 x 180.</p> <p> <b>Note:</b> Mobile devices don't display url widgets.</p>
visualizationType	<p><b>Type</b> String</p> <p><b>Available for These Widgets</b></p> <ul style="list-style-type: none"> <li>• chart</li> <li>• link</li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The type of chart used to show data. Possible values are:</p> <ul style="list-style-type: none"> <li>• calheatmap— calendar heat map</li> <li>• choropleth — choropleth (map)</li> <li>• combo — lines and bars to show multiple metrics</li> <li>• comparisontable — comparison table in the classic designer only</li> <li>• flatgauge — flat gauge in the Wave dashboard designer only</li> <li>• funnel — funnel</li> <li>• hbar — horizontal bar</li> <li>• hdot* — horizontal dot plot</li> <li>• heatmap— heat map</li> </ul>

Property Name	Details
	<ul style="list-style-type: none"> <li>• <code>matrix</code>— matrix</li> <li>• <code>parallelcoords*</code> — parallel coordinates</li> <li>• <code>pie</code> — donut</li> <li>• <code>pivottable*</code> — pivot table</li> <li>• <code>polargauge</code> — polar gauge in the Wave dashboard designer only</li> <li>• <code>pyramid</code> — pyramid in the Wave dashboard designer only</li> <li>• <code>rating</code> — rating in the Wave dashboard designer only</li> <li>• <code>scatter</code> — scatter plot</li> <li>• <code>stackhbar</code> — stacked horizontal bar</li> <li>• <code>stackpyramid</code> — stacked pyramid in the Wave dashboard designer only</li> <li>• <code>stackvbar</code> — stacked vertical bar</li> <li>• <code>stackwaterfall</code> — stacked waterfall</li> <li>• <code>time</code> — timeline</li> <li>• <code>valuestable</code> — raw data table in the classic designer only</li> <li>• <code>vbar</code> — vertical bar</li> <li>• <code>vdot*</code> — vertical dot plot</li> <li>• <code>waterfall</code> — waterfall</li> </ul> <p> <b>Note:</b> The Wave dashboard designer doesn't support chart types with an asterisk (*). If you specify an unsupported type, the designer replaces it with <code>hbar</code> in the dashboard.</p>
<code>url</code>	<p><b>Type</b> ConnectUri</p> <p><b>Available for This Widget</b></p> <ul style="list-style-type: none"> <li>• <code>url</code></li> </ul> <p><b>Exposed in the Dashboard Designer's User Interface</b> Yes</p> <p><b>Description</b> The URL of a YouTube video. Default is null.</p> <p> <b>Note:</b> Mobile devices don't display <code>url</code> widgets.</p>