
Wave Analytics Dashboard JSON Reference

Salesforce, Winter '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
- Dashboard JSON Properties 20
 - gridLayouts (for Wave Designer Dashboards Only) 21
 - widgetStyle Properties (for Wave Designer Dashboards Only) 33
 - layouts (for Classic Designer Dashboards Only) 35
 - steps 50
 - widgets 63

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 whether 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
    }
}

```

```

        },
        {
            "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": "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,
        "visualizationParameters": {
          "visualizationType": "hbar",
          "options": {}
        }
      }
    ],
    "CreatedDate_Year_CreatedDate_Month_9": {
      "datasets": [
        {
          "id": "0Fbx000000000LzCAI",
          "label": "Opportunities",
          "name": "opportunity1",
          "url": "/services/data/v38.0/wave/datasets/0Fbx000000000LzCAI"
        }
      ],
      "isFacet": true,
      "isGlobal": false,
      "query": {
        "measures": [
          [
            "sum",
            "Amount",
            "A",
            {
```

```

        "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",

```

```

        "CreatedDate_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",

```

```

        "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",

```

```

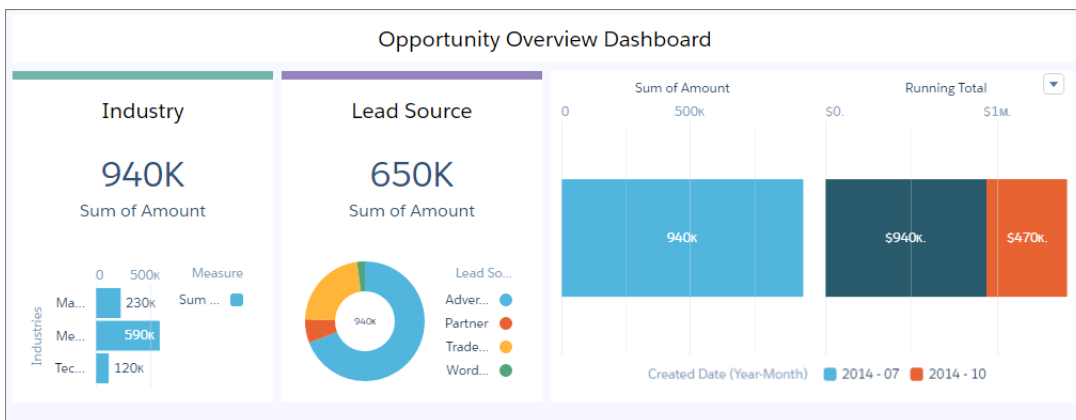
        "subtitleLabel": ""
    },
    "trellis": {
        "enable": false,
        "type": "x",
        "chartsPerLine": 4
    },
    "inner": 50
},
"type": "chart"
},
"chart_1": {
    "parameters": {
        "autoFitMode": "fit",
        "showValues": true,
        "barSize": 25,
        "legend": {
            "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": "0Fbx000000000LzCAI",
        "label": "Opportunities",
        "name": "opportunity1",
        "url": "/services/data/v38.0/wave/datasets/0Fbx000000000LzCAI"
      }
    ]
  }
}
```

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 0Fbx000000000LzCAI 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":
            "{\n\"groups\": [\n\"AccountId.Industry\"],\n\"measures\": [[\n\"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"
        }
      }
    }
  }
}
```

```

    "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
<code>label</code>	<p>Type String</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Name of the dashboard.</p>
<code>description</code>	<p>Type String</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Description of the dashboard.</p>
<code>state</code>	<p>Type Array</p> <p>Exposed in the Dashboard Designer's User Interface No</p> <p>Description 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>
<code>datasets</code>	<p>Type Array</p> <p>Exposed in the Dashboard Designer's User Interface No</p>

Property Name	Details
	Description Specifies all datasets used by steps in the dashboard.

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

[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 and clipped to the dashboard. The properties vary based on whether the step is in compact form or SAQL form.

[widgets](#)

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

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": 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
  }
}, {
  "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
}]
```

[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	Type String Exposed in the Dashboard Designer’s User Interface Yes. Description Name of the layout.
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	Type Integer Exposed in the Dashboard Designer’s User Interface Yes.

Property Name	Details
	Description The number of columns in the layout.
pages	Type Array Exposed in the Dashboard Designer's User Interface No Description Contains properties that determine the placement of each widget in the dashboard layout. Currently, Wave designer supports only one page for each layout.
selectors	Type Array Exposed in the Dashboard Designer's User Interface Yes. You can choose selectors when you configure the layout. Description A list of selector properties that determine when
style	Type Array Exposed in the Dashboard Designer's User Interface Yes. You can choose selectors when you configure the layout. Description A list of selector properties that determine when
version	

[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>Type Array</p> <p>Exposed in the Dashboard Designer's User Interface No</p> <p>Description 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
<code>name</code>	<p>Type String</p> <p>Exposed in the Dashboard Designer's User Interface No</p> <p>Description Internal name of the widget. This name is used to reference the widget in the dashboard JSON.</p>
<code>column</code>	<p>Type Integer</p> <p>Exposed in the Dashboard Designer's User Interface Yes. Value is determined based on the widget's placement.</p> <p>Description The column number where the widget starts. <code>column</code> and <code>row</code> specify the top left corner of the widget.</p> <p> Note: If this widget is included in a container, these properties are relative to the container widget.</p>
<code>row</code>	<p>Type Integer</p>

Property Name	Details
	<p>Exposed in the Dashboard Designer's User Interface Yes. Value is determined based on the widget's placement.</p> <p>Description The row number where the widget starts. <code>column</code> and <code>row</code> specify the top left corner of the widget.</p>
<code>colspan</code>	<p>Type Integer</p> <p>Exposed in the Dashboard Designer's User Interface Yes. Value is determined based on the widget's placement.</p> <p>Description 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>
<code>rowspan</code>	<p>Type Integer</p> <p>Exposed in the Dashboard Designer's User Interface Yes. Value is determined based on the widget's placement.</p> <p>Description 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>
<code>widgetStyle</code>	<p>Type Array</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • All widgets <p>Exposed in the Dashboard Designer's User Interface No</p> <p>Description 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
backgroundColor	<p>Type String</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Background color of the widget. The default is #FFFFFF.</p>
borderColor	<p>Type String</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Color of the widget's border. The default is #FFFFFF.</p>
borderEdges	<p>Type List</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>Type Integer</p> <p>Available for This Widget All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>

Property Name	Details
<code>borderWidth</code>	<p>Type Integer</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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(<width>)</code>	<p>Type Integer</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Minimum width (in pixels) of the devices supported by this layout.</p>
<code>maxWidth(<width>)</code>	<p>Type Integer</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Maximum width (in pixels) of the devices supported by this layout.</p>
<code>orientation(<orientation>)</code>	<p>Type String</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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(<platform>)</code>	<p>Type String</p>

Property Name	Details
	<p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</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>Type</p> <p>String</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</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>
<code>alignmentY</code>	<p>Type</p> <p>String</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>The vertical alignment of the background image applied to the dashboard.</p> <p>Valid values are: <code>top</code> (default), <code>center</code>, and <code>bottom</code>.</p>
<code>backgroundColor</code>	<p>Type</p> <p>String</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>Background color of the dashboard, specified in hex color code. The default is <code>#FFFFFF</code>.</p>
<code>cellSpacingX</code>	<p>Type</p> <p>Integer</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p>

Property Name	Details
	<p>Description</p> <p>Horizontal spacing (in pixels) between cells in the dashboard grid.</p> <p>Valid values are 0, 4, 8 (default), and 16.</p>
cellSpacingY	<p>Type</p> <p>Integer</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>Vertical spacing (in pixels) between cells in the dashboard grid.</p> <p>Valid values are 0, 4, 8 (default), and 16.</p>
documentId	<p>Type</p> <p>String</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>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 Externally Available Image option. The image doesn't show up if this option is not selected or the referenced document is not an image.</p>
fit	<p>Type</p> <p>String</p> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>Indicates how to scale the image.</p> <p>Valid values are: <code>original</code> (default), <code>stretch</code>, <code>tile</code>, <code>fitwidth</code>, and <code>fitheight</code>.</p>

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>Type String</p> <p>Available for This Widget All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Color of the widget's background, specified in hex color code. The default is #FFFFFF.</p>
borderColor	<p>Type String</p> <p>Available for This Widget All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>Type List</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>Type Integer</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>

Property Name	Details
<code>borderWidth</code>	<p>Type Integer</p> <p>Available for These Widgets All widgets</p> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Thickness of the border. 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.

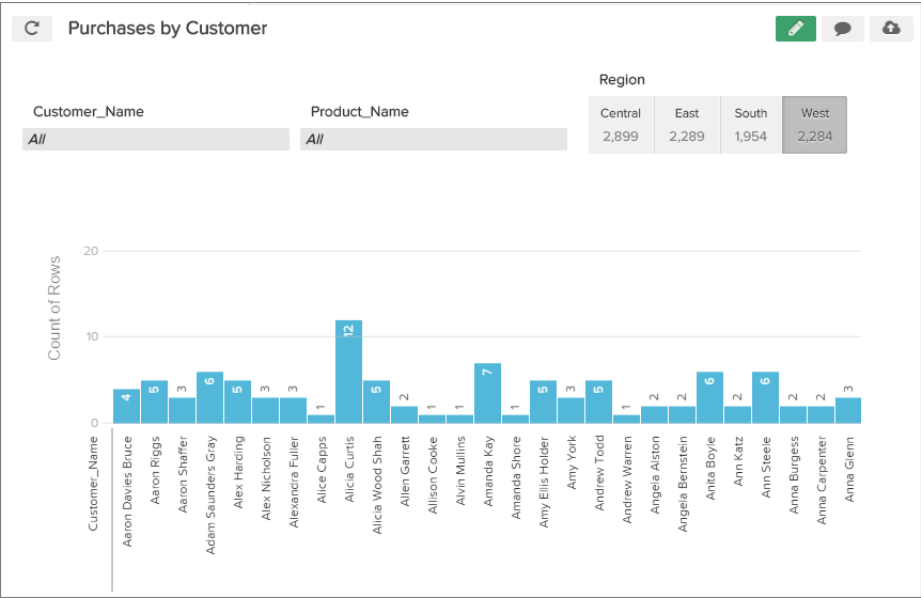
Mobile

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

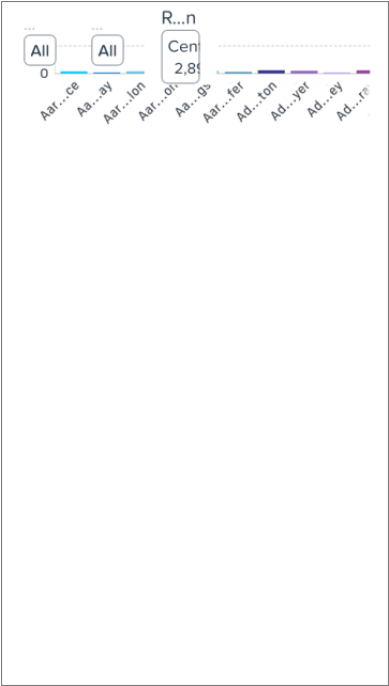
The mobile layout lets you optimize the position, order, and size of the widgets in your dashboard for display on mobile devices.

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.

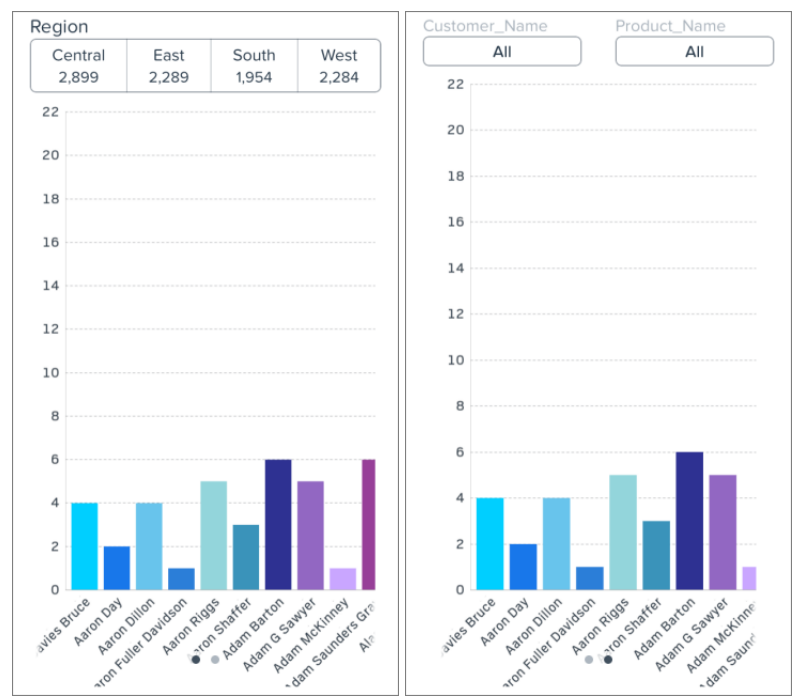
A dashboard with an absolute layout looks great in a Web browser:



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
  }
]

```

4. 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"
        ]
      }
    ]
  },
  {
    "version": 1
  }
]

```

```

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

```

5. 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 and 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 impracticably 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

Widget	Has a Fixed Width?	Has a Fixed Height?	Width Scaling Behavior	Height Scaling Behavior
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}"
        ]
      }
    ]
  },
  {
    "device": "ipad",
    "orientation": "landscape",
    "pages": [
      {
        "rows": [
          "dimfilter_1, box_1 {colspan=2, rowspan=3, zIndex=-1, vpad=5, hpad=5} | dimfilter_2"
        ]
      }
    ]
  }
]
"version": 1
}
```

```


|
|           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

Device	Maximum columns supported by autoformatting
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
<code>colspan</code>	<p>Type Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> All widgets <p>Description 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>Example In this example, the widget named <code>"chart_1"</code> spans 3 columns:</p> <pre>"layouts": [{ "device": "default", "pages": [{ "rows": [</pre>

Property Name	Details
	<pre> "dimfilter_1 dimfilter_2 dimfilter_3", "chart_1 {colspan=3}"] }] "version": 1 }</pre>
rowspan	<p>Type Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • All widgets <p>Description The number of rows that a widget spans—the height of the widget. If the dashboard doesn't have enough rows to accomodate the specified height, then rows are added.</p> <p>Example 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"] }] }] "version": 1 }</pre>
zIndex	<p>Type Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • All widgets <p>Description 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. The default value of <code>zIndex</code> is 0.</p>

Property Name	Details
	<p>Example</p> <p>In this example, the widget named “box_1” appears behind the widget named “number_1”:</p> <pre> "layouts": [{ "device": "default", "pages": [{ "rows": ["box_1 {zIndex=1}, number_1 {zIndex=2} chart_1"] }] }] "version": 1 </pre>
vpad	<p>Type</p> <p>Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> All widgets <p>Description</p> <p>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>Example</p> <p>In this example, the cell containing widget named “dimfilter_1” has 5 pixels of padding on its top and bottom sides:</p> <pre> "layouts": [{ "device": "default", "pages": [{ "rows": ["dimfilter_1 {vpad=5}"] }] }] "version": 1 </pre>
hpad	<p>Type</p> <p>Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> All widgets

Property Name	Details
	<p>Description</p> <p>The padding added to the left and right sides of the widget's cell in pixels. If <code>hpad</code> 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>The default value of <code>hpad</code> is 0.</p> <p>Example</p> <p>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", "pages": [{ "rows": ["dimfilter_1 {hpad=5}"] }] "version": 1 }]</pre>
<code>vAxisWidth</code>	<p>Type</p> <p>Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>chart</code> <p>Description</p> <p>The size of a chart widget's x-axis in pixels. Use <code>vAxisWidth</code> to align multiple chart widgets.</p> <p>Example</p> <p>In this example, the widget named <code>"chart_1"</code> has an x-axis that is 250 pixels wide:</p> <pre>"layouts": [{ "device": "default", "pages": [{ "rows": ["chart_1 {vAxisWidth=250}"] }] "version": 1 }]</pre>
<code>hAxisHeight</code>	<p>Type</p> <p>Integer</p>

Property Name	Details
	<p>Available for These Widgets</p> <ul style="list-style-type: none"> • chart <p>Description</p> <p>The size of a chart widget's y-axis in pixels. Use <code>hAxisHeight</code> to align multiple chart widgets.</p> <p>Example</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>Description</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 <i>preferred</i>, then the row's height is equal to the largest height</p> <p>Example</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"> compact placeholder
ChartWidget	<ul style="list-style-type: none"> legend miniBars fit normalize multiMetrics splitAxis backgroundColor measureAxis (true/false) categoryLabels (true/false) textColor limitBarThickness (true/false)
BoxWidget	<ul style="list-style-type: none"> backgroundColor borderColor imageUrl stretch
LinkWidget	<ul style="list-style-type: none"> destinationType destination visualizationType includeState
ListSelectorWidget	<ul style="list-style-type: none"> instant expanded
TextWidget	<ul style="list-style-type: none"> title or text textColor textAlignment fontSize
PillBoxWidget	<ul style="list-style-type: none"> backgroundColor textColor

Widget Type	Attributes
	<ul style="list-style-type: none"> • borderColor • selectedColor

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 and clipped to the dashboard. The properties vary based on whether the step is in compact form or SAQL form.

Example: Compact-Form Step

```
"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

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.

Also notice that the `'sum_Amount'` and `'sum_Amount'` projections in the SAQL query are referenced in measures as `[["count", "*", "sum_Amount"], ["count", "*", "sum_quantity"]]`. Measure projections in the SAQL always include the aggregation, underscore (`_`), and the name of the measure (`'sum_Amount'`) so that they can be referenced in the compact form `"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"
      ]
    ],
    "groups": [
      "Product",
      "StageName"
    ]
  },
  "isFacet": true,
  "useGlobal": true,
  "isGlobal": false,
  "datasets": [{
    "name": "Flexy_Sales",
    "url": "/services/data/v38.0/wave/datasets/0FbB00000000q5gKAA",
    "id": "0FbB00000000q5gKAA"
  }]
}

```



Example: aggregateflex Step with Bindings for Wave Dashboard Designer

```

"steps": {
  "Account_BillingCount_1": {
    "datasets": [{
      "id": "0FbB00000000oEkKAI",
      "label": "Opportunities",
      "name": "opportunity",
      "url": "/services/data/v38.0/wave/datasets/0FbB00000000oEkKAI"
    }],
    "isFacet": true,
    "isGlobal": false,
    "query": {
      "measures": "{{column(StaticMeasureNames.selection,
[\"value\"]).asObject()}}",
      "limit": "{{column(StaticLimits.selection, [\"value\"]).asObject()}}",
      "groups": "{{column(StaticGroupingNames.selection,

```

```
[\"value\\\")].asObject()}}\",
    \"filters\": \"{{column(StaticFilters.selection, [\"value\\\")].asObject()}}\",

    \"order\": \"{{column(StaticOrdering.selection, [\"value\\\")].asObject()}}\"
  },
  \"selectMode\": \"single\",
  \"type\": \"aggregateflex\",
  \"useGlobal\": true,
  \"visualizationParameters\": {
    \"visualizationType\": \"hbar\",
    \"options\": {}
  }
}
```



Example: aggregate Step with Bindings for Classic Designer

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 the 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

This example shows a compare table step for a classic designer dashboard for a mobile client. The `piqq1` 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	<div></div>	3.0679
Fin Svcs	<div></div>	3.1796
Mfg	<div></div>	2.8361
Healthcare	<div></div>	3.5238
Prof Svcs	<div></div>	3.4258
Consumer	<div></div>	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. A global SAQL definition is supported, 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*.


steps Properties

The `steps` key defines all steps available in the 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

The `steps` key defines all steps available in the 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>	An array of datasets used by this step. Specify the alias of each dataset. Currently, only the first dataset is used. If you don't specify the dataset even if it's specified in the <code>pigql</code> attribute, the dashboard doesn't render.

 **Note:** Faceted steps are filtered based on only the first dataset specified in this array.

Field Name	Description
visualizationParameters	<p>Visualization details about the step. Example:</p> <pre>"visualizationParameters": { "options": { "legend": false, "legendHideHeader": false, "legendWidth": 145, "maxColumnWidth": 200, "minColumnWidth": 30, "miniBars": 0, "multiMetrics": false, "splitAxis": false, "sqrt": false, "trellis": false }, "visualizationType": "hbar" }</pre>
isFacet	<p>Enables bi-directional faceting between this step and other steps built from the same dataset, which is specified in <code>datasets</code> field for this step. Set to <code>true</code> or <code>false</code>.</p> <p> Note: If a SAQL query is based on multiple datasets, only the first dataset specified in the <code>datasets</code> field is faceted. Also, <code>isFacet</code> works only for compact-form queries, by default. To enable them for a SAQL query, also set the <code>autoFilter</code> option to <code>true</code>.</p>
isGlobal	<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/global filter widget—all other steps ignore this property.</p> <p>A scope/global filter 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>
query	The query that the step uses. It can be in SAQL or compact form.
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 and toggle selectors are: <code>single</code>, <code>singlerequired</code>, <code>multi</code>, and <code>multirequired</code>.</p> <p> Note: <code>selectMode</code> isn't used by the number, values table, compare table, range, date, and global filter widgets.</p>
start	The initial filters that are applied when the dashboard first opens.
type	<p>The type can be set to:</p> <ul style="list-style-type: none"> <code>grain</code>—Use to build a values table. <code>multi</code>—Use to build a compare table in a classic designer dashboard. Use an <code>aggregate</code> or <code>aggregateflex</code> step to build a compare table in a Wave designer dashboard. <code>static</code>—Use to build a static step. Although Wave designer dashboards support <code>static</code> steps, Salesforce.com recommends using <code>staticflex</code> steps instead.

Field Name	Description
	<ul style="list-style-type: none"> <code>staticflex</code>—Use to build a static step in a Wave dashboard designer. <code>aggregate</code>—Use for all other steps. Although Wave designer dashboards support <code>aggregate</code> steps, Salesforce.com recommends using <code>aggregateflex</code> steps instead. <code>aggregateflex</code>—Use for all other steps in a Wave designer dashboards only. <p> Note: 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 Wave Analytics Bindings Developer Guide.</p>
<code>useGlobal</code>	Indicates whether the step uses the dashboard's scope/global filter widget (<code>true</code>) or not (<code>false</code>).
<code>dimensions</code>	<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>
<code>values</code>	<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>

Field Name	Description
	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 Wave Analytics Bindings Developer Guide .

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.


visualizationParameters Properties

The `visualizationParameters` key contains the definitions of steps that you created to support the dashboard widgets.


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>aggregateFilters</code>	Automatically generated. Don't modify.
<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.
<code>dimensions</code>	The dimensions to use are specified this way: <div data-bbox="378 1507 1243 1556" data-label="Text"> <pre>"dimensions": ["Department"]</pre> </div>
<code>facet_filters</code>	Automatically generated. Don't modify.
<code>filters</code>	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": <div data-bbox="378 1736 1243 1785" data-label="Text"> <pre>"filters": [{"dest", ["SFO", "LAX", "ORD", "DFW"]}]</pre> </div> <p> Note: 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>

Field Name	Description
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 <i>*</i>, <i>/</i>, <i>-</i>, <i>+</i>, <i>(</i>, and <i>)</i> 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": { "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, </pre>

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

```

        "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": []
      }
    }
  ]
}
},

```

groups

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 pigql parameter.

Field Name	Description
<code>limit</code>	<p>The number of results to return. For example, <code>"limit": 10</code>. The results that the limit statement returns aren't automatically ordered, so use this statement only with data that has been ordered.</p> <p> Note: 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>
<code>measures</code>	<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>pigql</code> parameter.</p>
<code>order</code>	<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: <code>"order": []</code>.</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> <p> Note: 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>pigql</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>pigql</code> attribute—Wave ignores the following attributes if they are set under the query attribute: filters, limit, and order. You must include each measure in the SAQL</p>


Field Name	Description
	<p>query and also specify it in the measures attribute. To specify a grouping, include a group by statement in the SAQL query and specify the same dimension in the groups attribute.</p> <p> Note: 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>
values	<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>

visualizationParameters Properties

The `visualizationParameters` key contains the definitions of steps that you created to support the dashboard widgets.

```
"visualizationParameters": {
  "options": {
    "legend": false,
    "legendHideHeader": false,
    "legendWidth": 145,
    "maxColumnWidth": 200,
    "minColumnWidth": 30,
    "miniBars": 0,
    "multiMetrics": false,
    "splitAxis": false,
    "sqrt": false,
    "trellis": false
  },
  "visualizationType": "hbar"
}
```

Field Name	Description
options	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 these options, see widget Properties .
visualizationType	Specifies the chart type. You can override the chart type at the widget level.

Field Name	Description
	Valid values for visualizationType are:
	<ul style="list-style-type: none"> calheatmap*—calendar heat map comparisontable—compare table hbar—horizontal bar hdot*—horizontal dot plot heatmap*—heat map matrix*—matrix parallelcoords*—parallel coordinates pie—donut pivottable*—pivot table scatter—scatter plot stackhbar—stacked horizontal bar stackvbar—stacked vertical bar time—timeline valuestable—raw data table vbar—vertical bar vdot*—vertical dot plot
	 Note: The Wave dashboard designer doesn't support charts with an asterisk (*). If you specify an unsupported type, the designer replaces it with a horizontal bar (hbar) in the dashboard.

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 parameters Properties .
<code>position</code>	<p>(For classic designer dashboards only.) Specifies the position of the widget in the dashboard. Position can consist of the following properties:</p> <p>x and y Specifies the top left corner of the widget. The values of these fields must be integers.</p> <p>w and h Specifies the width and height, respectively. You can enter "auto," percentages ("36%"), and integers ("20") as a string value.</p> <p>zIndex Determines 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. The value must be an integer.</p>

Field Name	Description
	<p>Example:</p> <pre>"position": { "x": 40, "y": 40, "w": "300", "h": "auto"}</pre> <p>Measurements are in pixels.</p> <p> Note: The Wave dashboard designer ignores these settings and uses the <code>position</code> attribute specified under the <code>gridLayouts</code> section of the dashboard JSON.</p>
<code>type</code>	<p>The widget type specifies one of the other supported widget types. The value of this field must be a string.</p> <ul style="list-style-type: none"> • <code>box</code>—available in the classic designer only • <code>chart</code> • <code>comparabletable</code> • <code>container</code>—available in the Wave dashboard designer only • <code>dateselector</code> • <code>globalfilters</code> • <code>image</code>—available in the Wave dashboard designer only • <code>link</code> • <code>listselector</code> • <code>number</code> • <code>pillbox</code> • <code>rangeselector</code> • <code>table</code>—available in the Wave dashboard designer only • <code>text</code> • <code>url</code>—available in the classic designer only • <code>valuetable</code> <p> Note: The Wave dashboard designer doesn't support <code>box</code> and <code>url</code> widgets. The designer removes these widget types when you open the dashboard. Also, the classic designer doesn't support the <code>container</code> widget—use a <code>box</code> widget instead.</p>

[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. For information about chart-specific properties in a Wave designer dashboard, see [Visualizing Data with Charts](#)—this section doesn't cover chart properties for Wave designer dashboards.

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



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

Property Name	Details
alignmentX	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • image


Property Name	Details
	<p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates the horizontal alignment of the image in the widget. Valid values are: <code>left</code> (default), <code>center</code>, and <code>right</code>.</p>
<code>alignmentY</code>	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>image</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates the vertical alignment of the image in the widget. Valid values are: <code>top</code> (default), <code>center</code>, and <code>bottom</code>.</p>
<code>compact</code>	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>listselector</code> • <code>number</code> • <code>pillbox</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates whether displayed numbers are abbreviated (<code>true</code>) or not (<code>false</code>). 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. Default is <code>false</code>.</p>
<code>computeTotal</code>	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>stackwaterfall</code> and <code>waterfall</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates whether to include the total measure column (<code>true</code>) or not (<code>false</code>).</p>


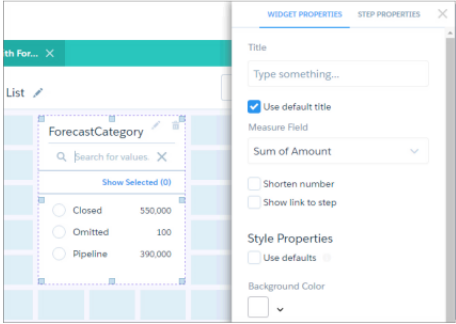
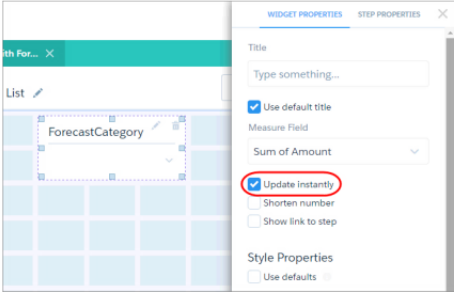
Property Name	Details
	Default is true.
containedWidgets	<p>Type List</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • container <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description A list of all widgets inside the container widget.</p> <p>Example This example shows 2 widgets (meafilter_1 and chart_1) included in the container widget (container_1).</p> <pre> "container_1": { "type": "container", "position": { "x": 0, "y": 0 }, "parameters": { "containedWidgets": ["meafilter_1", "chart_1"] } } </pre>
destination	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • link <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The ID of the dashboard, lens, or step. Default is null.</p>
destinationType	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • link



Property Name	Details
	<p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The destination type of a link. Possible values are:</p> <ul style="list-style-type: none"> • <code>dashboard</code> — a saved dashboard • <code>explore</code> — an unsaved, active exploration session of the lens • <code>lens</code> — a saved lens <p>Default is <code>lens</code>.</p>
<code>documentId</code>	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>image</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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 Externally Available Image 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>Example This example image widget (<code>image_1</code>) displays an image with ID <code>015R0000000DC1P</code>.</p> <pre> "image_1": { "type": "image", "parameters": { "documentId": "015R0000000DC1P", "fit": "stretch", "alignmentX": "center", "alignmentY": "center" } } </pre>
<code>dualAxis</code>	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>combo</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>



Property Name	Details
expanded	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>dateselector</code> • <code>listselector</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description (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> Note: Mobile devices display items in a list, regardless of this setting.</p>
exploreLink	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>chart</code> • <code>comparetable</code> • <code>listselector</code> • <code>number</code> • <code>pillbox</code> • <code>valuestable</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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 affects widgets based on steps in compact form, not SAQL form. Regardless of this setting, you can't explore widgets that are built on SAQL form steps. Default is <code>true</code>.</p> <p> Note: Mobile devices display the icon, regardless of this setting.</p>
fit (for chart widgets)	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>scatter</code>,



Property Name	Details
	<p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>
<code>fit</code> (for image widgets)	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>image</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>
<code>fontSize</code>	<p>Type Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>link</code> • <code>number</code> • <code>text</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The font size of a number or of text. Defaults are:</p> <ul style="list-style-type: none"> • <code>number</code>: 36 • <code>text</code>: 26
<code>hideHeaderColumn</code>	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>chart</code> • <code>valuestable</code> <p>Exposed in the Dashboard Designer's User Interface No. Only editable via JSON.</p>

Property Name	Details
	<p>Description</p> <p>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>).</p> <p>Default is <code>false</code>.</p> <p> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>imageUrl</code>	<p>Type</p> <p>String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>box</code> • <code>container</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>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 Externally Available Image 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>Example</p> <p>This example has a container widget (<code>container_1</code>) with a background image. The image has document Id <code>01599000000D8HP</code>.</p> <pre> "container_1": { "type": "container", "position": { "x": 0, "y": 0 }, "parameters": { "containedWidgets": [], "imageUrl": "01599000000D8HP" } } </pre>
<code>includeState</code>	<p>Type</p> <p>Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>link</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p>

Property Name	Details
	<div><div>Description</div><div>Apply selections in chart, list, toggle, range, and date widgets in the source dashboard as selections in the linked asset. For example, if you select North America in a list widget based on the Region dataset field, that same selection is applied 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>.</div></div>
<code>instant</code>	<div><div>Type</div><div>Boolean</div><div>Available for These Widgets</div><div><ul style="list-style-type: none"><code>dateselector</code><code>listselector</code><code>rangeselector</code></div><div>Exposed in the Dashboard Designer's User Interface</div><div>Yes</div><div><div>Description</div><div>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 Update for their changes to cascade to faceted widgets. When <code>true</code>, the Update button is hidden. Defaults are:<ul style="list-style-type: none"><code>dateselector: false</code><code>listselector: true</code><code>rangeselector: false</code><div><div> Note: 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.</div></div></div></div><div><div><div>Expanded List Widget</div><div></div></div><div><div>Collapsed List Widget</div><div></div></div></div></div>

Property Name	Details
legend	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> chart (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>parallelcoords</code>, <code>stackwaterfall</code>, <code>funnel</code>, or <code>choropleth</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates whether to display a legend (<code>true</code>), or not (<code>false</code>). Default is <code>false</code> for all chart types, except <code>pivottable</code>.</p> <p> Note: Mobile devices can only display legends for <code>pie</code> widgets.</p>
legendHideHeader	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> chart (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>, <code>combo</code>, or <code>parallelcoords</code>) <p>Exposed in the Dashboard Designer's User Interface No. Only editable via JSON.</p> <p>Description 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> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
legendWidth	<p>Type Integer</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> chart (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>) <p>Exposed in the Dashboard Designer's User Interface No. Only editable via JSON.</p> <p>Description The width of the legend area in pixels. Default is <code>145</code> for all chart types except <code>pivottable</code>.</p>



Property Name	Details
	 Note: This setting doesn't apply when viewing the widget on mobile devices.
maxColumnWidth	<p>Type Integer</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • chart (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>) • <code>comparisontable</code> • <code>valuestable</code> <p>Exposed in the Dashboard Designer's User Interface No. Only editable via JSON.</p> <p>Description The maximum display size (in pixels) of a dimension field on a web browser of a desktop or laptop. Default is 200, minimum value is 20, and maximum value is 200.</p> <p> Note: 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 100.</p>
measureField	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>listselector</code> • <code>number</code> • <code>pillbox</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The mathematical function performed on data. Specify the <code>measureField</code> in this format: <code><formula>_<field></code>. <code><formula></code> must match one of the formulas specified in the <code>measures</code> step property. Possible values for <code><formula></code> are:</p> <ul style="list-style-type: none"> • <code>avg</code> — calculate the mathematical average (mean) • <code>max</code> — the maximum value • <code>min</code> — the minimum value • <code>sum</code> — add all the values • <code>unique</code> — count the number of unique values. For example, use to count the number of unique dimensions. <p>The <code><field></code> paired with the <code><formula></code> must match the field name that is specified in <code>measures</code>.</p>

Property Name	Details
	<p>For example, if the <code>measures</code> step property is:</p> <pre>"measures": [["sum", "Profit"], ["avg", "Discount"], ["count", "ModelNumber"]]</pre> <p>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> Note: Unlike for measures, a count on a dimension in the user interface calculates the number of unique dimension values. As a result, <code>measureField</code> in the underlying JSON shows the unique formula, like <code>unique_<dimension_field_name></code>.</p> <p>Default is null.</p>
<code>minColumnWidth</code>	<p>Type Integer</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <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>) • <code>comparisontable</code> • <code>valuestable</code> <p>Exposed in the Dashboard Designer's User Interface No. Only editable via JSON.</p> <p>Description The minimum display size of a dimension field in pixels. Default is 30.</p> <p> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>miniBars</code>	<p>Type Integer</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>stackhbar</code>, <code>stackvbar</code>, <code>hbar</code>, or <code>vbar</code>)

Property Name	Details
	<p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description (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>
multiMetrics	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • chart (only when visualizationType is hbar or vbar) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates whether two or more measures are displayed as adjacent bars under each grouping (true) or as individual, adjacent graphs (false). Default is false (available only for bar charts and column charts).</p>
negativeColor	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • chart (only when visualizationType is waterfall) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The color of the measure columns that have decreased in value in the chart. Specify the color in this format: <code>rgb(a, b, c, d)</code>. Using a number between zero and 255, a indicates how much red is in the color, b how much green, and c 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, d 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. #000000 indicates black in hexadecimal. #ff0000 indicates red.</p>
normalize	<p>Type Boolean</p>

Property Name	Details
	<p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>stackhbar</code> or <code>stackvbar</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>
<code>numberColor</code>	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>number</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description (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, a indicates how much red is in the color, b how much green, and c 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, d 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>
<code>numberSize</code>	<p>Type Integer</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>number</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description (For Wave designer dashboards only.) The font size of the number. Default is 26.</p>

Property Name	Details
positiveColor	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>waterfall</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The color of the measure columns that have increased in value or remained the same in the chart. Specify the color in this format: <code>rgb (a, b, c, d)</code>. Using a number between zero and 255, a indicates how much red is in the color, b how much green, and c 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, d 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.</p>
showValues	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>chart</code> (only when <code>visualizationType</code> is <code>stackwaterfall</code> or <code>waterfall</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates whether to display the values of each measure column (<code>true</code>) or not (<code>false</code>). Default is <code>true</code>.</p>
splitAxis	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>chart</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description Indicates whether each dimension in a chart is measured on its own axis (<code>true</code>) or a shared axis (<code>false</code>). Only applicable when <code>multiMetrics</code> is <code>true</code>.</p>



Property Name	Details
	<p>Default is <code>false</code> (available only for bar charts and column charts).</p> <p> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>sqrt</code>	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> <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>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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 bar charts, column charts, line charts, and time series).</p> <p> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>startColor</code>	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> <code>chart</code> (only when <code>visualizationType</code> is <code>waterfall</code>) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The color of the first measure column in the chart. Specify the color in this format: <code>rgb(a, b, c, d)</code>. Using a number between zero and 255, a indicates how much red is in the color, b how much green, and c 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, d 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.</p>
<code>step</code>	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> <code>chart</code>


Property Name	Details
	<ul style="list-style-type: none"> • <code>comparabletable</code> • <code>dataselector</code> • <code>globalfilters</code> • <code>listselector</code> • <code>number</code> • <code>pillbox</code> • <code>rangeselector</code> • <code>valuablestable</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The name of the step that supplies data for the widget. Default is <code>null</code>.</p>
<code>stretch</code>	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>box</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>). Default is <code>false</code>.</p>
<code>stretchImage</code>	<p>Type Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>container</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description 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>). Default is <code>false</code>.</p>
<code>text</code>	<p>Type String</p>



Property Name	Details
	<p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>link</code> • <code>text</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>The message rendered in a text widget. For example, if <code>text</code> is assigned the value "Hello, world!", then "Hello, World!" appears in the text widget.</p> <p>Default is null.</p>
<code>textAlignment</code>	<p>Type</p> <p>String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>number</code> • <code>text</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</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> <ul style="list-style-type: none"> • <code>number: right</code> • <code>text: center</code>
<code>textColor</code>	<p>Type</p> <p>String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>link</code> • <code>number</code> • <code>text</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>The font color of text.</p> <p>Specify the color in this format: <code>rgb (a, b, c, d)</code>.</p> <p>Using a number between zero and 255, a indicates how much red is in the color, b how much green, and c 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>

Property Name	Details
	<p>Using a number between zero and one, d 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. #000000 indicates black in hexadecimal. #ff0000 indicates red.</p> <p>Default is #000.</p>
title	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • <code>dateselector</code> • <code>listselector</code> • <code>number</code> • <code>pillbox</code> • <code>rangeselector</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The title of a widget. Default is null.</p>
titleColor	<p>Type String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • <code>number</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description (For Wave designer dashboards only.) The font color of the title.</p> <p>Specify the color in this format: <code>rgb(a, b, c, d)</code>.</p> <p>Using a number between zero and 255, a indicates how much red is in the color, b how much green, and c 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, d 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>

Property Name	Details
	<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. #000000 indicates black in hexadecimal. #ff0000 indicates red.</p> <p>Default is #000.</p>
titleSize	<p>Type Integer</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> • number <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description (For Wave designer dashboards only.) The font size of the title. Default is 26.</p>
totalColor	<p>Type String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • chart (only when visualizationType is waterfall) <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The color of the total measure column 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, a indicates how much red is in the color, b how much green, and c 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, d 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. #000000 indicates black in hexadecimal. #ff0000 indicates red.</p>
totals	<p>Type Boolean</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • chart (only when visualizationType is pivottable) <p>Exposed in the Dashboard Designer's User Interface Yes</p>

Property Name	Details
	<p>Description</p> <p>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>).</p> <p>Default for <code>chart</code> is <code>false</code> (available only for <code>pivottable</code>).</p> <p> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>trellis</code>	<p>Type</p> <p>Boolean</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> <code>chart</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>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>).</p> <p>Default for <code>chart</code> is <code>false</code> (available only for bar charts and column charts).</p> <p> Note: This setting doesn't apply when viewing the widget on mobile devices.</p>
<code>videoSize</code>	<p>Type</p> <p>String</p> <p>Available for This Widget</p> <ul style="list-style-type: none"> <code>url</code> <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>The dimensions of a YouTube video. Possible values are:</p> <ul style="list-style-type: none"> <code>(4/3) 240 x 180</code> <code>(4/3) 420 x 315</code> <code>(4/3) 480 x 360</code> <code>(4/3) 640 x 480</code> <code>(4/3) 960 x 720</code> <code>(16/9) 320 x 180</code> <code>(16/9) 560 x 315</code> <code>(16/9) 640 x 360</code> <code>(16/9) 853 x 480</code> <code>(16/9) 1280 x 720</code> <p>Default is <code>(4/3) 240 x 180</code>.</p>

Property Name	Details
	 Note: Mobile devices don't display url widgets.
visualizationType	<p>Type</p> <p>String</p> <p>Available for These Widgets</p> <ul style="list-style-type: none"> • chart • link <p>Exposed in the Dashboard Designer's User Interface</p> <p>Yes</p> <p>Description</p> <p>The type of chart used to show data. Possible values are:</p> <ul style="list-style-type: none"> • calheatmap* — calendar heat map • choropleth — choropleth (map) • combo — lines and bars to show multiple metrics • comparisontable — comparison table in the classic designer only • flatgauge — flat gauge in the Wave dashboard designer only • funnel — funnel • hbar — horizontal bar • hdot* — horizontal dot plot • heatmap* — heat map • matrix* — matrix • parallelcoords* — parallel coordinates • pie — donut • pivottable* — pivot table • polargauge — polar gauge in the Wave dashboard designer only • pyramid — pyramid in the Wave dashboard designer only • rating — rating in the Wave dashboard designer only • scatter — scatter plot • stackhbar — stacked horizontal bar • stackpyramid — stacked pyramid in the Wave dashboard designer only • stackvbar — stacked vertical bar • stackwaterfall — stacked waterfall • time — timeline • valuestable — raw data table in the classic designer only • vbar — vertical bar • vdot* — vertical dot plot • waterfall — waterfall

Property Name	Details
	 Note: 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.
<code>url</code>	<p>Type ConnectUri</p> <p>Available for This Widget</p> <ul style="list-style-type: none">• <code>url</code> <p>Exposed in the Dashboard Designer's User Interface Yes</p> <p>Description The URL of a YouTube video. Default is null.</p> <p> Note: Mobile devices don't display <code>url</code> widgets.</p>