# Table of Contents

**Introduction**..........................................................................................................1  
Getting started........................................................................................................1

**Get Started with Visualizations**........................................................................3  
Visualization reference........................................................................................3  
Data structure requirements for visualizations..................................................4

**Events List**............................................................................................................7  
Using events lists..................................................................................................7

**Table Visualizations**........................................................................................10  
Table visualization overview.............................................................................10  
Generate a table....................................................................................................10  
Format table visualizations...............................................................................12  
Table column Simple XML................................................................................19

**Charts**..................................................................................................................29  
Chart overview......................................................................................................29  
Data for charts.....................................................................................................30  
Pie chart................................................................................................................31  
Column and bar charts.......................................................................................33  
Line and area charts............................................................................................37  
Scatter chart............................................................................................................42  
Bubble chart..........................................................................................................44  
Event annotations for charts..............................................................................45  
Chart display issues..............................................................................................49

**Single Value**.......................................................................................................54  
Overview...............................................................................................................54  
Generate a single value.........................................................................................54  
Customize a single value......................................................................................56

**Gauges**..................................................................................................................60  
Using gauges...........................................................................................................60

**Maps**....................................................................................................................63  
Mapping data.........................................................................................................63  
Generate a Choropleth map..................................................................................63  
Configure a Choropleth map...............................................................................68
# Table of Contents

**Maps**
- Cluster maps.............................................................................................................69

**Trellis Layout for Visualizations**.................................................................71
- Use trellis layout to split visualizations......................................................71

**Get Started with Dashboards**...................................................................80
- Dashboard overview..................................................................................80
- About the dashboard editor......................................................................82

**Build and Edit Dashboards in Splunk Web**.............................................84
- Create a dashboard...............................................................................84
- Working with dashboard panels.................................................................85
- Add panels to dashboards........................................................................87
- Edit dashboards.......................................................................................92
- Edit visualizations....................................................................................96
- Create and edit forms................................................................................99
- Convert a dashboard to HTML...............................................................115

**Create Dashboards with Simple XML**....................................................117
- Editing Simple XML.................................................................................117
- Searches power dashboards and forms................................................119
- Dashboards and forms..........................................................................134
- Dashboard examples................................................................................136
- Form examples........................................................................................148
- Using a third party XML editor..............................................................156

**Drilldown and Dashboard Interactivity**...................................................160
- Use drilldown for dashboard interactivity...........................................160
- Link to a search.......................................................................................165
- Link to a dashboard................................................................................170
- Link to a URL............................................................................................179
- Manage token values in the current dashboard.....................................184
- Token usage in dashboards....................................................................194
- Chart controls..........................................................................................219

**Share and Manage Dashboards**...............................................................225
- Generate dashboard PDFs.......................................................................225
- Configure dashboard permissions.......................................................232
Table of Contents

Share and Manage Dashboards
Clone and manage dashboards..............................................................234

Simple XML Reference....................................................................................236
Simple XML Reference...........................................................................236
Chart configuration reference..................................................................317
Event Handler Reference........................................................................343
Token reference......................................................................................371
Customize Simple XML...........................................................................374
Introduction

Getting started

Learn how to share insights with data visualizations and dashboards.

To view a PDF that offers you a short overview of the most common operations, definitions, and commands you will use when you create dashboards and visualizations, see the Splunk Dashboards Quick Reference Guide.

The visualization and dashboard workflow

You might need to generate a new visualization or edit an existing dashboard. Working with dashboards and visualizations includes one or more of the following tasks.

Select a visualization

- Select a visualization to show specific data insights.
- To find and compare visualization options, see the Visualization reference.

Generate and configure visualizations

- Write a search to generate a visualization. Make sure that the search returns results in the correct format for rendering the visualization. See Data structure requirements for visualizations for an overview about data formatting.
- Configure or update visualization appearance and behavior. Change color modes, add captions, or adjust other visualization elements. See the Visualization reference for an overview of options and links to details on each visualization type.
Build and edit dashboards

- Add visualizations to new or existing dashboards.
- Work with the editing user interface to adjust dashboard components.
- Convert a dashboard to a form by adding user inputs.
- To get started, see the Dashboard overview and Create dashboards.

Share and manage dashboards

- Export dashboards for sharing. To get started, see Generate dashboard PDFs.
- Manage permissions for viewing and editing dashboards. See Configure dashboard permissions for details.
- Clone a dashboard or display a dashboard on the app home page. To learn more, see Clone and manage dashboards.

Edit Simple XML

- Use Simple XML source code to customize dashboard content and behavior. See About editing Simple XML for an overview and the Simple XML reference for more details.
Get Started with Visualizations

Visualization reference

Compare options and select a visualization to show the data insights that you need.

To quickly view the most fundamental overview of common visualizations and their use cases, note that you can access the Splunk Dashboards Quick Reference guide by clicking the link in Getting started.

<table>
<thead>
<tr>
<th>Visualization</th>
<th>Usage</th>
<th>To learn more see</th>
</tr>
</thead>
</table>
| Events list   | Show the events that a search generates.  
|               | • Show events without additional processing.  
|               | • Show extracted fields and values directly in a dashboard.  
|               | • Users can click on event fields or timestamps to open a more specific search. | Using events lists |
| Table         | Compare and aggregate field values.  
|               | • Isolate one or more specific fields from search results.  
<p>|               | • Add formatting to highlight trends or patterns in specific fields. | Table visualization overview |
| Charts        | Visualize one or more dimensions in a data set. Use one of the following chart types depending on how many dimensions, or fields, you are visualizing. | Chart overview |</p>
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Single value</td>
<td>Show an aggregated metric in context.</td>
<td>Single value overview</td>
</tr>
<tr>
<td></td>
<td>• Track recent changes or trends in real time.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Use colors to add context dynamically.</td>
<td></td>
</tr>
<tr>
<td>Gauges</td>
<td>Show an aggregated metric against a range.</td>
<td>Using gauges</td>
</tr>
<tr>
<td></td>
<td>• Track a metric as it approaches a specific target.</td>
<td></td>
</tr>
<tr>
<td>Maps</td>
<td>Visualize data with geographic coordinates.</td>
<td>Mapping data</td>
</tr>
<tr>
<td></td>
<td>• Use a Choropleth map to show and compare regional trends or concentrations.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Use a marker map to plot geographic data.</td>
<td></td>
</tr>
<tr>
<td>Custom visualizations</td>
<td>Analyze and represent unique data sets.</td>
<td>See Custom visualizations for more details.</td>
</tr>
<tr>
<td></td>
<td>An admin must install custom visualization apps to make them available for Splunk users.</td>
<td></td>
</tr>
</tbody>
</table>

**Data structure requirements for visualizations**

Visualizations require search results in specific formats or data structures. Write queries to generate results in the correct format for the visualization that you are building.

This topic provides an overview of data structures for visualizations. To learn about requirements for a specific visualization and how to generate results in the correct format, see one of the following topics.
Events list
  Using events lists

Table visualizations
  Generate a table

Charts
  Pie chart
  Column and bar charts
  Line and area charts
  Scatter chart
  Bubble chart

Single value
  Generate a single value

Gauges
  Using gauges

Maps
  Mapping Data

For an overview of visualization options, see the Visualization Reference in this manual.

Data and formatting requirements

Depending on the visualization that you are creating, you can use specific search commands to generate results in the correct format. For example, many visualizations require a search using transforming commands, such as stats, chart, timechart, or geostats to render.

Charts visualize one or more data series, or related data points. Depending on the chart type or complexity, the number and ordering of data series can vary.

Single value and gauge visualizations represent a single numerical value.

Maps combine a query and other data components, including data with coordinates or place information, lookup definitions, and geographical markup files.
Using the statistics table

When creating a visualization, you can check the Statistics table after running a search to make sure that result fields are generated correctly. The number and order of Statistics table columns show you the data structure that a search generated.

Additional information

Review specific visualization topics to check data format requirements and query recommendations.

To learn more about search commands that can generate visualizations, see the following topics.

- Statistical and charting functions in the Search Reference
- About transforming commands in the Search Manual
Events List

Using events lists

Add an events list to a dashboard to give users access to the events, fields, and values generated by a search. An events list does not abstract or process search results like a chart or other visualization does.

Generate an events list

The content in an events list depends on the search that you run. There are no additional data format requirements.

Prerequisites
Review Configuration options.

Steps

1. From the Search page, run a search.
2. Select the Events tab to view the events list.
3. (Optional) Select Save As > Dashboard panel to add the events list to a dashboard.
4. (Optional) Use the Format menu or Simple XML to configure the events list.

Configuration options

Use the Format menu to configure one or more of the following events list components. You can also adjust these components and make additional configurations using Simple XML.

Display and format options

Use the following settings to adjust events list appearance.

- Choose an events display option.
  - List (default): Show timestamps for each event separately.
  - Raw: Show raw events.
  - Table: Display events as a table. This format is different from the Statistics table visualization.
Configure row numbers, wrapping, and maximum lines

**Drilldown**

Use the drilldown editor and/or Simple XML to enable and configure drilldown on an events list. See Use drilldown for dashboard interactivity for more details on enabling and configuring drilldown.

When configuring drilldown on an events list in Simple XML, you can specify one of the following drilldown settings to provide different segment selection options.

<table>
<thead>
<tr>
<th>Drilldown setting</th>
<th>Segmenting option enabled for users</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>Select a major segment or one or more contiguous minor segments.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The first example shows a minor segment selection. The second example shows a major segment selection.</td>
<td></td>
</tr>
<tr>
<td>Inner</td>
<td>Select a single minor segment.</td>
<td></td>
</tr>
<tr>
<td>Outer</td>
<td>Select a complete major segment.</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>Disables drilldown (default)</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Event segmentation processing for events with long single lines of text can cause browser performance issues.

For more details, see Types of event segmentation in the *Knowledge Manager Manual*.

**Use case scenario**

An admin uses an events list to give users access to recent notable system events. To generate the events list, the admin runs the following search.

```plaintext
error OR failed OR severe OR ( sourcetype=access_* ( 404 OR 500 OR 503 ) )
```

The admin adds the events list to a dashboard tracking system status. Dashboard users can click on event fields or a timestamp in the list to open a
search using the clicked content.

For example, clicking on the /opt/splunk/var/log/splunk/splunkd.log source value in an event opens the following search in a new window.

* source="/opt/splunk/var/log/splunk/splunkd.log"
Table Visualizations

Table visualization overview

Tables can help you compare and aggregate field values. Use a table to visualize patterns for one or more metrics across a data set. Start with a query to generate a table and use formatting to highlight values, add context, or create focus for the visualization.

Create a table visualization

Learn how to generate and configure a table visualization. See the following topics for details.

- Generate a table
- Format table visualizations
- Table column Simple XML

Generate a table

To generate a table, write a search that includes a transforming command. From the Search page, run the search and select the Statistics tab to view and format the table.

You can use the table command in a search to specify the fields that the table includes or to change table column order.
**Search examples**

- **Transforming search**
  This search uses the `chart` transforming command.

  ```
  index = _internal | chart avg(bytes) over sourcetype
  ```

  The search generates a table with two columns.

- **Transforming search with the table command**
  This search generates a table with `action`, `host`, and `count` columns.

  ```
  index = _internal | stats count by action, host
  ```

  To change the columns that appear in the table or to change column order, add the `table` command to this search. For example, add `| table host count` to generate a table with only the `host` and `count` columns.

  ```
  index = _internal | stats count by action, host | table host count
  ```

**Table sparklines**

Sparklines show data patterns or trends in a results set. To generate a table sparkline, use `stats` or `chart` with the `sparkline` function in a search.

Sparkline width is determined by default data binning. You can adjust data binning as a parameter of the `sparkline` command.
For more information, see Add Sparklines to your search results in the Search Manual.

Format table visualizations

Use the Format menu to configure a table visualization.

Add summary statistics

Use the Format menu Summary tab to include column totals and percentages. For each statistic, a highlighted summary row appears at the bottom of the table. Column totals and/or percentages appear at the bottom of each column that contains numeric values.

Note: Values in a summary row reflect statistics for the complete search result set. For tables with more than one page of results, summary row values do not apply only to the currently displayed page.

Summary and data row differences

There are some behavior and formatting differences between summary rows and data rows in a table.

<table>
<thead>
<tr>
<th>Behavior or format</th>
<th>Summary rows</th>
<th>Data rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static highlight color</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Values in the row can skew table color formatting or data overlay</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Column number formatting applied to the row</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Drilldown available for the row</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Included in PDF or CSV export</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Totals data row behavior

A static summary row fits most use cases. If you generate a totals data row using the addcoltotals SPL command in a search, note the following table behavior impacts.

- An addcoltotals row is treated as a data row in the table.
• Because they are handled as data rows, addcoltotals rows are included in a PDF or CSV dashboard export.
• Color scales or data overlay can be skewed if a table includes an addcoltotals data row.
• Tables should not include an addcoltotals data row and a column totals summary row. If you opt to include a totals summary row, adjust the search to remove the addcoltotals command.

Summary row examples

The following examples show use case scenarios for adding column totals and percentage rows to a table.

*Totals summary row*

An analyst for an online retailer is evaluating how customer actions, such as purchases or quantity changes, relate to product types. The analyst is also comparing the relative frequency of different customer actions.

The following query generates a table showing product type counts for each customer action.

```plaintext
... | chart count(itemId) over categoryId by action
```

Using the Format menu, the analyst adds a totals summary row to the table.

The totals row shows relative totals for each customer action. For instance, there were 2634 purchase events in the results set, compared to 276 product removal events.
**Percentage summary row**

An analyst creates a table showing purchasing activity on a retail website. The following query generates results comparing purchases for different product types.

```plaintext
... | chart count(itemId) over action by categoryId
```

The analyst uses the **Format** menu to include a percentage row in the table.

This row shows a percentage for each product type relative to all purchases. For example, arcade games make up 9.5 percent of all purchases.

**Format table columns**

You can format individual table columns to add context or focus to the visualization. Click on the paintbrush icon at the top of each column to customize color and number formatting.

**Note:** Column formatting is not available for columns representing the `_time` field or for sparkline columns.

**Column color**

Select and configure one of the following color modes for the column.
Note: Column color formatting overrides existing heat map or high/low value data overlay settings.

Scale

Use a sequential or divergent color scale on column cells. You can choose a preset scale or a custom configuration to manage how colors in the scale are applied to column cells.

Depending on search results and data distribution, column color gradation can vary. Columns with relatively similar values will show the most color gradation. Outlying values can limit the gradation.

Color scale options

<table>
<thead>
<tr>
<th>Scale type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>Use a sequential scale to show how results approach a high value in the column.</td>
<td>This example column has sequential coloring. It is also sorted to show the highest values at the top.</td>
</tr>
<tr>
<td>Divergent</td>
<td>A divergent scale can show how results approach high and low values.</td>
<td>This example column has divergent coloring. It shows the lowest values at the top and the highest values at the bottom.</td>
</tr>
</tbody>
</table>

Configure a custom color scale

You can configure custom color handling by indicating minimum, midpoint, and maximum value colors. Use one of the following options to configure the minimum, midpoint, and maximum value interpretation for the color scale.
## Configuration options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Use case example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest and lowest values</td>
<td>This option highlights the highest and lowest values in the column.</td>
<td>• Show which products had the most purchases in a sales data set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Show how recent customer satisfaction survey results trended towards highest and lowest scores.</td>
</tr>
<tr>
<td>Number</td>
<td>Indicate numeric value thresholds. Cell color is determined according to how values align with the three thresholds.</td>
<td>• Show department course enrollment according to small, medium, and large roster size.</td>
</tr>
<tr>
<td>Percent</td>
<td>Determine cell color using percentages of the results value range.</td>
<td>• Show student test scores on a final exam.</td>
</tr>
<tr>
<td>Percentile</td>
<td>Determine cell color using percentiles of the results value distribution.</td>
<td>• Compare customer satisfaction survey results.</td>
</tr>
</tbody>
</table>

### Ranges

Apply color to cells in this column according to value ranges.

Use ranges to compare cell values categorically. For example, use red, yellow and green range colors to indicate low, medium, and high sales results.
Range configuration options:

- Adjust the default range value and color settings.
- Add or remove ranges.

Values

Apply colors according to cell values.

Use automatic value coloring or define custom rules. Automatic coloring applies a color to every cell in the column. Cells with the same value appear in the same color.

Custom rules can help highlight specific values that you are monitoring. For example, use custom rules to highlight three new products in recent sales data.

**Number format**

Enable and adjust number formatting for each column. The number format settings panel includes the following options.

- Enable or disable number formatting.
- Set decimal precision.
- Opt to use thousand separators.
- Specify a measurement unit to add context to the values in this column. You can position the unit before or after each value.
Configure table properties

After generating a table, use the Format menu to configure one or more of the following table components.

- The number of rows shown in each table page
- Wrapping
- Table row number display

Data overlay

The Format menu also includes the following data overlay options.

Heat map
Add different shades of a particular color to the table to show value variation over table rows.

High and low value
Add high and low value colors to the table to highlight the highest and lowest values.

Use data overlay only if you are not adding column color formatting to the table. Column color formatting overrides data overlay configurations.

Drilldown

By default, drilldown is disabled when you save visualizations to a dashboard. You can use the drilldown editor or Simple XML to enable and configure drilldown options. For example, use drilldown to link to Splunk Answers posts relevant to the value users click in a table cell. See Use drilldown for dashboard interactivity for more details on enabling and configuring drilldown.

Simple XML drilldown options

In Simple XML, you can set the drilldown option to one of the following values. Use the <drilldown> element to change the drilldown behavior.

<table>
<thead>
<tr>
<th>Option</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>By default, opens a secondary search using the field and value in the selected cell.</td>
</tr>
<tr>
<td>Row</td>
<td></td>
</tr>
</tbody>
</table>
By default, opens a secondary search using the field and values from cells in the selected row.

| None   | Disables drilldown. |

**Table column Simple XML**

Use format rules to configure table columns in Simple XML.

Indicate color scale and color palette rules to manage column color formatting. You can also use a number format rule to manage the appearance of numeric cell values.

Put all table formatting rules inside the `<table>` dashboard element.

```
<table>
    [...]  
</table>
```

**Format rule syntax**

To create a new format rule, indicate a format rule type and a column where you want to apply the rule. Use the following syntax.

```
<format type= [ "color" | "number" ]  field="<column_name>">
    [...]  
</format>
```

If you do not specify a field, the format rule is applied to the entire table.

**Color format rules**

To add column color, create a format rule with type "color" and the column name where you want to apply the rule.

Start configuring column color by specifying a color scale type. The color scale type indicates how color is applied to values in the cell. After defining a color scale, you can add a color palette to indicate which colors to use for the column.
Use the following syntax to specify a color format rule.

```xml
<format type="color" field="<column_name>">
    <scale type="<color_scale_type>" [color scale option configurations] </scale>
    <colorPalette type="<color_palette_type>" [color palette option configurations] </colorPalette>
</format>
```

### Color scale types and options

**category**
Apply colors to the column based on category. You can provide an optional category list to pre-populate the color scale. Additional categories that occur in results are added after the specified categories.

<table>
<thead>
<tr>
<th>Options and accepted values</th>
<th>Example</th>
</tr>
</thead>
</table>
| (Optional) list one or more category strings. | `<format type="color" field="server_status">
    <scale type="category">online, offline</scale>
</format>` |

**linear**
Map numeric data on a linear scale.

<table>
<thead>
<tr>
<th>Options and accepted values</th>
<th>Example</th>
</tr>
</thead>
</table>
| None | `<format type="color" field="purchases">
    <scale type="linear"></scale>
</format>` |

**log**
Map numeric data on a logarithmic scale.

<table>
<thead>
<tr>
<th>Options and accepted values</th>
<th>Example</th>
</tr>
</thead>
</table>
| None | `<format type="color" field="purchases">
    <scale type="log"></scale>
</format>` |
**minMidMax**
Map numeric data according to a range with a minimum, midpoint, and maximum value.

Indicate a type and a value for each of the range segments.

<table>
<thead>
<tr>
<th>minType, midType, maxType options</th>
<th>minValue, midValue, maxValue options for this type</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any valid floating point number.</td>
</tr>
<tr>
<td>Interpret values as discrete numbers.</td>
<td></td>
</tr>
<tr>
<td>percent</td>
<td>Any number between 0 and 100.</td>
</tr>
<tr>
<td>Interpret values as a percentage of the value range of the data.</td>
<td></td>
</tr>
<tr>
<td>percentile</td>
<td>Any number between 0 and 100.</td>
</tr>
<tr>
<td>Interpret values as a percentage of the distribution of the data.</td>
<td></td>
</tr>
</tbody>
</table>

**Type and value defaults**

All segment types default to `number`.

All percent and percentile values default as follows.

- minValue: use the lowest value from the data.
- midValue: use a value halfway between the lowest and highest value in the range.
- maxValue: use the highest value in the data.

**Example**

<format type="color" field="field">
sharedCategory
Use this scale type with the sharedList palette to apply automatic formatting to this column.

<table>
<thead>
<tr>
<th>Options and accepted values</th>
<th>Example</th>
</tr>
</thead>
</table>
| None. Use this scale with the sharedList palette as shown in the example. | <format type="color" field="sourcetype">
  <scale type="sharedCategory">
  </scale>
  <colorPalette type="sharedList">
  </colorPalette>
 </format> |

threshold
Specify a set of finite value thresholds for binning data.

<table>
<thead>
<tr>
<th>Options and accepted values</th>
<th>Example</th>
</tr>
</thead>
</table>
| List values in ascending order. You can use any finite numbers, including floating point values. All values less than the first threshold go into the first bin. All values equal to or greater than the last threshold go into the last bin. | <format type="color" field="purchase_count">
  <scale type="threshold">0,30,70,100</scale>
 </format> |

Color palette types and options
Once you define a color format rule and add a color scale to it, include a color palette type and options. The color palette determines which colors the scale applies to column cells.

expression
Use a logical expression that returns a color for a particular value.
Color string formats
Use any of the following formats.

- #FFF
- #FFFFFFFF
- 0xFFF
- 0xFFFFFFFF
- rgb(255, 255, 255)
- rgba (255, 255, 255, 1)

Example
This example expression applies the color #65A637 to cells with the value splunkd. For cells with other values, the color #0000CC is used.

```xml
<colorPalette type="expression">if (value == "splunkd", ", "#65A637", ", "#0000CC")
</colorPalette>
```

list
Specify a list of color strings for this palette.

Interpolate listed colors
Add the interpolate boolean value to indicate whether to interpolate colors adjacent to the ones in the list. Setting interpolate to "true" creates a smoother color gradient.
interpolate defaults to false.

Color string formats
Use any of the following formats.

- #FFF
- #FFFFFFFF
- 0xFFF
- 0xFFFFFFFF
- rgb(255, 255, 255)
- rgba (255, 255, 255, 1)

Example
<colorPalette type="list" interpolate="true">[#65A637,#6DB7C6,#F7BC38,#F58F39,#D93F3C]
</colorPalette>

**map**
Specify a map of one or more cell value and color string pairs.

Use the following map format.

```
{ {<cell_value_string>} : {<color>}, {<cell_value_string>} : {<color>}}
```

**Color string formats**
Use any of the following formats.

- #FFF
- #FFFFFF
- 0xFFF
- 0xFFFFFF
- rgb(255, 255, 255)
- rgba (255, 255, 255, 1)

**Example**

```
<colorPalette type="map">{"online":#65A637, "offline":#6A5C9E}
</colorPalette>
```

**minMidMax**
Specify minimum and maximum or minimum, mid, and maximum colors to use in generating a color gradient. Gradient values are interpolated between the specified colors.

Indicate colors for the following options.

- minColor *(Required)*
- midColor *(Optional)*
- maxColor *(Required)*
Color string formats
Use any of the following formats.

- #FFF
- #FFFFFFFF
- 0xFFF
- 0xFFFFFFFF
- rgb(255, 255, 255)
- rgba (255, 255, 255, 1)

Example

```xml
<colorPalette type="minMidMax" minColor="#FFFFFF" maxColor="#65A637">
</colorPalette>
```

sharedList
Use this palette with the sharedCategory color scale to apply automatic formatting to this column.

Example

```xml
<format type="color" field="sourcetype">
  <scale type="sharedCategory"></scale>
  <colorPalette type="sharedList"></colorPalette>
</format>
```

Number format rules
Specify how numeric values appear.

Use this syntax to create a number format rule.

```xml
<format type="number" field="count">
  <option name="<number_format_option_name>">[number_format_option_value]</option>
</format>
```
**Number format options**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Accepted values and defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>Specify the number of decimal precision places.</td>
<td>Use a number between 0-20.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to 2.</td>
</tr>
<tr>
<td>useThousandSeparators</td>
<td>Indicate whether to insert a comma or other symbol between every three digits. Symbols are set according to the language and region for the Splunk platform instance.</td>
<td>Boolean. Defaults to true.</td>
</tr>
<tr>
<td>unit</td>
<td>Indicate a unit label to place before or after the value.</td>
<td>Use any String. For best results, use an abbreviation or other brief label text.</td>
</tr>
<tr>
<td>unitPosition</td>
<td>Indicate where to place the unit label.</td>
<td>[before</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defaults to after.</td>
</tr>
</tbody>
</table>

**Number format example**

```xml
<table>
  <search>
    <query>index=_internal | head 10000 | stats count by sourcetype</query>
  </search>
  <format type="number" field="count">
    <option name="precision">3</option>
    <option name="useThousandSeparators">false</option>
    <option name="unit">MB</option>
    <option name="unitPosition">before</option>
  </format>
</table>
```

**Table format source code example**

This example table visualizes recent sales performance.
Columns represent product categories and id codes, as well as item sales totals. Format rules help distinguish categories, highlight particular items, and show relative sales metric density across all products.

The source code includes color scale, palette, and number format rules.

```xml
<dashboard>
  <label>Sales performance</label>
  <row>
    <panel>Sales this month</panel>
    <table>
      <title>Accessories and arcade game sales</title>
      <search>
        <query>source="tutorialdata (1).zip:*" | stats count by categoryId, itemId | table categoryId itemId count</query>
        <earliest>0</earliest>
        <sampleRatio>1</sampleRatio>
      </search>
      <option name="count">20</option>
      <option name="dataOverlayMode">none</option>
      <option name="drilldown">cell</option>
      <option name="rowNumbers">false</option>
      <option name="wrap">true</option>
      <format type="color" field="itemId">
        <colorPalette type="map">{"EST-15":#D93F3C,"EST-7":#6DB7C6}</colorPalette>
      </format>
      <format type="color" field="categoryId">
        <colorPalette type="map">{"ACCESSORIES":#6DB7C6,"ARCADE":#F7BC38,"STRATEGY":#AFE3EE}</colorPalette>
      </format>
    </table>
  </row>
</dashboard>
```
Charts

Chart overview

Select a chart type to show one or more data dimensions in a results set. Learn how charts visualize data series.

For a quick glance at common charts and common chart use case commands, you can view the Splunk Dashboards Quick Reference guide by clicking the link in Getting started.

Select a chart

You can select a chart depending on the number of data dimensions that you want to visualize. For example, use a pie chart to show how values combine in a single field. A bubble chart can show relationships between multiple fields in a data set.

<table>
<thead>
<tr>
<th>Chart type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pie</td>
<td>Shows a single dimension. Pie slice size represents the density or frequency of values in a field.</td>
</tr>
<tr>
<td>Column and bar</td>
<td>Represent one or more dimensions in a results set. These charts plot data on two axes. Each axis represents a results field. Column and bar charts can compare values and fields.</td>
</tr>
<tr>
<td>Line and area</td>
<td>Line charts can show value changes over time.</td>
</tr>
<tr>
<td>300px</td>
<td>Area charts show changes in an aggregated value over time.</td>
</tr>
</tbody>
</table>
**Scatter and bubble**

Represent multiple dimensions in a results set. These charts plot data on two axes. Data point appearance, size, and/or distribution show additional patterns or relationships.

### Get started

The following topics show you how to build and configure charts.

- Data for charts
- Pie chart
- Column and bar charts
- Line and area charts
- Scatter chart
- Bubble chart

### Data for charts

To build any chart, start with a transforming search that generates one or more data series.

A series is a sequence of related data points. These points can be plotted on a chart. For example, each line in a line chart shows one series.

When you run a transforming search, select the **Statistics** tab. Review the statistics table to see the series generated. After the first column, each additional column represents a series. A single series search generates two columns. A multiple series search generates three or more columns.

Different chart types are optimized to visualize one or more data series.

<table>
<thead>
<tr>
<th>Chart name</th>
<th>Optimized for single series?</th>
<th>Optimized for multiple series?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pie</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Pie charts can only render a single series.

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bar</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Line</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Area</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Scatter</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Bubble</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Use a pie chart to show how different field values combine over an entire data set. Each slice of a pie chart represents the relative importance or volume of a particular category.

**Data formatting**

Pie charts represent a single data series.

Use a transforming command in a search to generate the single series.

For example, count events in each source field category.

```plaintext
...| stats count by source
```

Check the **Statistics** table after running the search to make sure that a single series generated. The table should have two columns.

The example search generates the following table.
The first table column contains labels for each pie slice. The second column contains the numerical values that correspond to each label. The numerical values determine the relative size of each slice.

If the search generates a table with more than two columns, the extra columns are ignored.

**Configuration options**

You can use the **Format** menu to configure the following pie chart components.

**Drilldown**

Drilldown in a pie chart lets users click on a pie slice to open a secondary search using the clicked values. You can enable or disable drilldown in the Dashboard editor. See Use drilldown for dashboard interactivity for more details.

**Minimum size**

Set a minimum percentage size to apply when there are more than 10 slices. Data values below the minimum percentage are combined into an **other** slice.

**Create a pie chart**

**Prerequisites**

Review the following details about building pie charts.

- Data formatting
- Configuration options

**Steps**

1. Write a search that uses a transforming command to aggregate values in a field.
2. Run the search.
3. Select the **Statistics** tab below the search bar. The statistics table here should have two columns.
4. Select the **Visualization** tab and use the **Visualization Picker** to select the pie chart visualization.
5. (Optional) Use the **Format** menu to configure the visualization.
Examples

This search portion aggregates events by Code field values.

... | stats count by Code
The search generates a single data series representing values in the Code field.

The chart is configured with a 5% minimum size. Field values that represent less than 5% of the total data set are combined into an other slice.

This search uses the bytes and source fields to generate a single series.

index = _internal | chart avg(bytes) over source

Here, the source column provides pie slice labels. The avg(bytes) column provides the relative size of each slice, as percentages of the sum of avg(bytes) returned by the search.

Column and bar charts

Use column and bar charts to compare field values across a data set.

Data formatting

Column and bar charts represent one or more data series. To make sure that a search generates one or more series, check the Statistics tab. The table should have at least two columns.

Search results not structured as a table with valid x-axis or y-axis values cannot generate column or bar charts. For example, using the eval or fields commands might change search result structure.
Statistics table order and chart axes

Column and bar charts handle Statistics table values differently.

Column charts get x-axis values from the first column in the table. The next table columns contain y-axis values.

Bar charts get y-axis values from the first column in the table. The next table columns contain x-axis values.

As an example, any search using the timechart reporting command generates a table where _time is the first column. A column chart generated with this search has a _time x-axis. A bar chart using this search has a _time y-axis.

Single and multiple data series

Column and bar charts can visualize single or multiple data series. The following examples show you how to generate these series.

Single series

A search compares the average number of bytes passed through each source. In this search, the over operator indicates that source is the first table column.

...| chart avg(bytes) over source
The search produces the following table.

Column and bar charts represent this single series differently.

Column chart
source values are used for the x-axis. The y-axis in the column chart is avg(bytes).

Bar chart
avg(bytes) values are used for the x-axis. The bar chart y-axis would represent source field values.
Multiple data series

To generate multiple data series, introduce the `timechart` command to add a `_time` field to search results. You can also change the query to introduce a split-by field.

For example, change the previous single series search by adding `clientip` as a split-by field.

```bash
...| chart avg(bytes) over source by clientip
```

The split-by field produces multiple data series. Each `clientip` is a data series with its own `avg(bytes)` values for each `source`.

To show multiple series in a bar or column chart, use the **Format** menu to configure stacking and multi-series mode.

**Configuration options**

Use the **Format** menu to customize one or more of the following column and bar chart components.

- Chart titles
- Axis titles
- Minimum and maximum axis values
- Use a logarithmic unit scale. This option is helpful when there are very small and very large axis values.
- Specify whether to abbreviate y-axis numerical values. For example, a value of 20,000 will be abbreviated to 20K if you toggle this option to **On**. Only y-axis values can be abbreviated in column and bar charts.
- Chart legend placement and text truncation
- Label rotation

**Multiple series options**

If the chart represents multiple data series, you can also configure the following options.
Multi-series mode

Compare trends across multiple series. Enable the mode to show independent axis ranges for each series.

Stacked charts

Use a stacked chart to see more details for values in a particular field. You can select unstacked, stacked, and 100% stacked bar and column charts. See the following comparison.

<table>
<thead>
<tr>
<th>Stack option</th>
<th>Column or bar appearance</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstacked</td>
<td>Columns or bars for different series appear next to each other.</td>
<td>An unstacked chart is useful for a lower number of series. As the number of series increases, the chart can become more difficult to understand.</td>
</tr>
<tr>
<td>Stacked</td>
<td>Data points within a series appear as segments of a column or bar. The total column or bar value is the sum of all of the segments.</td>
<td>Use a stacked column or bar chart to highlight the relative volume, frequency, or importance of data points in a series. See the stacked chart example below.</td>
</tr>
<tr>
<td>Stacked 100%</td>
<td>Each bar or column is divided into segments representing the distribution percentage for each data value in one series.</td>
<td>Use stacked 100% to show data distributions when there is significant segment size variation in each column or bar.</td>
</tr>
</tbody>
</table>

Create a column or bar chart

Prerequisites
Review the following details about building column and bar charts.

- Data formatting
- Configuration options

Steps

1. Write a search that generates one or more data series.
2. Run the search.
3. Select the Statistics tab below the search bar. The statistics table here should have two or more columns.
4. Select the Visualization tab and use the Visualization Picker to select the column or bar chart visualization.

5. (Optional) Use the Format menu to configure the visualization.

Examples

Bar chart

This search calculates a CPU seconds sum for each processor. The search also sorts the processors with the ten highest sums in descending order.

\[
\text{index=\_internal "group=pipeline" | stats sum(cpu\_seconds) as totalCPUSeconds by processor | sort 10 totalCPUSeconds desc}
\]

The search generates this bar chart.

Stacked column chart

This search portion aggregates events according to code values over time. The query specifies the \_time field and Code field values to include. This query generates a series for each Code field value.

\[
\ldots | \text{timechart count by Code | fields \_time L B N}
\]

The stacked columns show event counts for each code at different points in time. You can compare how many L, B, and N flagged events there were at each point in time.

Line and area charts

Use line and area charts to track value trends over time. You can also use a line or area chart x-axis to represent a field value other than time.
Data formatting

Line charts can represent one or more data series. Area charts represent multiple data series.

If a search generates multiple series, each line or area in the chart appears in a different color.

To make sure that a search generates data series correctly, check the Statistics tab below the search bar. The Statistics table should have at least two columns for a single series, and three or more columns for multiple series.

Statistics table order and chart axes

Line and area charts get x-axis values from the first column in the Statistics table. The next table columns contain y-axis values.

As an example, any search using the timechart reporting command generates a table where _time is the first column. A line or area chart generated with this search has a _time x-axis.

Search results not structured as a table with valid x-axis or y-axis values cannot generate line or area charts. For example, using the eval or fields commands might change search result structure.

Single and multiple data series

Typically, line or area charts represent multiple series. Line charts can also be used for a single data series, but area charts cannot.

Single series

A search compares the average number of bytes passed through each source. In this search, the over operator indicates that source is the first table column.

...| chart avg(bytes) over source
The search produces the following table.
In a line chart, source values are used for the x-axis. The y-axis represents \( \text{avg(bytes)} \) values.

**Multiple data series**

To generate multiple data series, introduce the `timechart` command to add a `_time` field to search results. You can also change the query to introduce a split-by field.

For example, change the previous single series search by adding `clientip` as a split-by field.

```plaintext
... | chart avg(bytes) over source by clientip
```

The split-by field produces multiple data series. Each `clientip` is a data series with its own \( \text{avg(bytes)} \) values for each source.

**Configuration options**

Use the **Format** menu to configure one or more of the following line and area chart components.

- Chart title
- Axis titles
- Null y-axis value handling. Choose one of the following options.
  - Show null data points as a gap. The chart shows markers for any disconnected data points in this case.
  - Connect null data points to zero data points.
  - Connect to the next positive data point.
- Show minimum and maximum y-axis values.
- Use a logarithmic unit scale for y-axis values. This option is helpful when there is a wide range in y-axis values.
- Specify whether to abbreviate y-axis numerical values. For example, a value of 20,000 will be abbreviated to 20K if you toggle this option to **On**. Only y-axis values can be abbreviated in area and line charts.
- Chart legend position and label truncation
**Multiple series options**

If the chart represents multiple data series, you can also configure the following options.

**Multi-series mode**

Compare trends across multiple series. Enable the mode to show independent axis ranges for each series.

**Stacked area charts**

Stacked area charts are available when a search generates multiple data series. Stacking is not available for line charts.

Use a stacked area chart to see more details about a series and how it relates to the entire data set. Review the comparison table here to select a stacking option.

<table>
<thead>
<tr>
<th>Stack option</th>
<th>Column or bar appearance</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unstacked</strong></td>
<td>Areas for different series share the same space on the chart.</td>
<td>An unstacked chart is useful for a lower number of series. As the number of series increases, the chart can become more difficult to understand.</td>
</tr>
<tr>
<td><strong>Stacked</strong></td>
<td>Each series area is shown separately.</td>
<td>Use a stacked area chart to highlight the relative volume, frequency, or importance of a series. See the stacked chart example below.</td>
</tr>
<tr>
<td><strong>Stacked 100%</strong></td>
<td>The chart shows distribution percentage for each series over the whole data set.</td>
<td>Use stacked 100% to focus on data distributions.</td>
</tr>
</tbody>
</table>

**Create a line or area chart**

**Prerequisites**
Review the following details about building column and bar charts.

- Data formatting
- Configuration options
**Steps**

1. Write a search that generates multiple data series. If you are building a line chart you can opt to generate a single data series.
2. Run the search.
3. Select the **Statistics** tab below the search bar. The statistics table here should have two or more columns.
4. Select the **Visualization** tab and use the **Visualization Picker** to select the line or area chart visualization.
5. (Optional) Use the **Format** menu to configure the visualization.

**Examples**

**Line chart**

This search tracks sourcetype frequency over time.

```plaintext
index=_internal | timechart count by sourcetype
```

The search generates multiple data series. The line chart represents each series with a different line.

**Area chart**

Shading in an area chart emphasizes quantities. This example search tracks historical and real-time search volume over time.

```plaintext
index=_internal source=*metrics.log group=search_concurrency "system total" NOT user=* | timechart max(active_hist_searches) as "Historical Searches" max(active_realtime_searches) as "Real-time Searches"
```

The search generates two data series. Each series appears as a different shaded area on the chart.
Stacked area chart

This search tracks throughput for different series over time. The following example uses the Search tutorial data file. To learn more about getting this data into your Splunk instance, see Get the tutorial data into Splunk.

```
sourcetype=access_* status=200 action=purchase categoryId!=NULL | timechart count(categoryId) by categoryId
```

The search generates multiple series. Each series appears as a colored area of the stacked chart. The stacking lets you compare the sums for different series.

Scatter chart

Use a scatter chart to show relationships between discrete data points. Data point distribution can show trends or relationships across two dimensions.

Data formatting

Scatter charts work best with two data series. Use a transforming command to aggregate values. You can use the table command with the following syntax to manage result field ordering.

```
... | table <marker_name_field> <x-axis_field> <y-axis_field>
```

Check the Statistics tab after running the search to make sure that there are three columns in the Statistics table. You can use the table command to
change the order of the columns if needed.

**Configuration options**

Use the **Format** menu to configure one or more of the following scatter chart components.

- Axis titles
- Legend placement and truncation
- Axis scale and interval values
- Axis minimum and maximum values
- Abbreviate y-axis and x-axis numerical values

**Create a scatter chart**

**Prerequisites**
Review the following details about building column and bar charts.

- Data formatting
- Configuration options

**Steps**

1. Write a search that generates two data series.
2. Run the search.
3. Select the **Statistics** tab below the search bar. The statistics table here should have three columns.
4. Select the **Visualization** tab and use the **Visualization Picker** to select the scatter chart visualization.
5. (Optional) Use the **Format** menu to configure the visualization.

**Example**

An analyst creates a scatter chart to track recent earthquake locations, magnitude, and depth.

This search generates a **Statistics** table with three columns. The first column shows earthquake location values. The second column represents earthquake magnitude values, plotted on the x-axis. The third column represents earthquake depth values, plotted on the y-axis.

```
source="earthquake.csv" | table Region Magnitude Depth
```
Use Simple XML to build more complex scatter charts. For more information see the Area, Bar, Column, line, and Scatter Charts and Scatter chart specific properties entries in the Chart Configuration Reference.

**Bubble chart**

Use a bubble chart to visualize multiple series data in three dimensions. Bubble position represents two dimensions of the data series. Bubble size represents the third dimension.

**Data formatting**

To create a bubble chart, start with a search that generates multiple data series. Use this syntax to generate the series.

```
... | <stats_command> <y-axis_field> <x-axis_field> <bubble_size_field>
```

A single group-by field in the query generates a visualization with all bubbles in the same color. To get series colors with the `stats` command, use two group-by fields. This generates a bubble for each unique combination of those two fields. The value of the second field determines the series color.

**Configuration options**

Bubble chart configurations include the following options. Use the **Format** menu to adjust these settings.

- Minimum and maximum bubble marker size
- Axis titles
- X-axis label rotation and truncation
- Axis scale, interval, minimum and maximum values
- Abbreviate y-axis and x-axis numerical values
Create a bubble chart

Prerequisites
Review the following details about building column and bar charts.

- Data formatting
- Configuration options

Steps

1. Write a search that generates three data series.
2. Run the search.
3. Select the Statistics tab below the search bar. The statistics table here should have four columns.
4. Select the Visualization tab and use the Visualization Picker to select the bubble chart visualization.
5. (Optional) Use the Format menu to configure the visualization.

Example

This search aggregates earthquake events by location. It generates data series representing the magnitude, depth, and count for each earthquake location.

source="earthquakes.csv" | stats count by Region, Magnitude, Depth

The search generates a bubble chart where the x-axis and y-axis plot magnitude and depth. The bubble size indicates the relative count value for a particular location.

Event annotations for charts

Event annotations allow you to add context to the trends returned by your time charts. For example, if you have a chart that shows website login errors over the last week, you can add an event annotation that flags the times when your servers were down over that period. If the majority of the login errors occurred...
when your servers were down, you can conclude that the two events are related. Using event annotations in this way gives you the ability to correlate discrete data sets.

Event annotations are presented as colored flags that display time stamp information and custom descriptions in labels when you hover your cursor over them.

Event annotations are supported only for line charts, column charts, and area charts.

If you set an Auto refresh delay in the Edit Search panel, you must manually add the same refresh time in the XML for the event annotation search. For example, if you've set the delay in the UI, it will look similar to this 30 second delay code in the Source tab of the panel for your primary search:

```xml
<refresh>30s</refresh>
<refreshType>delay</refreshType>
```

To ensure that the event annotations also refresh on the same schedule, add the same lines to the secondary search that runs the event annotations.

The following image is an example of a dashboard panel that includes event annotations. The panel shows a chart depicting average search run times overlaid with event annotations taken from a secondary search of internal logs with "WARN" and "ERROR" notifications. The yellow label shows the custom message for the event annotation for "WARN."
Creating event annotations

You create event annotations from the dashboard editor using Simple XML. Event annotation searches are sourced from the data in logs and lookups or from a data source that you have manually added. If you are not familiar with using Simple XML to edit dashboards, see Editing Simple XML.

To add an event annotation to a chart, use the command `search type="annotation"` followed by the query that searches the data for the events to annotate, and the time frame for the query. For example, the following is the Simple XML for the dashboard panel above.

```xml
<dashboard>
  <label>Search Analysis</label>
  <description>Search metrics correlated with user activity and log events</description>
  <row>
    <panel>
      <title>Average Search Run Time with WARN/ERROR event annotations</title>
      <chart>
        <search>
          <query>index=_audit action=search result_count="*" | timechart avg(total_run_time) as avgRunTime</query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
        </search>

        <!-- Secondary search that drives the annotations -->
        <search type="annotation">
          <query>index=_internal (log_level="WARN" OR log_level="ERROR") | eval annotation_label = message | eval annotation_category = log_level</query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
        </search>

        <!-- Customize the event annotation colors based on category name -->
        <option name="charting.annotation.categoryColors">{"ERROR":"0xff3300","WARN":"0xffcc00"}</option>

        <option name="charting.chart">line</option>
        <option name="charting.drilldown">none</option>
        <option name="charting.legend.placement">none</option>
        <option name="charting.lineWidth">1</option>
        <option name="charting.seriesColors">[0x339933]</option>
      </chart>
    </panel>
  </row>
</dashboard>
```
Annotation search fields

Use the following fields to define your event annotation search.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_time</td>
<td>epoch time</td>
<td>Yes</td>
<td>The timestamp of the event.</td>
</tr>
<tr>
<td>annotation_label</td>
<td>string</td>
<td>No</td>
<td>The message that is displayed in the annotation label.</td>
</tr>
<tr>
<td>annotation_category</td>
<td>string</td>
<td>No</td>
<td>Use this field to group annotation events by type.</td>
</tr>
<tr>
<td>annotation_color</td>
<td>string</td>
<td>No</td>
<td>Use this field to assign a color to an annotation event. Colors must be specified using either HEX (0xffff00) or RGB (255,0,0) notation.</td>
</tr>
</tbody>
</table>

If you use this field to specify a color for an event annotation, it will override the chart configuration property, `charting.annotation.categoryColors`. If you have specified the field `annotation_category` in your search, you can use the `charting.annotation.categoryColors` property instead of `annotation_color` to assign colors to the markers and labels of the categories returned.

For more details about this property, see `charting.annotation.categoryColors` in the Chart Configuration Reference.
Chart display issues

This topic covers display issues using chart visualizations.

Searches with non-transforming commands

You cannot render charts using searches that do not include transforming commands, such as the following options.

chart
timechart
stats
eval

For more information, see About Transforming commands and searches in the Search Manual.

Time charting

You can only plot time-based data using the `timechart` command, which generates the `_time` based output. If you try to plot a time-based `series` using any other transforming search command, or rename the `_time` value, the chart treats the timestamp data as a series of strings.

Data truncation

To avoid browser performance impacts, Splunk software limits the the amount of data rendered in an individual chart. When search results exceed limits, a message appears with the chart indicating that data was truncated.

Depending on your Splunk instance type and dashboard editing permissions, you can change default rendering behavior using configuration settings and Simple XML.

Users with editing permissions can modify Simple XML data truncation settings for individual charts in a dashboard.

Splunk Enterprise administrators can also add or edit local copies of `visualizations.conf` and `web.conf` to configure truncation settings for charts across a Splunk deployment.

Use the following table to compare configuration options.
<table>
<thead>
<tr>
<th>Rendering limit type</th>
<th>In</th>
<th>Setting</th>
<th>Default</th>
<th>Where to configure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total data points</td>
<td>One chart in a dashboard</td>
<td>charting.chart.resultTruncationLimit</td>
<td>50000</td>
<td>Simple XML</td>
</tr>
<tr>
<td>Data points per series</td>
<td>One chart in a dashboard</td>
<td>charting.data.count</td>
<td>10000</td>
<td>Simple XML</td>
</tr>
<tr>
<td>Data points per series</td>
<td>One chart type</td>
<td>data_sources.primary.params.count</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Replaces the deprecated jschart_results_limit setting in web.conf.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>When set to a numeric value, overrides charting.data.count Simple XML setting in charts of this type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total data points</td>
<td>All charts in all browsers</td>
<td>jschart_truncation_limit</td>
<td>No default. Specify to override individual browser settings.</td>
<td>web.conf</td>
</tr>
<tr>
<td>Data series</td>
<td>All charts</td>
<td>jschart_series_limit</td>
<td>100</td>
<td>web.conf</td>
</tr>
</tbody>
</table>

**Simple XML data truncation options**

You can adjust data truncation in individual charts using Simple XML.

**Data point limit for one chart**

You can configure the maximum number of points that can be plotted in a specific chart by editing the Simple XML for the chart. In the `<chart>` element, edit the `charting.chart.resultTruncationLimit` property as described in Area, bar, column, line, and scatter charts in the Chart Configuration Reference.
**Data point limit per series in one chart**

You can limit the number of search result data points rendered per series in a chart. Edit the `charting.data.count` Simple XML setting to override the default value of 10000 data series.

**Configuration file data truncation settings**

Splunk Enterprise admins can add or edit truncation settings in `$SPLUNK_HOME$/etc/system/local/web.conf` or `$SPLUNK_HOME$/etc/system/local/visualizations.conf`.

**Data point limit for all charts in one or more browsers**

The `web.conf` configuration file specifies the maximum number of points that can be plotted for charts in various browsers. All browser limits default to 50000 data points.

You can override individual browser settings or add a `jschart_truncation_limit` setting in `$SPLUNK_HOME$/etc/system/local/web.conf` to define a limit for all browsers. The `jschart_truncation_limit` overrides any individual browser settings.

**Note:** The `charting.chart.resultTruncationLimit` Simple XML option overrides this limit in an individual chart.

**Data series limit for all charts**

You can limit the number of data series that charts can render. Add or edit the `jschart_series_limit` setting in `$SPLUNK_HOME$/etc/system/local/web.conf` to override the default value of 100 data series.

If search results exceed this limit, a chart displays only the number of series that this limit allows. A warning message appears to indicate that the chart is showing truncated search results. For example, if the `jschart_series_limit` is 40 and a search returns 50 data series, a chart renders only the first 40 series.

**Data point limit per series for one chart type**

Limit the number of search result data points rendered per series in a specific chart type. Edit the `data_sources.primary.params.count` setting in `visualizations.conf` to override the default value of 1000 result rows.
For example, you can set the data point limit per series in pie charts. Edit the [pie] chart type stanza in $SPLUNK_HOME/etc/system/local/visualizations.conf:

[pie]
data_sources.primary.params.count = 10

As of the latest software version, the web.conf jschart_results_limit data point per series setting for all charts is deprecated. Use data_sources.primary.params.count instead.

**Data series and data point limit precedence**

In the case of an individual chart, if the jschart_series_limit and the charting.data.count Simple XML options combine to indicate a number greater than the jschart_truncation_limit in web.conf, then data points per series are reduced to meet the jschart_truncation_limit setting.

For example, you might have a jschart_series_limit of 10 and a charting.data.count limit of 100 in a chart. When multiplied, these two settings indicate a 1000 total data point limit for the chart. If the js_chart_truncation_limit is 800, however, then data points per series are reduced to meet the 800 total point limit.

To override the js_chart_truncation limit on all charts, you can use charting.chart.resultTruncationLimit Simple XML option to change the limit for an individual chart.

**About editing configuration files**

For more information about working with the web.conf and visualizations.conf configuration files, see the following topics in the Admin Manual.

- How to edit a configuration file
- web.conf spec file
- visualizations.conf spec file

**Category limit**

When you are plotting data by category, Splunk software limits chart label display. This limit differs for the horizontal axis (X-axis) and the vertical axis (Y-axis).
The X-axis must have at least 20 pixels available for each label. The Y-axis must have at least 15 pixels available. If the requisite pixels are not available, the labels do not display.

You can zoom into the X-axis to view labels that are hidden by the category limit. See Pan and zoom chart controls for details.
Single Value

Overview

Use a single value visualization to show a metric and its context. Single value visualizations display results and context for searches returning a discrete number.

A single value can be a count or other aggregation of specific events. For instance, this visualization shows sales for a popular lemonade stand.

A caption, unit notation, and range colors add emphasis. A trend indicator to the right of the value and a sparkline underneath show how data has changed over time.

To start working with single value visualizations, see the following topics.

- Generate a single value
- Customize a single value

Generate a single value

Learn how to write a query to generate a single value visualization.

Single value visualizations work best for queries that create a time series chart using the `timechart` command or aggregate data using the `stats` command.

Use `timechart` to generate a single value

This search and visualization use `timechart` to track daily errors for a Splunk deployment.
To access sparklines and trend indicators, it is important that the search includes the `timechart` command. Using `timechart` means that time series data becomes available to sparkline and trend indicator processing.

**Note:** If you pipe to `stats` as part of a full `timechart` query, the visualization does not include a sparkline or trend indicator.

**Use stats to generate a single value**

If you use the `stats` command to generate a single value, the visualization shows the aggregated value without a trend indicator or sparkline. As an example, this query and visualization use `stats` to tally all errors in a given week.

```
index = _internal source = "*splunkd.log" log_level = "error" | stats count
```

**Queries and time ranges for single values**

It is important to set up the single value query that best drives the visualization that you expect.

- Search for a single value to avoid unexpected results in the visualization. In the Dashboard Editor, you can select single value visualizations even if a search returns multiple values. In this case, the single value visualization uses the value in the first cell of the results table.

- The time range picker and the query command work together to generate the results for a single value visualization. A query using `stats` results in a visualization showing the aggregated total of results in the time range. A query using `timechart` generates a visualization showing the most recent result within that range.

For details about the `stats` command, see stats in the *Search Reference*.
For details about the `timechart` command, see `timechart` in the *Search Reference*.

### Queries to generate a sparkline and trend indicator

A sparkline appears below a single value generated with the `timechart` command. It shows increases and decreases in a metric over the time range you specify in a search.

This visualization shows results for a search over the past week’s data. Using the time range picker to select **Week to date** means that the sparkline reflects the data changes over the last seven days.

This visualization shows results for the same search over the past day’s data. Using the time range picker to select **Today** means that the sparkline shows data changes over the past twenty-four hours.

A trend indicator appears to the right of a single value generated with the `timechart` command. It shows recent data behavior over a customizable time range. The trend indicator is composed of a number and an arrow to represent what happened most recently in the data.

Depending on data behavior, the trend arrow can point up, down, or directly to the side to show no change. By default, the trend indicator value evaluates to the difference between the two most recent values in the results. You can change the trend time window in the Format menu’s General settings panel or by adjusting the span parameter for `timechart`.

**Note:** To include sparklines and trend indicators in a visualization, it is important that the search includes a `timechart` command.

### Customize a single value

Learn how to configure single value visualization components.
Value ranges and colors

Colors can emphasize range values or trends in a single value visualization. In the Format menu, you can choose whether to use colors. If you opt to use colors, you can select whether to color by value or trend.

**Note:** For queries using `stats` to aggregate results, only the color by value option is available.

You can also adjust the color mode to change whether colors appear in the foreground or background.

Depending on the color mode you choose, coloring a single value generated with `timechart` by value means that the sparkline and trend indicator appear black (for foreground color) or white (for background color).

**Color by value**

Coloring by value is available for single value visualizations generated with either `stats` or `timechart`. Color by value means that the single numeric value in the visualization changes color based on the value the search generates and the range for that value. For example, if you map a value range from 30-50 to the color yellow, then a single value of 35 appears yellow.

You can adjust value ranges for the query to change how different results are visualized. By default, there are five ranges and colors for coloring by value. You can add or remove ranges, modify the values for each range, and change the colors associated with each range using the Format menu.

For example, this `timechart` generated single value visualization shows color by value and has the background color mode selected.

**Color by trend**

Coloring by trend is available for single values generated with a query including the `timechart` command. Coloring by trend means that the sparkline and trend indicator in this visualization change color to show changes in data. By default, positive changes make the sparkline and trend indicator appear green, while
negative changes make them red. When results show no change, the trend color is black.

For example, this visualization shows color by trend and has the foreground color mode selected.

You can reverse the settings for trend colors in the Format menu. You can also specify a different trend time window for the visualization.

**Migration for rangemap settings in existing single value visualizations**

Existing single value visualizations might use a query with the `rangemap` command to configure ranges and colors.

By default, a single value visualization has this color mapping configuration for ranges.

- **low**: green
- **guarded**: blue
- **elevated**: yellow
- **high**: orange
- **severe**: red

**Caution:** As support for the `rangemap` command is limited, it is not recommended for building new single value visualizations. Queries using `rangemap` currently generate a single value, but UI configurations override the query-based settings.

For existing single value visualizations, it is recommended to migrate `rangemap` command settings out of the query. Replace query-based settings with equivalent range and color settings in the Format menu **Color** panel.

**Captions and units**

Use the Format menu’s General options panel to add a caption for a single value visualization. You can specify a unit of measurement and its position in the Number Format panel. For instance, you can add $ before a value reflecting sales in the United States or MB after a value tracking data transfers.

**Note:** If you are migrating from earlier versions of Splunk software and your visualization includes Before and After labels, the Format menu shows prompts
to update label and unit text using the Unit and Caption fields.

Captions

Captions add descriptive context to a single value visualization. To add a caption, select the Format menu General panel. Use the caption text field here to enter a description. Captions appear below the single value.

Units

Units can indicate standard measurements for single values. To add a unit to the visualization, select the Format menu Number Format panel and edit the Unit field. You can choose whether a unit appears before or after the value. It is recommended to keep unit text to five characters or fewer. Use a caption for longer text.

Number formatting

If you are working with a large single value or one that requires precision, you can change the number formatting for the visualization. In the Format menu Number Format panel, you can choose thousand separators or different degrees of decimal precision.

Drilldown

By default, drilldown is disabled for single value visualizations. For more information about enabling and configuring drilldown, see Use drilldown for dashboard interactivity.
Gauges

Using gauges

Use a radial, filler, or marker gauge to map a value in relation to a range. A gauge visualization provides metric status and range information that you can interpret quickly. You can use a real-time search to generate a gauge tracking value fluctuations as they occur.

Data formatting

To generate a gauge, use a search that returns a single numerical value. For example, use a search that returns an event count for events with a specific field value in a time period or real-time window. If you are using a real-time search, the range marker moves to show the metric changing over time.

Gauge types

All gauge types visualize a single aggregated metric.

For example, this search aggregates error log events.

```
index=_internal source="*splunkd.log" log_level="error" | stats count
as errors
```

The search can generate any of the available gauge types.

Radial gauge

A radial gauge includes a round value scale and a pointer to show the current value on the scale. The current value also appears at the bottom of the gauge. You can configure a radial gauge to use specific colors for each value range in the scale.
If the search generates a current value outside of the configured minimum or maximum range, the gauge pointer bounces at the lower or upper end of the value scale.

**Filler gauge**

A filler gauge includes a value scale container that fills and empties as the current value changes. The fill level shows where the current value is on the value scale.

The current value also appears inside the filled portion of the gauge. The container appears empty for a value lower than the minimum and full for a value higher than the maximum.

**Marker gauge**

A marker gauge shows value ranges and colors with a marker that moves to indicate the current value.

If the search generates a current value outside of the configured minimum or maximum range, the marker bounces at the lower or upper end of the value scale.

**Configuration options**

Use the **Format** menu to configure gauge style and color ranges.
**Color ranges**

Use the Format > Color Ranges panel to select manual or automatic color range configuration. By default the first three ranges are green, yellow, and red.

Set the Color Ranges handling to Automatic if the query includes the `gauge` command for range configuration.

If the query includes `gauge`, Format menu range configurations override the `gauge` command settings in the query.

**Create a gauge visualization**

**Prerequisites**
Review the following details about building column and bar charts.

- Data formatting
- Gauge types
- Configuration options

**Steps**

1. Write a search that generates a single aggregated value.
2. Run the search.
3. Select the Visualization tab and use the Visualization Picker to select a radial, filler, or marker gauge.
4. (Optional) Use the Format menu to configure the visualization.
Maps

Mapping data

There are several options for visualizing data that includes geographic information.

A Choropleth map uses shading to show relative metrics, such as population or election results, for predefined geographic regions. You can also create other visualizations with geographic data, such as cluster maps or charts.

Getting started

Use the following topics to learn about creating Choropleth maps and other geographic visualizations.

- Generate a Choropleth map
- Configure a Choropleth map
- Cluster maps

See also

To learn about geospatial lookups, see Configure geospatial lookups in the Knowledge Manager Manual.

Generate a Choropleth map

Geographic visualizations aggregate events by location. Location names might already be included in events. You can also use a search to generate locations from signed degree latitude and longitude coordinates in each event.

Choropleth maps have specific data and component requirements. A search
uses the data and components to generate a Choropleth map.

**Working with map components and geographic data**

Review the following component and data details before running a search.

**Components for building geographic visualizations**

These components are required for creating geographic visualizations.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Available options</th>
</tr>
</thead>
</table>
| Data with geographic coordinates | Geographic visualizations start with data that includes location information for each event. This data can come from several sources, including a sensor or forwarded data source. | Either:  
• Data with signed degree latitude and longitude coordinates.  
• Data with location names that match the location names in a lookup. |
| Lookup table file            | A lookup table file defines region boundaries, such as the boundaries of each state in the United States.                                                                                                     | Either:  
• Built-in files for the United States, `geo_us_states`, and countries of the world, `geo_countries`.  
• Upload a KML or KMZ file for other places. Upload the file to the `Lookup table files` manager page. |
| Geospatial lookup            | A geospatial lookup matches coordinates to region definitions in the lookup table file.                                                                                                                      | Either:  
• Built-in lookups for the United States and for world countries.  
• Create a geospatial lookup. For more |
Use normalized data

Choropleth maps work best when data is normalized. Normalization adjusts data to more accurately reflect the metric that you are visualizing. For example, a Choropleth map can compare sales performance in two cities with significantly different populations. Using normalized data to generate this map means that the population difference alone does not determine how the cities' sales compare on the map.

Test custom lookup files

If you are working with a custom lookup table file and geospatial lookup, you can use the `inputlookup` command to make sure that they are working properly before building a Choropleth map.

For more information, see Configure geospatial lookups in the Knowledge Manager Manual.

Show all features on a map regardless of data coverage

If you have a data set that does not include values to aggregate for every feature in a Choropleth map, you can use the `geom` command `allFeatures` parameter to show all shapes on the map when it renders.

For more information, see `geom` in the Search Reference.

Create the search

A search coordinates data, a transforming search, and a geospatial lookup to build a Choropleth map or other geographic visualization. The following steps show you how to create a Choropleth map search. Optionally, you can use the steps to generate other visualizations for geographic data.

Prerequisites

Make sure that you have the correct data and components for building a geographic visualization. See Data and component requirements.
Steps
Run each portion of the search as you build it to ensure that it is working correctly. Depending on the visualization you are creating and the components that you have, some steps are optional.

1. Indicate an events data source.
   
   ```
   source=my_data.csv | 
   ```
   Start with an events data source that has signed degree geographic coordinates or location name fields. For example, here is one record in a .csv file listing retail locations for a business. This file includes latitude and longitude coordinates for each record.

   ```
   Store Number,Name,Facility ID,Products,Services,Country,Latitude,Longitude
   12345,Buttermilk Tea Shop,54321,"Tea, Cake",Wi-Fi,US,43.031873,-71.073203
   ```

2. (Optional) Add a lookup.
   ```
   lookup geo_us_states longitude as Longitude, latitude as Latitude | 
   ```
   If the events data already includes location name or featureId fields, you can skip this step.

   The lookup uses the geographic coordinates to generate featureId and featureCollection fields for events. A featureId is the name of a geographic feature that includes a particular set of geographic coordinates, such as a state or city name. By default, the featureCollection is the lookup definition name.

   After adding the lookup and running the search, check the available Selected Fields or Interesting Fields to ensure that featureId is listed. If it is not, then the lookup did not generate the featureId from the geographic coordinates. Fields are case-sensitive.

3. Use a transforming command.
   ```
   stats count by featureId | 
   ```
   Aggregate the data using the lookup's geographic output field, featureId. If you did not need a lookup, aggregate by the location name field already in the events data.
4. **(Optional) Select and configure a visualization.**
You can use the search to generate non-map visualizations for geographic data. If you are not building a Choropleth map, the search is complete. Use the **Visualization Picker** to select a visualization type. Use the **Format** menu to configure it.

5. **(Optional) Use geom to complete the Choropleth map search.**
If you are building a Choropleth map, add the `geom` command and pass in the lookup name for the `featureCollection` parameter.

Depending on whether the events include a `featureId` field, select one of the following options.

<table>
<thead>
<tr>
<th>Events have</th>
<th>Next steps</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>featureId</code> fields</td>
<td>1. Use the lookup to which those fields belong.</td>
<td><code>geom geo_us_states</code></td>
</tr>
<tr>
<td>Location names, no <code>featureId</code> field. This might be the case if you skipped the lookup earlier.</td>
<td>1. Use a lookup that contains the location names. For example, if events have US state names, use <code>geo_us_states</code>. 2. Indicate which events field <code>geom</code> should interpret as the <code>featureIdField</code>.</td>
<td><code>geom geo_us_states featureIdField=&quot;State&quot;</code></td>
</tr>
</tbody>
</table>

For more information and advanced options for Choropleth map queries, see `geom` in the **Search Reference**.

**Example search**

The full search assembled in the previous steps looks like this.

```
source=my_data_source.csv
| lookup geo_us_states longitude as Longitude, latitude as Latitude
```
Configure a Choropleth map

To review or change Choropleth map configuration, select the Format menu and one of the following settings panels.

General

Adjust general settings including initial geographic coordinates and zoom on scroll. You can also opt to hide the map legend.

Colors

Color mode and data bin settings determine how a Choropleth map uses color to visualize data. Select a color mode and configure data bins in the Colors panel.

Color modes

<table>
<thead>
<tr>
<th>Color mode</th>
<th>Description and use cases</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categorical</td>
<td>Color regions by category value. For example, you can track top product purchases by state. If multiple states have the same top product, they share a color.</td>
<td></td>
</tr>
<tr>
<td>Sequential</td>
<td>Color regions with light to dark shades of a single hue. This mode helps you find regions where a metric is particularly high.</td>
<td></td>
</tr>
<tr>
<td>Divergent</td>
<td>Color regions in light to dark shades of two distinct hues. This mode shows regions where a metric is particular high or low. Shading fades as regional metrics approach the middle of the range.</td>
<td></td>
</tr>
</tbody>
</table>

Data bins

Aggregated data values are divided into a set of bins. Each bin corresponds to a specific value range and has a unique color or shade. You can adjust the number of bins and bin color assignments for the selected color mode.
The Choropleth map legend to the right of the map shows bins with their colors and value ranges.

**Shapes**

A shape corresponds to an individual region on a Choropleth map. For example, each state in a Choropleth map of the United States is a shape. You can adjust shape opacity and borders.

**Tiles**

Tiles represent map background features, such as oceans. Show or hide tiles.

**Cluster maps**

Use the cluster map visualization to plot aggregated values on a map.

**Data formatting**

To generate a cluster map, use the `geostats` command. The `geostats` command generates events that include latitude and longitude coordinates for markers. It is similar to the `stats` command, but provides options for zoom levels and cells for
mapping.

For more information, see geostats in the Search Reference.

**Configuration options**

Use the **Format** menu to adjust the following cluster map components.

- Tile appearance and source
- Cluster marker appearance
- Zoom on scroll behavior

**Example**

The following search generates a map showing California earthquakes of magnitude greater than 3 for the past 30 days.

```
index=main mag>3 | geostats latfield=latitude longfield=longitude count
```

When a user clicks on a cluster indicating earthquake data, a search launches using the latitude and longitude boundaries of that cluster.

```
index=main mag>3 | search latitude>=36.21094 latitude<36.56250
longitude>=-122.34375 longitude<-121.64062
```
Trellis Layout for Visualizations

Use trellis layout to split visualizations

Trellis layout lets you split search results by fields or aggregations and visualize each field value separately.

This is a single value visualization with trellis layout applied. It splits customer purchase results by product category values. Users can see how the purchase metric varies for different product types.

Use cases

Use trellis layout to make value differences in a given data dimension more visible.

*Highlight outlying values*

Trellis layout can help to make outlying field values more noticeable.

For example, your dashboard users might want to track status across multiple servers in a network. A single value visualization with trellis layout can show the status of multiple servers at once. Servers with unusual status values stand out.

*Compare trends for a specific metric*

You can split search results so that it is easier to compare different field values visually.

For example, you can apply trellis layout to a bar chart showing recent customer activity for different product types. Splitting on the customer action field lets you scan variations in purchase frequency across different product types.
Monitor multiple resources with one search

Trellis layout can be helpful if you want to monitor multiple resources without creating and running multiple searches or generating multiple visualizations. You can use one search to generate metrics for each resource in a category or group and then split the visualization on the field that you are tracking.

Data formatting for trellis layout

You can use trellis layout to split your search results on a field. You can also split results on an aggregation if your search includes two or more aggregations, such as a count or sum.

Before you enable trellis layout, consider the comparisons or trends that you want to provide at a glance. Make sure that your search results include the fields or aggregations that represent these values.

Generating split fields

To use trellis layout, make sure that your search results include the field that you want to use for splitting the visualization. The split field is additional to any fields that you might need to generate the visualization without trellis layout. For example, you can generate a single value visualization using the following search.

```
index=_internal | stats count
```

To use trellis layout, adjust the search to generate an additional field for splitting the visualization.

```
index=_internal | stats count by sourcetype
```

You can split the single value on the sourcetype field to show a count for each sourcetype in your search results.

Use additional fields to add insight

Depending on your use case, you can generate multiple result fields to add further data dimensions to each visualization segment.

As an example, you can aggregate recent retail website data by customer action and product type. With these fields in your search results, you can use one of them to split the visualization. Users can use the split visualizations to compare
customer actions across product types or see how product types relate to customer actions.

For more information on creating searches, see Statistical and charting functions in the Search Reference.

**Access the trellis layout menu**

You can access the trellis layout menu when creating a visualization or editing a visualization in a dashboard.

Trellis layout is not available for table visualizations or cluster maps.

**Access the menu from a visualization**

If you are building a visualization on the Search page, you can access the Trellis configuration menu on the Visualization tab.

**Access the menu from a dashboard**

1. From a dashboard, click Edit to open the dashboard editor.
2. Find the panel where you want to apply trellis layout.
3. Click the "More actions" icon and select Trellis.
Configure trellis layout

The trellis layout menu lets you select "Split by" result fields or aggregations and configure segments.

Select a split field or aggregation

When you split the visualization by a field or aggregation, a separate visualization segment appears for each value in the selected field.

Result fields generated with the `eval` command appear in the aggregations list.

If you do not see the split field or aggregation that you want to use in this list, adjust the search to make sure that it generates the field in your search results. You might need to adjust your search to return additional fields that are not necessary for generating the visualization without trellis layout.

Adjust segment size

Select one of the segment size options. Segment size affects panel data density. Panels can show more small sized segments at once. Larger segments can help users make more detailed visual comparisons depending on the number of segments.

Small segments

Large segments
Configure scale sharing

The trellis layout menu includes a scale sharing option for the following visualization types.

- Line and area charts
- Bar and column charts
- Choropleth map

Select "Shared" to use the same axis or legend value ranges in all segments. Select "Independent" to use axis or legend value ranges scaled for the values in each segment.

Segment formatting and appearance

Use the Format menu or Simple XML to configure visualization segment appearance. Format a visualization with trellis layout in the same way that you format a visualization without trellis layout. Each segment gets the format configurations that you apply.

Choropleth map scroll and zoom

In Choropleth maps with trellis layout, you might need to scroll and zoom a segment to focus it on a specific location. Limit scrolling and zooming to one segment to avoid triggering display changes in other segments. After focusing this segment, open the Format menu and click "Populate with current map settings" to update the focus in all segments.

Trellis layout and dashboard display

Segment density

The number of segments that trellis layout generates varies according to the number of split field or aggregation values in your search results. Dashboard panels might include a scroll bar if there are too many visualization segments to show at once.
Use the **Trellis** menu to adjust the segment size and show more segments in the panel. You can also use the dashboard editor to drag the panel size or change the panel height option in Simple XML.

To change the order in which segments appear, adjust the search to sort or change search result order.

**Panel and row best practice**

Separate panels using trellis layout into their own dashboard rows. Displaying additional content in the same row can constrain trellis layout content and make the dashboard difficult to scan.

<table>
<thead>
<tr>
<th>Avoid</th>
<th>Best practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additional panels in the same row constrain display.</td>
<td>Put the panel with trellis layout in its own row.</td>
</tr>
</tbody>
</table>

**Drilldown in trellis layout**

Use the drilldown editor or Simple XML to enable and configure drilldown. After you apply Trellis layout to a visualization, drilldown is available in each visualization segment.

**Drilldown options**

Typical drilldown actions, such as linking to a search or an external URL, are available for visualizations using Trellis layout. Depending on the behavior that you want, you can configure drilldown to capture and use details from the
element and the visualization segment that a user clicks.

**Example**

This drilldown links to a search. By default, the secondary search modifies the original search to include field values from the clicked segment.

For example, trellis layout splits a retail activity visualization by customer action. If a user clicks the "ARCADE" category column in the "purchase" customer action segment, a secondary search using these field values opens.

**Clicked visualization segment and column**

**Search generating the visualization**

```plaintext
...| stats count by action, categoryId | rename categoryId as "category"
```

**Drilldown search**

```plaintext
...| action=purchase | rename categoryId as "category" | search category=ARCADE
```

**Predefined tokens for trellis split fields**

If your trellis layout splits on a search result field, you can use the `trellis.name` and `trellis.value` predefined tokens to access the split field name and value from a clicked visualization segment.

You can pass these values to a drilldown target, such as a form or external URL. This example drilldown links to an external retail website, using the `trellis.value` token to pass in the product field value from the clicked segment.
Aggregations are not available in predefined tokens.

**Limitations**

- Fields generated with the `eval` command appear as aggregations in the trellis layout configuration menu.
- Trellis layout is not available for table visualizations or cluster maps.
- A predefined token is not available for aggregations used for splitting a visualization.
- Visualizations using trellis layout do not render in dashboard PDFs.

**Configure trellis layout in Simple XML**

Use the following Simple XML options to configure trellis layout.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trellis.enabled</td>
<td>Boolean</td>
<td>0</td>
<td>Enable or disable trellis layout. Defaults to 0 (disabled).</td>
</tr>
<tr>
<td>trellis.scales.shared</td>
<td>Boolean</td>
<td>1</td>
<td>Indicate whether to share scale for axes in line, area, column, and bar charts or value ranges in Choropleth map segments. Use one of the following values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Shared scales</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Independent scales</td>
</tr>
<tr>
<td>trellis.size</td>
<td>String</td>
<td>medium</td>
<td>Configure the visualization segment size. Segment size affects panel display density for the split visualization. Use one of the following values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• small</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• large</td>
</tr>
<tr>
<td>trellis.splitBy</td>
<td>Result field name</td>
<td>N/A</td>
<td>Indicate the search result field or aggregation name to use for splitting the visualization.</td>
</tr>
</tbody>
</table>


Example

Research sales trends

An analyst for an online retailer researches customer actions by product category. They use the following search to find recent customer action events by product category.

```plaintext
source=recent_sales_data action != NULL
| rename categoryId as "category"
| chart count by category, action
```

The search generates the following column chart, showing customer actions across all product categories.

The analyst can use trellis layout to split the visualization into separate column charts for each customer action category. Trellis layout lets the analyst compare trends in the product types associated with each action.
Get Started with Dashboards

Dashboard overview

Create new dashboards or edit existing ones.

For a quick glance at the most common use cases and commands for creating dashboards, note that you can access the Splunk Dashboards Quick Reference guide by clicking the link in Getting started.

The dashboard and form workflow

Working with dashboards includes one or more of the following tasks.

Build dashboards

- Create a new dashboard
- Add new visualizations to a dashboard

For more information on building dashboards, see Create a dashboard

Edit dashboards

- Add a panel to a dashboard
- Edit dashboard panels and panel visualizations
- Manage dashboard searches

For more information on editing dashboards, see Edit dashboards

Convert a dashboard to a form

- Add user inputs to a dashboard to convert it to a form
- Edit forms
- Work with user input settings

For more information on forms, see Create and edit forms

Customize Simple XML

- Edit Simple XML source code to customize a dashboard or form.
Add interactive and dynamic behavior

- Use tokens to capture and transfer data.
- Add event handlers to implement dynamic behavior.

For more information on event handlers and tokens, see Use drilldown for dashboard interactivity and Token usage in dashboards

Tools and frameworks

To build and edit dashboards, use one or more of the following tools and frameworks.

Dashboard editor user interface

Build and edit dashboards using the Splunk Web user interface.

Simple XML

Dashboards use Simple XML source code to define their content and behavior. You can use the dashboard editor in Splunk Web to edit this source code.

To learn more, see Editing Simple XML.

Developer options

Splunk Enterprise users can implement additional dashboard customizations.

- Extend Simple XML using CSS and JavaScript.
- Convert a Simple XML dashboard to HTML and use JavaScript to implement customizations including inputs and REST API access.

For more information, see the following Splunk developer portal resources.

- Convert Simple XML dashboards to HTML
- Modify dashboards using Simple XML
- About SplunkJS stack
- Web Framework overview

Dashboards converted to HTML have some editing limitations in Splunk Web. They also cannot be exported to PDF.
Examples

The Dashboard Examples app on Splunkbase provides many dashboard implementation examples, including source code. Install the app to view and interact with the example dashboards.

Deprecated options

The following dashboard framework options are deprecated as of version 6.3.0.

<table>
<thead>
<tr>
<th>Option</th>
<th>For more information see</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced XML</td>
<td>Advanced XML deprecation</td>
</tr>
<tr>
<td>Module System (Deprecated as part of Advanced XML).</td>
<td>Advanced XML deprecation</td>
</tr>
<tr>
<td>Django Bindings</td>
<td>Django Bindings Deprecation Notice</td>
</tr>
</tbody>
</table>

About the dashboard editor

Use the Splunk Web dashboard editor to create and edit dashboards. The dashboard editor provides access to an editing user interface and Simple XML source code.

Editing user interface

You can create and update dashboards and panels in the editor user interface.

To learn more, see Create dashboards and Edit dashboards.

Source code editor

Use the dashboard editor to access and edit Simple XML source code.
The editor provides validation, error messaging, and warnings as you make changes.

**Keyboard shortcuts**

Keyboard shortcuts consistent with Ace code editor shortcuts are available in the dashboard editor.

You can format Simple XML source code by using **Command + Shift + F** on a Mac or **CTRL + Shift + F** on Windows.
Build and Edit Dashboards in Splunk Web

Create a dashboard

Dashboards are created in the context of a particular app. For example, if you are using the Search and Reporting app, dashboards use this app context.

After you create a dashboard, you can modify its permissions to share or manage access for other users. You can also modify the app context.

Steps

1. Use one of the following options.

<table>
<thead>
<tr>
<th>From</th>
<th>What to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboards page</td>
<td>Click Create new dashboard</td>
</tr>
<tr>
<td>Saving a visualization</td>
<td>1. Select Save as &gt; Dashboard panel</td>
</tr>
<tr>
<td></td>
<td>2. Click New to create a new dashboard using this panel.</td>
</tr>
</tbody>
</table>

2. Provide a Title, ID, and Description for the dashboard.
3. Specify permissions.
4. Save the dashboard. Use one of the following options.

<table>
<thead>
<tr>
<th>From</th>
<th>Click</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboards page</td>
<td>Click Create Dashboard</td>
</tr>
<tr>
<td>Saving a visualization</td>
<td>Click Save</td>
</tr>
</tbody>
</table>

5. Add panels, convert the dashboard to a form, or edit dashboard content.

For more information, see the following topics.

- Add panels to dashboards
- Edit dashboards
- Create and edit forms
- Configure dashboard permissions
Working with dashboard panels

A dashboard contains one or more panels. Dashboard panels use searches to generate visualizations. Select a panel type depending on the type of search behavior and configuration options that you want.

Inline panel

An inline panel contains a search directly in its source code. The search generates the results rendered in the panel visualization. You can edit an inline search using the dashboard editor.

Panel from a report

This panel type uses the saved search and visualization from a report.

When using a panel from a report, you cannot modify the search string in the panel, but you can change and configure the visualization. If the report search changes, the panel using that report updates accordingly.

Access and performance considerations

Depending on your deployment, you might need to adjust report configurations for dashboard performance and access. For instance, depending on your permissions, you can control whether the report is accelerated, scheduled, or embedded. You can also change the report permissions.

User context

Reports in dashboard panels can run as the report owner or the report user. These settings can affect data visibility and concurrent search limits.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Data visibility impact</th>
<th>Concurrent search limit impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run as owner (default)</td>
<td>Run the report using the permissions of the user who created the report.</td>
<td>A report run with owner permissions renders search results that some users might not otherwise have permission to see. In some cases, you might</td>
<td>If a dashboard containing a report-backed panel loads multiple times simultaneously, it</td>
</tr>
</tbody>
</table>
Scheduled reports always run using report owner permissions. want to provide this kind of access. In other cases, you might want to restrict search result visibility. can impact the report owner's concurrent search limit. When the limit is reached, the report scheduler causes additional report search runs to be queued for later execution.

Dashboard users might see slower panel loading and the report owner might not be able to run searches and reports immediately.

| Run as user | Run the report with the permissions of the user viewing the dashboard. Scheduled reports cannot run with user permissions. If the report accesses data that the current user does not have permission to see, the panel does not render those results. When the report runs, it counts against the concurrent search limit of the user loading the dashboard, not the report owner. |

Use scheduled reports for dashboard panels when possible

Back dashboard panels with scheduled reports whenever possible to reduce search processing load for your Splunk deployment.

Benefits of scheduled reports
Not using scheduled reports can impact search processing loads and concurrent search limits. For example, If fifty users access a particular dashboard, panels not backed by scheduled reports cause their reports to rerun fifty times.
Scheduled reports do not require the search to run each time a user loads the dashboard. Panels backed by scheduled reports show results from the last scheduled run of the report.

**Using real-time scheduled reports**
To show dashboard users the most current results, back dashboard panels with real-time scheduled reports. This report type runs in the background at all times. It does not launch a new report instance each time a user loads the dashboard. Instead, it shows results for the currently running real-time scheduled report.

**Prebuilt panel**
Save and reuse Simple XML panels in multiple dashboards. You can display a prebuilt panel in a dashboard by using a reference to the panel. Edit the panel directly to change the title, search, or visualizations in it.

**Additional information**
- To learn about using the dashboard editor to add or edit dashboard panels, see Add panels to dashboards.
- For more details on panel searches, see Searches power dashboards and forms.
- See Edit dashboards and Edit visualizations to learn more about editing panel visualizations.
- To learn about working with reports, see Create and edit reports and Schedule reports in the *Reporting Manual*.

**Add panels to dashboards**
Learn how to add and edit dashboard panels.

To learn about types of dashboard panels, see Working with dashboard panels.

**Add panels using the Dashboard Editor**
Add panels to a dashboard with the dashboard's **Edit** menu. Access the **Edit** menu directly from the dashboard or from the list of dashboards on the Dashboards page.

1. Select **Edit** to open the dashboard editor.
2. Select **Add Panel**.
3. Expand one of the panel categories.
   - New
   - New from Report
   - Clone from Dashboard
   - Add Prebuilt Panel

4. (Optional) To search for specific panels, enter text in the **Filter** text box.
5. Select a panel and preview the selection.
6. Click **Add to Dashboard**.

**Filter the search for available panels**

Use filters in the search field to locate or create specific panels. The search looks for specified terms in existing dashboards, panels, and reports. It provides results for new panels using the specified search terms, and links to existing dashboards and panels containing the terms.

The following tips can help you with searching and filtering.

- Panel title or panel ID are useful items to search for.

- Use visualization element names, input types, chart types, and other keywords to filter a search. For example:

  - Search for **map** to return results to dashboards implementing a map visualization or to create a new panel with the map visualization.
  - Search for **multiselect** for results with a multiselect form input.

- You can filter for multiple items, but all items must appear in the order that you specify in the search field.

**Rearrange panels on a dashboard**

Drag and drop panels to rearrange their position on a dashboard.

1. If you are not in edit mode for the dashboard, select **Edit**.
2. Select a panel and drag it to its new position.

**Create an inline panel for a dashboard**

When you create an inline panel, you select a visualization and specify a search for the panel.
1. Select **Edit** to open the dashboard editor.
2. Select **Add Panel**.
3. Expand the panel category **New** and select a visualization for the data.
4. (Optional) Enter a title for the panel.
5. Enter a search string that returns the data to display in the panel.
6. (Optional) Select **Run Search** to preview the search results.
7. Select a time range for the search.
8. Click **Add to Dashboard**.

**Create a panel from a report**

When you create a panel from a report, you select from a list of available reports.

1. Select **Edit** to open the dashboard editor.
2. Select **Add Panel**.
3. Expand the panel category **New from Report** to view available reports.
   (Optional) Use the **Filter** option to search for specific reports. See Filter the search for available panels.
4. Select a report to view a preview of the report.
5. Click **Add to Dashboard**.

**Clone a panel from another dashboard**

You can clone a panel from another dashboard. The panel appears on your dashboard with the same editing capabilities as the cloned panel.

1. Select **Edit** to open the dashboard editor.
2. Select **Add Panel**.
3. Expand the panel category **Clone from Dashboard** to view available reports.
   (Optional) Use the **Filter** option to search for specific panels. See Filter the search for available panels.
4. Select and expand a dashboard. Select a panel to view a preview of the panel.
5. Click **Add to Dashboard**.

**Create and add a panel by reference**

You can create a panel that you can later add to dashboards by reference. This prebuilt panel is useful if you plan to reuse it often in various dashboards.

There are two ways to create a panel that you can reference from other dashboards.
• Convert an existing panel to a prebuilt panel that you can reference.
• Create a panel in simple XML code from the Settings page.

Typically, you create the panel using the dashboard editor, and then convert it to a prebuilt panel. You can also create the panel in simple XML code.

**Convert an existing panel to a prebuilt panel**

You can convert a panel to a prebuilt panel only if the panel does not contain a post-process search. A post-process search is a search that uses the `base` attribute to reference another search.

1. In the dashboard containing the panel that you want to convert, select **Edit > Edit Panels**.
2. From the **Options Menu** for a panel, select **Convert to Prebuilt Panel**.
3. (Optional) In the dialog that opens, specify the following details.

   - **ID**: The filename for the panel. Only alphanumeric characters, '-' character, and '_' are allowed.
   - **Panel Permissions**: Select either **Private** or **Shared in App**.
     - **Private**: Only you have permissions to view and edit the panel.
     - **Shared in App**: The panel is available to view and edit by other users of the app.

**Create a panel in Simple XML**

If this is your first time working with Simple XML, see Editing simple XML. See also the Simple XML Reference for more information on panel configurations.

1. From Splunk Web, go to **Settings > User Interface > Prebuilt Panels**.
2. In the Panels page, select **New** to open the Simple XML Editor.
3. In the Simple XML Editor, specify the following:

   - **Destination app**: Select an app for the context of the panel.
   - **Prebuilt Panel ID**: Enter a name for panel.
     - The name you enter is the filename on disk. Only alphanumeric characters, '-' character, and '_' are allowed.
   - **Prebuilt Panel XML**: Simple XML code to define a panel element.
     - The simple XML code for a reference panel contains only the `<panel>` element and its child elements.
**Add a prebuilt panel to a dashboard**

1. From the dashboard, select **Edit > Edit Panels**.
2. Select **Add Panel**.
3. Expand the panel category **Add Prebuilt Panel** to view the reference panels available. 
   (Optional) Use the **Filter** option to search for specific panels. See **Filter the search for available panels**.
4. Select a reference panel to view a preview of the panel.
5. Click **Add to Dashboard**.

**Convert a prebuilt panel to an inline panel**

You can convert a prebuilt panel to an inline panel. The prebuilt panel cannot contain a post-process search. A post-process search is a search that uses the `base` attribute to reference another search.

The conversion of a prebuilt panel to an inline panel lets you customize the search and visualization.

1. From a dashboard, select **Edit > Edit Panels**.
2. From the prebuilt panel you want to convert, click the **Options Menu** and select **Convert to Inline Panel**.

**Edit a panel title**

Panels and visualizations have separate titles.

You can specify a title when creating a panel. You can also use the dashboard editor to change panel titles, with one exception. Prebuilt panel titles cannot be edited in the dashboard editor. See **Edit a prebuilt panel** for more information.

**Steps**

1. Locate the panel that you want to edit in a dashboard.
2. Click **Edit** to open the dashboard editor.
3. Use one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>What to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing user interface</td>
<td>1. Next to <strong>Edit Dashboard</strong> at the top left of the page, make sure that the UI editor is selected.</td>
</tr>
</tbody>
</table>
| Edit Simple XML | 2. Click the panel title that you want to edit and change the text.
| 1. Next to **Edit Dashboard** at the top left of the page, make sure that the **Source** editor is selected.
| 2. Locate the `<title>` element inside the `<panel>` that you want to edit.
| 3. Change the panel title text.
| 4. Click **Save**. |

### Edit a prebuilt panel

Use the **Prebuilt Panels** page to access a panel source code editor.

**Steps**

1. From the home page, navigate to **Settings > User Interface > Prebuilt Panels**.
2. Locate the panel that you want to edit and select **Edit**.
3. Edit the Simple XML source code.
4. Click **Save**. The panel is updated in dashboards that include it by reference.

### Delete a panel from a dashboard

You can delete a panel from a dashboard using the Dashboard Editor or by editing the simple XML code:

- From the Dashboard Editor, in panel-editing mode, click the **Options** menu for a panel and select **Delete**.
  Or you can click the **Delete icon, X**, which is in the upper right corner of the panel.

- In simple XML source code, delete the `<panel>` element and its contents.

### Edit dashboards

Use the dashboard editor to customize dashboard panels, layout, or add interactivity.
Open the dashboard editor

1. From the **Dashboards** listing page, open the dashboard that you want to convert.
2. Click **Edit** to open the dashboard editor.
3. Select **UI** or **Source** to change the editing mode.
4. (Optional) Preview dashboard edits as you make them and click **Save** to save changes. Click **Cancel** at any point to discard changes.

Change dashboard panel layout

You can change dashboard layout to prioritize specific panels or make room for additional content.

1. From the **Dashboards** listing page, open the dashboard that you want to convert.
2. Click **Edit** to open the dashboard editor.
3. Drag and drop panels to reposition them.

Edit a panel search

Update the search driving a particular dashboard panel.

When you are working with inline searches, the dashboard search bar has syntax highlighting and auto-complete features that can help you build a search string. To learn more, see Help reading searches in the *Search Manual*.

Search editing options

<table>
<thead>
<tr>
<th>All search types</th>
<th>Reports</th>
<th>Inline searches and inline pivots</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Edit the title. • Delete the search.</td>
<td>• View and edit the report in a new window. • Open the report search in a new window. • Clone to an inline search or pivot.</td>
<td>• Edit the search, specifying a new inline search or pivot. • Convert the inline search or pivot to a report.</td>
</tr>
<tr>
<td>Steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>• Select a different report for the panel.</td>
<td>• Select the visualization specified in the report for this panel.</td>
<td>• Specify an automatic refresh interval delay and indicator option.</td>
</tr>
<tr>
<td>Steps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. From the <strong>Dashboards</strong> listing page, open the dashboard that you want to edit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Click <strong>Edit</strong> to open the dashboard editor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>At the top right of each panel, editing icons appear. The first editing icon represents the search for the panel. The search icon varies to represent the type of search being used.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Select the search icon to view configuration options for the search.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Select the search configuration that you want to change. Depending on the option you select, additional configuration dialogs or windows might open.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. After editing the search, click <strong>Save</strong> to save changes to the dashboard.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Edit a panel visualization**

Use the dashboard editor to edit a panel visualization for panels that are not generated with pivot or pivot report searches.

If you are working with visualizations generated from pivot or pivot report searches, you can use the Pivot Editor. See Design pivot charts and visualizations with the Pivot Editor for details.

**Prerequisites**

- Review Data structure requirements for visualizations for details on generating search results in the correct format for a visualization.
- See Properties available from the Visualization Editor to review visualization configurations.
Steps

1. From the **Dashboards** listing page, open the dashboard that you want to edit.
2. Click **Edit** to open the editing dashboard.
   
   At the top right of each panel, editing icons appear. The second icon represents the Visualization Picker. The icon varies to represent the visualization type. The third editing icon represents the visualization Format menu.
3. (Optional) Use the Visualization Picker to select a different visualization.
   
   Make sure that the panel search generates results in the correct format for the new visualization. You can select any visualization, but the panel search results might not render if they are not formatted for the selected visualization.
4. (Optional) Use the Format menu to configure the visualization.
5. Click **Save** to save changes to the dashboard.

---

**Edit dashboard source code**

Edit dashboard Simple XML source code to customize settings that are not accessible from the user interface. The dashboard source code editor provides interactive validation as you make updates.

**Prerequisites**

- For information about editing Simple XML source code, see About editing Simple XML.

**Steps**

1. From the **Dashboards** listing page, open the dashboard that you want to edit.
2. Select **Edit** to open the dashboard editor.
3. Click **Source** to open the dashboard XML source code editor.
4. Edit the source code.
5. (Optional) Observe that the editor provides automatic tag closing and validation. The editor displays validation warning or error messages as needed. Hover over a warning or error icon next to a line of source code to view the message for that line.
6. (Optional) Validation warnings and errors disable the **Save** button. If the button is disabled, correct any code with validation warnings or errors.

7. If there are no warnings or errors, the **Save** button is enabled. Click **Save** to save the source code edits.

---

**Edit a prebuilt panel**

Prebuilt panels cannot be edited in the dashboard editor. Use the **Prebuilt Panels** page to access a panel source code editor.

**Steps**

1. From the home page, navigate to **Settings > User Interface > Prebuilt Panels**.
2. Locate the panel that you want to edit and select **Edit**.
3. Edit the Simple XML source code.
4. Click **Save**. The panel is updated in dashboards that include it by reference.

**Additional resources**

- To learn about creating or editing visualizations in a dashboard, see **Edit Visualizations**.
- For details on converting a dashboard to a form and working with forms, see **Create and edit forms**.

**Edit visualizations**

Edit a visualization to configure its search, type, appearance, and behavior.

**Visualization component editing**

You can edit visualizations in the dashboard editor or on the **Search** page. In either location, you can adjust the following visualization components.
<table>
<thead>
<tr>
<th>Visualization components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search string</td>
<td>Use the dashboard search editor or the search bar to change the query driving the visualization.</td>
</tr>
<tr>
<td>Type</td>
<td>Use the <strong>Visualization Picker</strong> to select a visualization type. Ensure that the query generates results in the proper structure for the selected visualization.</td>
</tr>
<tr>
<td>Format and behavior</td>
<td>Use the <strong>Format</strong> menu to adjust appearance, drilldown, and other settings for the visualization's user interface.</td>
</tr>
</tbody>
</table>

**Caution:** Changing visualization settings in the dashboard editor can overwrite related token settings and behavior. If you are using tokens to configure part of a dashboard or form, use caution when updating related elements in the dashboard editor. For example, if a form input configures chart legend placement, selecting a legend placement in the Format menu overwrites the dynamic token setting from the input. In this case, the input remains in the dashboard but no longer configures legend placement.

**Visualization editing workflow**

The workflow for editing a visualization search, type, or format is slightly different depending on whether you are editing in the dashboard editor or the Search page.

**Dashboard editing permissions**

Write permission is required for editing dashboard panels. By default, you have write permission for any dashboard that you create. However, you might have read-only access to other dashboards. Users with the `admin` role can change editing permissions.

**Edit visualizations in the dashboard editor**

1. In the Search and Reporting app, select the Dashboards tab.
2. Locate the dashboard to edit. Use one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Additional steps for this option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select <strong>Edit</strong>.</td>
<td>None</td>
</tr>
</tbody>
</table>

97
Click on the dashboard name to view it. After the dashboard opens, select **Edit**.

3. In the panel you are editing, locate the icons for editing the search, visualization type, and format. Select the icon for the component you are editing.

4. Edit the selected visualization component.

**Edit visualizations on the Search page**

1. In the Search and Reporting app, select the Search tab.
2. Enter a query.
3. When results are available, select the Visualization tab.
4. To edit the visualization, use one of the following tools.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization</td>
<td>Change the visualization type.</td>
</tr>
<tr>
<td>Picker</td>
<td></td>
</tr>
<tr>
<td>Format menu</td>
<td>Change the visualization format and behavior.</td>
</tr>
<tr>
<td></td>
<td>Format options vary by visualization type.</td>
</tr>
<tr>
<td>Search bar</td>
<td>Edit the query and rerun it to refresh the visualization.</td>
</tr>
</tbody>
</table>

**Using the Format menu**

Format menu configurations are applied immediately to visualizations.

- Each edit that you make is saved to the visualization. You can see each change in the visualization and make adjustments as you go.
- Edits are reflected in the dashboard Simple XML source code as they are made.
- Click and drag the Format menu to move it anywhere on the screen.
- Close the Format menu or click anywhere outside of it to exit and save changes.

**Additional resources**

Formatting and other options vary by visualization type. To compare visualizations, see the Visualization reference. See also Chart Configuration Reference.
For information on using Pivot to edit visualizations, see Design pivot charts and visualizations with the Pivot Editor.

Create and edit forms

Let users select or filter content by converting a dashboard to a form. A form is a dashboard that includes one or more inputs, such as radio buttons or a checkbox.

Workflow for creating forms

The typical workflow for creating forms includes the following steps. Some steps are optional and you do not have to complete them in this order.

- Create a dashboard and add one or more inputs to convert it to form.
- Configure the inputs by specifying options available to users, default behavior, and how to handle user selections.
- Work with tokens that capture selected values from the input. Modify searches and other content to use the token values.
- Make any additional input configurations depending on the input type.
- Adjust input and panel layout.

Add inputs to convert a dashboard to a form

Forms are dashboards with interactive user inputs for selecting or filtering content. When you add an input to a dashboard, the top-level source code element changes from `<dashboard>` to `<form>`.

Steps

1. From the Dashboards listing page, open the dashboard that you want to convert.
2. Click Edit to open the dashboard editor.
3. Select one or more inputs from the Add input list. As soon as you select an input, the dashboard is converted to a form.
4. (Optional) Drag and drop inputs to rearrange them.
5. (Optional) Drag an input into a specific panel. Use tokens to make the input control only this panel.
Tokens in form inputs

Use tokens to respond to user selections dynamically.

When you add an input to a form, a unique token is generated for the input. You can use this token to change panel content based on what users select. For example, use a token in the panel search to modify the results that the panel visualization shows. Or, change the panel label or drilldown behavior using the token.

The following examples show you typical options for using tokens in a form input.

Reference a token in a search

In this example, a dashboard has one panel. The panel search aggregates events for all sourcetypes.

```
index=_internal | timechart count by sourcetype
```

Add interactivity by converting the dashboard to a form. Add a text input to let users specify a sourcetype to filter the events in the visualization.

1. From the dashboard, click **Edit** to open the dashboard editor.
2. Select **Add input > Text** to add a text input. The dashboard is converted to a form.
3. Observe that the input appears at the top of the form. A label and token are generated for the input.
4. Click the editing icon to open the input editor.
5. Change the label text to "Sourcetype".
6. Change the token name to `sourcetype_token` to make it more specific. Token names should be unique within the form.
7. Update the panel search string to use the value that the token captures from user input. This search uses the specified sourcetype to filter events.
   ```
   index=_internal sourcetype=$sourcetype_token$ | timechart count
   ```
   Observe the `$[token name]` syntax used to refer to the token in the search.
8. Click **Apply** and **Save** to save the updates to the form.

The form now contains a text input for users to specify a sourcetype to show in the visualization. Before users specify a sourcetype, the visualization does not
render because the token has not captured a value to use in the search. After a sourcetype is entered in the text field, the search uses this value to generate results and the visualization renders.

**Add a time input to a form**

A time input lets users apply a time range to filter the events shown in one or more panels.

Add the time input and update panel searches to incorporate the user specified time range.

1. From the dashboard or form, click **Edit** to open the dashboard editor.
2. Select **Add Input > Time**.
3. (Optional) Select the input editing icon and update the input label and token name.
4. (Optional) Click **Apply** to save the input updates.
5. For each panel where you want the time input to apply, make the following changes.
   1. Click the search editing icon.
   2. Select **Edit search**.
   3. For **Time Range Scope** select **Shared Time Picker**. If there is more than one time input in the form, each shared time picker is listed with the unique token for its time input. Select the shared time picker that includes the token for the input that you want to use.
6. Click **Save** to save the panel search updates.
7. Click **Save** to save the form updates and exit the dashboard editor.

**Apply time inputs to panels**

A form can have one or more time inputs. Depending on the behavior you want, a time input can be used globally for all panels or only for specific panels.

**Global time picker**

When you create a time input using the dashboard editor, a token is generated automatically for the input. If you delete this token, the time input becomes global. Each panel in the form that does not specify a time range or another time input uses the global time picker.

When configuring a panel search time range, you can select **Shared Time Picker (global)** to apply this input.
**Token-based time picker**
When configuring a panel search time range, you can select a shared time picker that includes a specific token name. Use the token name for the input that you want to apply to the panel search.

**Panel searches that do not use time inputs**
You can specify an explicit time range for a panel search or use other token values if you do not want to apply a time input to a particular panel.

---

**Configure input value handling**

You can configure how input values populate a form.

**Submit token values when the page loads**

To submit token values when the page loads, enable autorun behavior.

1. From the dashboard, click **Edit** to open the dashboard editor.
2. Select the **Autorun dashboard** check box.
3. Click **Save** to save the form update.

**Submit token values when an input changes**

By default, inputs are configured to submit token values whenever users make a new selection. To change this behavior, follow these steps.

1. From the dashboard, click **Edit** to open the dashboard editor.
2. Select the input to edit.
3. Clear the **Search on change** check box to disable this behavior, or select the check box to enable it.
4. Save input and form changes.

**Add a Submit button input**

Add a Submit button to a form to let users control when input selections are submitted. This can be helpful for managing how often panels or forms with multiple inputs update. Typically, **Search on change** is disabled for inputs if you use a "Submit" button in the form.

You cannot change the position of the form Submit button.

1. From the dashboard, click **Edit** to open the dashboard editor.
2. Select **Add Input > Submit**.
3. Disable **Search on change** behavior for inputs as needed and save input updates.
4. Click **Save** to save form updates.

**Specify initial and default input values**

Handle cases where user input values are not available.

**Default**

Use a default value for an input when users do not make a selection.

**Initial**

Use an initial value for text inputs only. The initial value appears only when the form page loads. If a user clears the text field, the initial value does not reappear and the token value is set to an empty string.

If you specify both an initial and default value for a text input, only the default value applies.

---

**Specify multiple options for inputs**

Several form input types can include multiple static or dynamically populated options.

- check box
- dropdown
- link list
- radio
- multiselect

All of these inputs display multiple options, while multiselect and check box inputs let users choose multiple values. The following tasks show you how to configure options for each of these inputs.

**Specify static options**

The following example shows you how to specify multiple static options. The example uses a dropdown input but it applies to any multi-option input.

1. From a dashboard, select **Add Input > Dropdown**.
2. Select the edit icon for the input. Select **Static Options**.
3. Specify a name and value for the first option.
4. For each additional option, Click **Add Option** and specify the name and value for the option.
5. (Optional) Scroll to the **Default** field and specify a default value.
6. (Optional) Drag and drop the options to rearrange them.
7. Click **Apply** to save input changes.
8. Click **Save** to save dashboard changes.

**Specify dynamic options**

Use a search to generate option labels and values dynamically.

These steps show you how to configure a dynamically populated dropdown. They do not include steps for updating the search to use the token value from the input.

1. From the dashboard or form, add a line chart panel that uses the following search.
   ```plaintext
   index=_internal | timechart count
   ```
2. Add an input to let users filter the panel visualization for a particular sourcetype.
   - Click **Edit** to open the dashboard editor.
   - Select **Add Input > Dropdown**.
3. Configure the input:
   - Select the input editing icon.
   - Select **Dynamic options**.
   - Add the following search to generate input labels and values.
     ```plaintext
     index=_internal | stats count by sourcetype | eval label=sourcetype." (".count.")"
     ```
   - Observe that the search aggregates events by sourcetype and generates a label field that combines sourcetype names and event counts.
   - Use the search result fields for input labels and values. Specify the following fields. **Field for Label**: label **Field for Value**: sourcetype
4. Click **Apply** to save input updates.
5. Click **Save** to save form changes.
Users can now view sourcetype names and event counts in the dropdown.

**Handle multiple value selections**

Multiselect and check box form inputs let users select multiple values.

This example panel includes a check box for users to specify sourcetypes to render in the chart.

**Search to generate multiple selected values**

To handle one or more user selected values in a multiselect or check box, use a search that generates results for one or more values.

To specify the source type values in the above form, build a search string indicating the values to return. For this example, the following search string allows the selection of multiple values for source types:

```
(sourcetype="splunkd" OR sourcetype="splunk_web_access" OR sourcetype="splunkd_access")
```

The search driving the panel accesses the token value of check box and multiselect differently than the other form inputs. Use the submitted modifier to the token.

```
index=_internal $src_type_tok$ | chart count by sourcetype
```

The Input Editor provides editing fields to specify multiple values for selection in a check box or multiselect. The table below describes these fields and provides example values that build the following search string:
<table>
<thead>
<tr>
<th>Editor Field</th>
<th>Description</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token Prefix</td>
<td>String prefixed to the value of the input element.</td>
<td>()</td>
</tr>
<tr>
<td></td>
<td>For multiple selections, this is typically an open parenthesis to enclose the string selecting the values.</td>
<td></td>
</tr>
<tr>
<td>Token Suffix</td>
<td>String appended to the value of the input element.</td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>For multiple selections, this is typically a close parenthesis to enclose the string selecting the values.</td>
<td></td>
</tr>
<tr>
<td>Token Value Prefix</td>
<td>String prefixed to the value of the input element. Can be a regular expression. Default value is an opening double quote (&quot;).</td>
<td>sourcetype=&quot;</td>
</tr>
<tr>
<td></td>
<td>Typically, this is the opening part of a sub-string that selects the multiple values.</td>
<td></td>
</tr>
<tr>
<td>Token Value Suffix</td>
<td>String appended to the value of the input element. Can be a regular expression. Default value is an closing double quote (&quot;).</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td>Typically, this is the closing part of a sub-string that selects the multiple values.</td>
<td></td>
</tr>
<tr>
<td>Delimiter</td>
<td>A string placed between each selected value. Typically, you specify &quot; OR &quot; or &quot; AND &quot; using upper case. Do not specify the quote marks, but specify a space in the input.</td>
<td>OR</td>
</tr>
</tbody>
</table>
The following procedure shows how to enable multiple selections for a check box or multiselect input.

1. From the dashboard, click Edit to open the dashboard editor.
2. Select Add Input. Select either Checkbox or Multiselect.
4. Specify choices as described in Specify choices with static options and Specify choices with dynamic options.
5. Build the multiselect search string using the editing fields in the above table. 
   (Recommended) Use the preview feature to verify the multiselect search string.
6. (Optional) Specify a default value.
7. Click Apply. Click Done.

Form input examples

This section provides an example of each form input, with a list of the key fields for implementing the example.

Check box

This example uses the check box input to indicate which source types to display in a timechart. A populating search specifies the available options to select. Three source types are selected by default:

  splunk_web_access
  splunk_web_service
  splunkd

This example enables Search on Change. The form loads when a selection is made.
The panel displays results in the default column chart, using the following base search. The visualization references the input values using the value specified for Token. In this example, the token name is src_type_tok.

index=_internal $src_type_tok$ | timechart count by sourcetype

General settings

Specify the Label for the input and the Search on Change behavior. This example enables search on change.

Token options

Use the Token Options to specify the value returned by the check box input.

For the Token field, specify a name for the token that returns the value. The base search for the visualization references this token. In this example, specify src_type_tok.

Use the following fields to build the search for the returned value. The Preview field in the Input Editor updates as you edit these fields.

- Token Prefix
- Token Suffix
- Token Value Prefix
- Token Value Suffix
- Delimiter

The example values listed in the table below build the following search string:

(sourcetype="splunkd" OR sourcetype="splunk_web_access" OR ...)

After you dynamically create the check boxes, from the Default field, select the check boxes that are enabled by default.
Static options

Use the static options to explicitly define the Name and Value of the check boxes for the input.

This example leaves the static options blank. It uses a populating search to define the check boxes for the input.

Dynamic Options

Reference a report or define an inline populating search to define the check boxes for the input.

This example uses the following inline search:

```
| metadata type=sourcetypes index=_internal
```

The example runs the search against all time.

Use field names to specify a name/value pair for the check boxes. This example specifies the sourcetype field for both Field for Label and Field for Value.

Example values for check box input

This table lists the example values for the check box input example.

<table>
<thead>
<tr>
<th>Editor Field</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Source Types (Check Box)</td>
</tr>
<tr>
<td>Search on Change</td>
<td><em>Enabled</em></td>
</tr>
<tr>
<td><strong>Token Options</strong></td>
<td></td>
</tr>
<tr>
<td>Token</td>
<td>src_type_tok</td>
</tr>
<tr>
<td>Default</td>
<td>splunk_web_access</td>
</tr>
<tr>
<td></td>
<td>splunk_web_service</td>
</tr>
<tr>
<td></td>
<td>splunkd</td>
</tr>
<tr>
<td>Token Prefix*</td>
<td>(</td>
</tr>
<tr>
<td>Token Suffix*</td>
<td>)</td>
</tr>
</tbody>
</table>
Token Value Prefix* | sourcetype="

Token Value Suffix* | "

Delimiter* | OR

Dynamic Options

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Inline Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search String</td>
<td>| metadata type=sourcetypes index=_internal</td>
</tr>
<tr>
<td>Time Range</td>
<td>All time</td>
</tr>
<tr>
<td>Field for Label</td>
<td>sourcetype</td>
</tr>
<tr>
<td>Field for Value</td>
<td>sourcetype</td>
</tr>
</tbody>
</table>

*These fields build the search string that dynamically create the checkboxes. For the Delimiter field, be sure to specify an opening and closing space.

**Dropdown input**

This example uses a dropdown input to indicate which source types to display as a time chart. The panel displays results as a bar chart, using the following base search.

```
index=_internal sourcetype=$src_type_tok$ | timechart count by sourcetype
```

The token $src_type_tok$ references the values specified by the dropdown.

The example uses static options to define choices for the dropdown.

The example specifies splunk for Token Prefix. Each selected value prefixes the token prefix to the value.

There is a default value for the dropdown.
The example relies on a Submit button to run the search. Changes to the selection do not apply until you click the Submit button.

<table>
<thead>
<tr>
<th>Editor Field</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Source Types (Dropdown)</td>
</tr>
<tr>
<td>Search on Change</td>
<td>Not specified</td>
</tr>
<tr>
<td><strong>Token Options</strong></td>
<td></td>
</tr>
<tr>
<td>Token</td>
<td>src_type_tok</td>
</tr>
<tr>
<td>Default</td>
<td>Daemon</td>
</tr>
<tr>
<td>Token Prefix</td>
<td>splunk</td>
</tr>
<tr>
<td><strong>Static Options</strong></td>
<td></td>
</tr>
<tr>
<td>Name : Value</td>
<td>Daemon : d</td>
</tr>
<tr>
<td>Name : Value</td>
<td>Web Service : _web_service</td>
</tr>
<tr>
<td>Name : Value</td>
<td>Web Access : _web_access</td>
</tr>
<tr>
<td>Name : Value</td>
<td>Daemon Access : d_access</td>
</tr>
</tbody>
</table>

*Multiselect*

This example uses a multiselect input to indicate which source types to display in a timechart. The panel displays results in the default column chart, using the following base search.

```
index=_internal $src_type_tok$ | timechart count by sourcetype
```

The example uses static options to define choices for the dropdown.

Two source types are selected by default:

- Daemon
- Web Access
This example enables **Search on Change**. The form loads when a selection is made.

For a multiselect input, you define multiple values to select by building the following search string.

**(sourcetype="splunkd" OR sourcetype="splunk_web_access" OR ...)**

The token $src_type_tok$ references this search string in the search that drives the panel contents. The fields that build the search string are indicated in the table below.

<table>
<thead>
<tr>
<th><strong>Editor Field</strong></th>
<th><strong>Example Values</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>Source Types (Multiselect)</td>
</tr>
<tr>
<td><strong>Search on Change</strong></td>
<td>Enabled</td>
</tr>
<tr>
<td><strong>Token Options</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Token</strong></td>
<td>src_type_tok</td>
</tr>
</tbody>
</table>
| **Default**            | Daemon
                  | Web Access                                            |
| **Token Prefix**       | (                                                      |
| **Token Suffix**       | )                                                     |
| **Token Value Prefix** | sourcetype="                                          |
| **Token Value Suffix** | "                                                   |
| **Delimiter**          | OR                                                     |
| **Static Options**     |                                                        |
| **Name : Value**       | Daemon : splunkd                                       |
| **Name : Value**       | Web Service : splunk_web_service                       |
| **Name : Value**       | Web Access : splunk_web_access                         |
| **Name : Value**       | Daemon Access : splunkd_access                         |
| **Name : Value**       | Version : splunk_version                              |
| **Name : Value**       | Error : splunkd_stderr                                 |

*These fields build the search string that supplies the token value. For the Delimiter field, be sure to specify an opening and closing space.*
**Radio input**

This example uses a radio input to indicate which source types to display as a time chart. The panel displays results as an area chart, using the following base search.

```
index=_internal sourcetype=$src_type_tok$ | timechart count by sourcetype
```

The token `$src_type_tok$` references the values specified by the dropdown.

The example uses static options to define choices for the dropdown.

There is a default value for the radio input.

This example enables **Search on Change**. The form loads when a selection is made.

<table>
<thead>
<tr>
<th>Editor Field</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>Source Types (Radio)</td>
</tr>
<tr>
<td><strong>Search on Change</strong></td>
<td><strong>Enabled</strong></td>
</tr>
<tr>
<td><strong>Token Options</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Token</strong></td>
<td>src_type_tok</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>Web Service</td>
</tr>
<tr>
<td><strong>Static Options</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Name : Value</strong></td>
<td>Daemon : splunkd</td>
</tr>
<tr>
<td><strong>Name : Value</strong></td>
<td>Web Service : splunk_web_service</td>
</tr>
<tr>
<td><strong>Name : Value</strong></td>
<td>Web Access : splunk_web_access</td>
</tr>
<tr>
<td><strong>Name : Value</strong></td>
<td>Daemon Access : splunkd_access</td>
</tr>
</tbody>
</table>
**Text input**

This example uses a text input to indicate which source types to display as a time chart. The panel displays results as a pie graph, using the following base search.

```
index=_internal sourcetype=$src_type_tok$ | timechart count by sourcetype
```

The token `$src_type_tok$` references the values specified in the text input.

This example specifies an initial value of `splunkd*` without specifying a default value. Upon initial load, the seed value is applied. The form reloads when you specify a new value.

Because there is no default value, an empty text input does not return any results.

<table>
<thead>
<tr>
<th>Editor Field</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Source Types (Text input)</td>
</tr>
<tr>
<td>Search on Change</td>
<td>Enabled</td>
</tr>
<tr>
<td>Token Options</td>
<td></td>
</tr>
<tr>
<td>Token</td>
<td>src_type_tok</td>
</tr>
<tr>
<td>Default</td>
<td>Not specified</td>
</tr>
<tr>
<td>Initial</td>
<td>splunkd*</td>
</tr>
</tbody>
</table>

**Time input**

This example shows how to use a time input to specify time ranges for a panel in a form. The form contains a radio input to indicate which source types to display as a time chart. The panel displays results as a column chart, using the following base search.
The examples specifies `time_input_tok` to reference the time input in a panel.

In the Panel Editor, select **Edit Search String**. From the **Time Range Scope** dropdown, select **Shared Time Picker (time_input_tok)**.

The default value for the time input is **Last 4 days**.

The example enables **Search on Change** for the time input. The form loads when a new time range is selected.

<table>
<thead>
<tr>
<th>Editor Field</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Time Input</td>
</tr>
<tr>
<td>Search on Change</td>
<td><strong>Enabled</strong></td>
</tr>
<tr>
<td><strong>Token Options</strong></td>
<td></td>
</tr>
<tr>
<td>Token</td>
<td><code>time_input_tok</code></td>
</tr>
<tr>
<td>Default</td>
<td>Last 4 days</td>
</tr>
</tbody>
</table>

**Convert a dashboard to HTML**

Splunk Enterprise users can implement additional behavior and appearance customizations by converting a dashboard to HTML.

Learn about creating and editing HTML dashboards on the Splunk developer portal.

- See Convert Simple XML dashboards to HTML for information and procedures.
- See About file precedence and caching to learn about caching and refreshing Splunk assets, including HTML files.
Limitations to HTML dashboards

There are some limitations on dashboards converted to HTML.

- HTML dashboards cannot be exported to PDF.
- The Splunk Web dashboard editor cannot be used to edit HTML source code for converted dashboards.
Create Dashboards with Simple XML

Editing Simple XML

You can use interactive editors to create and edit dashboards without having to edit Simple XML source code. However, some advanced dashboard features are not available from interactive editors. You can access these features by editing the underlying simple XML code.

Edit dashboard source code

Edit dashboard Simple XML source code to customize settings that are not accessible from the user interface. The dashboard source code editor provides interactive validation as you make updates.

Prerequisites
If you are unfamiliar with Simple XML, review the following information before you edit source code.

- Dashboard and form structure and elements in Dashboards and forms
- Available options and elements in the Simple XML Reference

Steps

1. From the Dashboards listing page, open the dashboard that you want to edit.
2. Select Edit to open the dashboard editor.
3. Click Source to open the source code editor.
4. Edit the source code.
   The editor provides automatic tag closing and validation. It also displays warnings or error messages as needed. Hover over a warning or error icon next to a line of source code to view details.
5. If the Save button is disabled, correct any code with validation warnings or errors. Otherwise, click Save to save your edits.

Special characters in XML files

Some characters have special meaning in Simple XML files. To prevent the source code parser from treating them as special characters, wrap them in 
<![CDATA[[]]> tags.
You can also escape these characters using HTML entities.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
<th>HTML Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td>apostrophe</td>
<td>'</td>
</tr>
<tr>
<td>&quot;</td>
<td>quote</td>
<td>&quot;</td>
</tr>
<tr>
<td>&lt;</td>
<td>left angle bracket</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>right angle bracket</td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>ampersand</td>
<td>&amp;</td>
</tr>
</tbody>
</table>

**Read-only access to dashboard Simple XML code**

Access a read-only version of dashboard source code by appending the `showsource` query parameter to the dashboard URL. See the following example.


**Note:** Read-only source code access is available only for Simple XML dashboards. Read-only HTML or Advanced XML source code is not accessible using the URL.

**Additional information**

Before you edit Simple XML, review the following resources.

- Simple XML dashboard and form structure in Anatomy of dashboards and forms.

- The Simple XML Reference and the Chart Configuration Reference provide details on Simple XML elements and options.

Splunk Enterprise users can edit Simple XML using a third-party editor. This option is not available in Splunk Cloud. See Using a third party XML editor.
Searches power dashboards and forms

Searches generate visualizations and other content in dashboards and forms. You can also use searches to implement dynamic or interactive behavior in a dashboard.

Learn about the available search types and how to work with them in Simple XML.

Overview of search types and uses

You can use one or more search types in a dashboard depending on the content or behavior that you are creating.

Inline searches

Inline searches exist directly in a dashboard or visualization. You can create or modify inline searches when you edit dashboard panels.

Each visualization can have its own inline search. You can also use an inline search as a base search in a dashboard. Base searches require post-process searches to modify results and generate visualizations in dashboard panels.

Referenced report searches

Use a search saved as a report to generate dashboard content. Use a reference to the report to add its search and visualization to a dashboard or form.

You can edit the dashboard to adjust the visualization and time range from a referenced report. However, you cannot edit the search directly in the dashboard. Edit the report to change the search.

When you change a report search, referenced report searches in dashboards update automatically.

Searches to populate form inputs

You can use a search to generate form input choices dynamically. Indicate a results field to use for the choice labels and values that users see in the form.
**Post-process searches**

Post-process searches perform additional processing on results from a base search. You can use multiple post-process searches to generate different results from the same base search. You can also populate form inputs with a post-process search.

**Prebuilt panels**

Create a prebuilt panel if you want to reuse a search and other content. To change a prebuilt panel search, edit the prebuilt panel directly.

Changes to a prebuilt panel appear automatically in dashboards where the panel is used.

**Searches generated with Pivot**

You can generate pivots for export to dashboards. For more information, see Design pivot tables with the Pivot Editor in the *Pivot Manual*.

**Create searches in Simple XML**

The `<search>` element defines a search in Simple XML source code. Search elements include child elements, such as `<query>` for the search string and elements for the time range.

You can use a `<search>` element to define searches generating dashboard or form content. You can also use a `<search>` to generate form input choices or define post-process searches.

See the Simple XML Reference for more details on the the `<search>` element, its child elements, and usage requirements.

**Examples**

**Inline search**

Use an inline search to generate data for a visualization.

The search in this example generates data for a single value visualization showing system error counts in the last week.
In the source code for this visualization, the <search> element includes these child elements.

- <query> contains the search string.
- <earliest> and <latest> define the search time range.

**Simple XML source code**

```xml
<panel>
  <single>
    <title>Errors in the last week: single value visualization</title>
    <search>
      <!-- Search string -->
      <query>index=_internal source="*splunkd.log"
        ( log_level=ERROR OR log_level=WARN*
        OR log_level=FATAL OR log_level=CRTICAL )
        | stats count as log_events</query>
      <!-- search time range -->
      <earliest>-7d@h</earliest>
      <latest>now</latest>
    </search>
    <option name="underLabel">Errors in the last week</option>
    <option name="useColors">1</option>
  </single>
</panel>
```

**Referenced search from a report**

Use and modify a referenced saved search to generate dashboard panel content.

Referenced a saved search from a report. You can use the original time range and visualization from the report or you can modify them in Simple XML. To change the search string, edit the report.
To use a saved search in dashboard Simple XML, start by adding a `ref` attribute to the `<search>` element. The reference indicates the name of the report to add to the dashboard panel.

**Examples**
The following examples show you how to use and modify a referenced saved search in a dashboard panel.

- **Reference a saved search**
  This dashboard panel source code includes a saved search from a report. The search time range and visualization defined in the report are not modified.

```xml
<panel>
  <title>Referenced saved search</title>
  <chart>
    <title>Sales by product code</title>
    <search ref="saved_search_simple_example">
    </search>
  </chart>
</panel>
```

- **Modify the search time range**
  This example changes the search time range using the `<earliest>` and `<latest>` modifiers. The panel search uses the custom time range.

```xml
<panel>
  <title>Referenced saved search with custom time range</title>
  <chart>
    <title>Sales by product code in the last week</title>
    <search ref="saved_search_simple_example">
      <earliest>-7d@h</earliest>
      <latest>now</latest>
    </search>
  </chart>
</panel>
```

- **Modify the visualization type**
  This example uses a different visualization for the search results.

```xml
<panel>
  <title>Referenced saved search with custom time range and visualization</title>
  <chart>
    <title>Sales by product code in the last week</title>
    <search ref="saved_search_simple_example">
      <earliest>-7d@h</earliest>
      <latest>now</latest>
    </search>
  </chart>
</panel>
```
Search populating a form input

Generate form input choices with a search.

You can use a search to populate choices dynamically for the following form inputs.

- Checkbox
- Dropdown
- Multiselect
- Radio buttons

Do not use a real-time search to populate form input choices. Input choices do not update as search results change.

Example
A form lets users select a product category to see the current sales total for that category.

In this Simple XML source code, the input includes a `<search>` element for the search that generates input choices. The `fieldForLabel` and `fieldForValue` elements indicate the results field to use for choice labels and values.

A static `<initialValue>` indicates an initial choice value to use before users make a selection.
When users choose a category, the input captures the selected value in the **category** token. The single value visualization in this form uses the token to update dynamically. The panel search includes the token value to generate results for the selected category. The token value also appears in the panel title.

```html
<form>
  <label>Search populated form input</label>
  <fieldset autoRun="true" submitButton="false">
    <input type="radio" token="category" searchWhenChanged="true">
    <label>Select a Product category</label>
    <default>ACCESSORIES</default>
    <search>
      <query>source="tutorial*" NOT null | stats count by categoryId</query>
      <earliest>−7d@h</earliest>
      <latest>now</latest>
    </search>
    <fieldForLabel>categoryId</fieldForLabel>
    <fieldForValue>categoryId</fieldForValue>
    <initialValue>ACCESSORIES</initialValue>
  </input>
</fieldset>
</form>

**Using tokens to add dynamic values to searches**

Tokens are references to dynamic data, such as search result counts or user specified values. Use tokens within a search to access dynamic values and generate more customized results.

To use a token in a search, use $...$ delimiter characters around the token name. When the search runs, the $<token_name>$ is replaced with the dynamic
value that it references.

Example
This search string includes the $series_tok$ token to represent a dynamic series field value. This value might come from a user selection in a form input.

```
index=_internal source=*metrics.log group="per_sourcetype_thruput"
series=$series_tok$ | table sourcetype eps, kb, kbps
```

The search can generate a visualization representing the user selected $series_tok$ value.

In addition to searches, tokens can also be used to capture user input values and control dynamic dashboard behavior. See Token usage in dashboards for more details.

Post-process searches

Post-process searches perform additional processing on results from a base search. A base search can be a global search or any other search within a dashboard. Use the base attribute in a post-process <search> to indicate the base search id.

You can use a single post-process search to generate results or you can chain multiple post-process searches together.

Best practices

Use these best practices to make sure that post-process searches work as expected.

Use a transforming base search

A base search should be a transforming search that returns results formatted as a statistics table.

Non-transforming base search issues

Non-transforming base searches can cause the following search result and timeout issues. If you observe these issues in a dashboard, check the base search to make sure that it is a transforming search.

No results returned

If the base search is a non-transforming search, you must explicitly state in the
base search what fields will be used in the post-process search using the \| \fields \ command. For example, if your post-process search will search for the top selling buttercup game categories over time, you would use a search command similar to the following.

\| \fields \ _time, \ categoryId, \ action

**Event retention**

If the base search is a non-transforming search, the Splunk platform retains only the first 500,000 events that it returns. A post-process search does not process events in excess of this 500,000 event limit, silently ignoring them. This can generate incomplete data for the post-process search.

This search result retention limit matches the \max_count\ setting in \limits.conf. The setting defaults to 500,000.

**Client timeout**

If the post-processing operation takes too long, it can exceed the Splunk Web client timeout value of 30 seconds.

For more information about transforming searches, see transforming commands and searches in the *Search Manual*.

**Do not reference fields in post-process searches that are not referenced in the base search**

A post-process search depends entirely on the fields present in the base search. If you are not referencing a particular field in the base search, do not reference it in the post-process search. Fields used in transforming commands will automatically be available for post-process searches. When transforming commands are not used in a base search, fields without a reference in the base search appear null in a post-process search. The post-process search returns no results in this case.

**Limit base search results and post-process complexity**

Passing a large number of search results to a post-process search can cause server timeout issues. In this scenario, consider adjusting the base search to reduce the number of results and fields that it returns. You can also consider reducing the complexity of post-process operations on the base search results.
**Examples**

The following examples show you how to use base searches and post-process searches.

**Basic post-process search**

The base search in this example uses the `<stats>` transforming command. Two post-process searches use the base search results differently.

**Base search**

```
index=_internal source=*splunkd.log | stats count by component, log_level
```

**First post-process search**

```
| stats sum(count) AS count by log_level
```

**Second post-process search**

```
| search log_level=error | stats sum(count) AS count by component
```

**Dashboard source code**

```
<dashboard>
   <label>Dashboard with post-process search</label>

   <!-- Example uses stats transforming command -->
   <!-- This limits events passed to post-process search -->
   <search id="baseSearch">
     <query>
       index=_internal source=*splunkd.log | stats count by component, log_level
     </query>
   </search>
</dashboard>
```

127
Chained post-process searches

Chain two or more post-process searches together. Start with a base search and post-process search. Use the first post-process search as the base search for another post-process search.

Dashboard source code

```xml
<row>
  <panel>
    <chart>
      <title>Event count by log level</title>

      <!-- post-process search -->
      <search base="baseSearch">
        <query>
          stats sum(count) AS count by log_level
        </query>
      </search>

    </chart>
  </panel>
  <panel>
    <chart>
      <title>Error count by component</title>

      <!-- post-process search -->
      <search base="baseSearch">
        <query>
          search log_level=error | stats sum(count) AS count by component
        </query>
      </search>

      <option name="charting.chart">bar</option>
    </chart>
  </panel>
</row>
</dashboard>
```
Complex post-process search

When creating a complex base search that includes statistical aggregations such as percentiles or standard deviations, use summary indexing commands.

Summary indexing commands, such as the following commands, provide more flexibility for post-process searches.

- `sistats`
- `sitimechart`
- `sitop`
- `sichart`
- `sirare`

For more information on summary indexing, see Use summary indexing for increased reporting efficiency and About transforming commands and searches in the Reporting Manual.

Example
This example includes a base search using the `sistats` summary indexing command.

The base search reports event size (min, avg, max) by source and sourcetype for the `_internal` index. Use the `sistats count` with the various group-by clauses. You lose the benefits of map-reduce in distributed search if you do not include these.

**Base search**

```
index=_internal | eval event_size=len(_raw) <br />
| sistats count
min(event_size) avg(event_size) max(event_size)<br />
by source sourcetype
```
Post process 2

| stats avg(event_size) by sourcetype

Dashboard source code

```xml
<dashboard>
  <label>Dashboard with post-process using summary indexing</label>
  <!-- Base search with summary indexing transforming command -->
  <search id="baseSearch">
    <query>
      index=_internal | eval event_size=len(_raw)
      | sistats count min(event_size) avg(event_size) max(event_size)
      by source sourcetype
    </query>
  </search>
  <row>
    <panel>
      <single>
        <title>Total event count</title>
        <search base="baseSearch">
          <query>stats count</query>
        </search>
        <!-- post-process search -->
        <option name="rangeColors">"0x53a051","0x0877a6","0xf8be34","0xf1813f","0xdc4e41"</option>
        <option name="underLabel">Total Events</option>
      </single>
    </panel>
    <panel>
      <chart>
        <title>Average event size by source type</title>
        <search base="baseSearch">
          <query>stats avg(event_size) by sourcetype</query>
        </search>
        <!-- post-process search -->
        <option name="charting.axisY.scale">log</option>
      </chart>
    </panel>
  </row>
</dashboard>

130
Post-process search to populate form inputs

Use a post-process search to populate a form input

The first input in this form lets users select an index to search. This input has statically defined choices. The second input uses a post-process search to define choices dynamically.

Base search

index=_internal | stats count by sourcetype

Post process to generate input choices

| search sourcetype=splunkd*

Form source code

<form>
  <label>Post Process in Form Inputs</label>
  <search id="searchInput">
    <query>index=_internal | stats count by sourcetype</query>
    <earliest>-60min</earliest>
    <latest>now</latest>
  </search>
  <fieldset submitButton="false">
    <!-- Input with statically defined choices -->
    <input type="dropdown" token="index_tok" searchWhenChanged="true">
      <label>Select an index to search</label>
  </fieldset>
</form>
Panel search controls

Click refresh and stop buttons on a dashboard panel to generate new search results or stop a running search job. The stop button replaces the refresh button.
in the panel while a search is running. Visualizations render available results from stopped search jobs and a timestamp shows the time of the most recent job.

By default, the stop search button is visible. You can hide this button using the \texttt{link.stopSearch.visible} Simple XML option in your dashboard source code. See the Simple XML Reference for more details.

**Stop a post-process search**
Stopping a post-process search causes the base search for that post-process search to stop anywhere it is used. If the same base search is used in multiple panels, it stops in those panels too.

**Troubleshoot referenced real-time searches in search head clusters**

In a search head clustering (SHC) deployment, if you are referencing a real-time saved search in a dashboard on a search head, the real-time search might not continue to stream data after initial results are returned.

There are two workarounds for this issue.

<table>
<thead>
<tr>
<th>Option</th>
<th>Example dashboard source code</th>
<th>Performance considerations</th>
</tr>
</thead>
</table>
| Use an inline real-time search in the dashboard panel instead. | \begin{verbatim}<search>
    <query>index=_internal | stats count
   </query>
    <earliest>rt-5m</earliest>
    <latest>rtnow</latest>
</search>\end{verbatim} | This type of search runs only when users view the dashboard. However, a new real-time search spawns for each user that accesses the dashboard from the search head or another member. |
| Create a scheduled saved search. | \begin{verbatim}<search>
    <query> | loadjob
    savedsearch=admin:search:SavedSearch
</query>\end{verbatim} | Only one instance of the saved search runs at the |
Use the `loadjob` command in an inline panel search to update the dashboard with the saved search results.

Additional search resources

If you are new to the Splunk platform and the search processing language (SPL), start with the Search Tutorial. This tutorial introduces you to the Search and Reporting application. The tutorial guides you through adding data to your Splunk deployment, searching your data, and building simple reports and dashboards.

The Search Manual includes detailed information about creating and optimizing searches, retrieving events, specifying time ranges, and using subsearches.

The Search Reference is a reference guide for the Search Processing Language (SPL). The Search Reference contains a catalog of the search commands with syntax, descriptions, and examples.

Dashboards and forms

Use dashboards and forms to visualize, organize, and share data insights.

Dashboards and forms have one or more rows of panels. Each panel contains a visualization, such as chart, table, or map. In each panel, a search generates data for the visualization.

Forms are different from dashboards because they include `<input>` elements, such as text boxes or radio buttons, for user interactions. You can configure elements in a form, such as a panel, to respond to user input by customizing the searches that drive visualizations or changing other behavior.

For more details on building a `<dashboard>` or `<form>`, see the Simple XML Reference.
## Anatomy of dashboards and forms

See the Simple XML Reference for complete information on dashboard and form element hierarchy.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>top-level element</strong></td>
<td><code>&lt;dashboard&gt;</code> or <code>&lt;form&gt;</code></td>
</tr>
<tr>
<td>Title</td>
<td><code>&lt;label&gt;</code> (Optional)</td>
</tr>
<tr>
<td>Description</td>
<td><code>&lt;description&gt;</code> (Optional)</td>
</tr>
<tr>
<td>Global search</td>
<td>Global search is for use with post-process searches. Post-process searches have limitations. See Post-process limitations.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;search id=&quot;[identifier]&quot;&gt;</code></td>
</tr>
<tr>
<td><strong>Form inputs</strong></td>
<td><code>&lt;fieldset&gt;</code></td>
</tr>
<tr>
<td>(Forms only)</td>
<td><code>&lt;input&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;text&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;checkbox&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;dropdown&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;multiselect&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;radio&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;search&gt;</code> (to populate input choices)</td>
</tr>
<tr>
<td><strong>Rows</strong></td>
<td>Each row contains one or more panels.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;row&gt;</code></td>
</tr>
<tr>
<td><strong>Panels</strong></td>
<td>Each panel contains an optional title, optional inputs, and one or more visualizations. See Dashboard panels for the types of panels available.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;panel&gt;</code></td>
</tr>
<tr>
<td><strong>Visualizations</strong></td>
<td>A visualization displays data returned from a search.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;chart&gt; &lt;event&gt; &lt;map&gt; &lt;single&gt; &lt;table&gt;</code></td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>A search for a visualization.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;search id=&quot;[identifier]&quot;&gt; Base search for post-process searches.</code></td>
</tr>
</tbody>
</table>
### Dashboard examples

This topic shows the source simple XML code behind dashboards. After you become familiar with the simple XML source code, you can further customize the dashboard.

### Basic dashboard

This example uses a few simple XML elements to create a basic dashboard.

```xml
<dashboard>
    <!-- A title for the dashboard -->
    <label>Basic Dashboard</label>

    <!-- Provide a description -->
    <description>Illustrate the basic structures of a dashboard</description>

    <!-- Place panels within rows -->
    <row>
        <!-- Insert panels here -->
    </row>
</dashboard>
```
<!-- This basic dashboard has only a single panel -->
<panel>
  <table>
    <title>Top Sourcetypes (Last 24 hours)</title>

    <!-- A search powers the panel -->
    <search>
      <query>
        index=_internal | top limit=100 sourcetype | eval percent = round(percent,2)
      </query>
      <!-- Specify a time range for the search -->
      <earliest>-24h@h</earliest>
      <latest>now</latest>
    </search>

    <!-- Use options to further define how to display result data -->
    <option name="wrap">true</option>
    <option name="rowNumbers">true</option>
  </table>
</panel>
</row>
</dashboard>

**Searches power panels**

This dashboard illustrates the following searches:

- Inline search
- Search saved as a report
- Search from a prebuilt panel
- Inline search derived from a pivot
<dashboard>
  <label>Searches power dashboards</label>
  <description>Show the various searches to power a panel.</description>
  <!-- This row contains three panels -->
  <row>
    <panel>
      <table>
        <title>(Inline Search) Top Source Types</title>
        <!-- Inline Search -->
        <search>
          <query>
            index=_internal | top limit=100 sourcetype | eval percent = round(percent,2)
          </query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
        </search>
        <option name="rowNumbers">true</option>
      </table>
    </panel>
    <panel>
      <chart>
        <title>(Report) Top Source Types</title>
        <!-- Reference to a search saved as a report -->
        <search ref="Top Source Types Report" />
      </chart>
    </panel>
    <panel ref="top_source_types_in_the_last_hour" app="search" />
  </row>
</dashboard>
Use panels to visualize search results

You can display search results in a table or event listing, but also specify various charts. Use the `<chart>` element, specifying the chart type with the `<option>` child element.

```xml
<panel>
  <chart>
    <title>(Pivot) Game Purchases</title>
    <!-- Inline search derived from a pivot -->
    <search>
      <query>
        pivot Buttercup_Games Successful_purchases
        count(Successful_purchases) AS "Count of Successful purchases" SPLITROW product_name AS "product name" SORT 100 product_name
      </query>
    </search>
    <option name="charting.chart">pie</option>
  </chart>
</panel>
</dashboard>
```
<!-- specified for the other panels -->
<table>
<title>Top Source Types (Table)</title>
<search>
  <query>
    index=_internal | top limit=10 sourcetype
  </query>
  <earliest>-24h</earliest>
  <latest>now</latest>
</search>
</table>
</panel>

<!-- display same search as various charts -->
<panel>
  <chart>
    <title>Top Source Types (Bar)</title>
    <search>
      <query>
        index=_internal | top limit=10 sourcetype
      </query>
      <earliest>-24h</earliest>
      <latest>now</latest>
    </search>
    <option name="charting.chart">bar</option>
    <option name="charting.axisY.scale">log</option>
  </chart>
</panel>

<panel>
  <chart>
    <title>Top Source Types (Column)</title>
    <search>
      <query>
        index=_internal | top limit=10 sourcetype
      </query>
      <earliest>-24h</earliest>
      <latest>now</latest>
    </search>
    <option name="charting.chart">column</option>
    <option name="charting.axisY.scale">log</option>
  </chart>
</panel>

<panel>
  <chart>
    <title>Top Source Types (Pie)</title>
    <search>
      <query>
        index=_internal | top limit=10 sourcetype
      </query>
      <earliest>-24h</earliest>
    </search>
  </chart>
</panel>
</row>
</row>

<row>
  <panel>
    <chart>
      <title>Top Source Types (Line)</title>
      <search>
        <query>
          index=_internal | top limit=10 sourcetype
        </query>
        <earliest>-24h</earliest>
        <latest>now</latest>
      </search>
      <option name="charting.chart">line</option>
    </chart>
  </panel>
</row>
Dashboard with real time search

You can build a real-time dashboard using the Splunk Dashboard Editor or coding the dashboard using simple XML. This example shows how to code the simple XML.

To enable real-time searching, use the `<earliest>` and `<latest>` child elements to the `<search>` element. For example, if you want to enable real-time searching and display the data in a table, specify the following:

```xml
<table>
  <title>Look here for errors</title>
  <search>
```

You can also set a window for the real-time dashboard. For example, if you want to show real-time events but only from the last 5 minutes.

For more information on setting a search window, see Specify real-time time range windows in your search in the Search Manual.

**Specify custom colors for fields in charts**

Use the `charting.fieldColors` Simple XML property to customize field colors in a chart. The colors you select are the same each time the chart displays, regardless of other charts or color specifications in the dashboard.

For more details about this property, see `charting.fieldColors` in the Chart Configuration Reference.

**Example**

The following example shows how to specify colors for a chart showing error counts per sourcetype. The example uses this search.

```bash
index = _internal log_level=* | stats count(eval(log_level="ERROR")) as ERROR count(eval(log_level="WARN")) as WARN count(eval(log_level="INFO")) as INFO by sourcetype
```
Without `charting.fieldColors`, the visualization uses default field color mapping based on the order of values returned. Here, `ERROR` appears blue.

To change the field color mapping, add the `charting.fieldColors` property to the dashboard's Simple XML source code. For example, the `charting.fieldColors` configuration below defines these colors for each log level.

- INFO: green
- WARN: orange
- ERROR: red

```xml
<option name="charting.fieldColors">
  {"ERROR": 0xFF0000, "WARN": 0xFF9900, "INFO":0x009900, 
  "NULL":0xC4C4C0}
</option>
```

After adding `charting.fieldColors`, the chart now looks like this.

The following code implements a similar chart with custom field colors.

```xml
<panel>
  <html>
    Use <tt>eval</tt> function in the search to transpose the value of the log_level field into individual fields for <tt>charting.fieldcolors</tt>.
  </html>
  <chart>
    <title>Field colors example</title>
    <search>
```
Simple XML provides a set of simple XML elements that define properties that can be applied to all visualizations. For properties specific to certain types of visualizations, such as <chart> or <map>, use the <option> element to specify a property.

The use of a specific element or the <option> element varies. Consult the Simple XML Reference and Chart Configuration Reference for details on specifying panel properties.

The following table summarizes some of the elements available for all visualizations.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;title&gt;</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td>Add a title to your panel, such as Failed logins. The title displays at the top of the panel.</td>
</tr>
<tr>
<td>&lt;earliest&gt;</td>
<td>Splunk time format</td>
</tr>
<tr>
<td>&lt;latest&gt;</td>
<td>Restrict search results to a specific time window, starting with the earliest time and ending with the latest time. Specify &quot;rt&quot; to enable real-time searches.</td>
</tr>
</tbody>
</table>
The following example of a panel with a `<chart>` element shows how to specify a title and an inline search. It restricts search results to a 5 hour window and to three fields:

```xml
<dashboard>
  <label>My dashboard</label>
  <row>
    <panel>
      <table>
        <title>Top users, five hours ago</title>
        <search>
          <query>
            host=production | top users
          </query>
          <earliest>-10h</earliest>
          <latest>-5h</latest>
        </search>
        <fields>host,ip,username</fields>
      </table>
    </panel>
  </row>
</dashboard>
```

The following example specifies various properties with the `<option>` element for a `<table>`:

```xml
<dashboard>
  <label>My dashboard</label>
  <row>
    <panel>
      <table>
        <title>Errors in the last 24 hours</title>
        <search>
          <query>
            Errors in the last 24 hours
          </query>
        </search>
        <option name="count">15</option>
        <option name="displayRowNumbers">true</option>
        <option name="maxLines">10</option>
        <option name="segmentation">outer</option>
        <option name="softWrap">true</option>
      </table>
    </panel>
  </row>
</dashboard>
```
The following example specifies a column chart visualization, with display names for the X and Y axes.

```xml
<dashboard>
  <label>My dashboard</label>
  <row>
    <panel>
      <chart>
        <search>
          <query>
            sourcetype=access_* method=GET | timechart count by categoryId
            | fields _time BOUQUETS FLOWERS
          </query>
          <earliest>-7d</earliest>
          <latest>now</latest>
        </search>
        <title>Views by product category, past week (Stacked)</title>
        <option name="charting.axisTitleX.text">Views</option>
        <option name="charting.axisTitleY.text">Date</option>
        <option name="charting.chart">column</option>
      </chart>
    </panel>
  </row>
</dashboard>
```

**Use the HTML panel to display static text**

The HTML panel displays inline HTML. Use the HTML panel to add documentation, links, images, and other Web content to a dashboard.

Content between the HTML tags is displayed according to the specified HTML formatting. Relative link references are relative to the current view location. The HTML panel does not use any of the other general panel options and there are no specific options to set for HTML.

For details on using HTML panels, refer to the `<html>` element entry in the Simple XML Reference.
In the example, the anchor tag accesses system reports using the special Splunk locator: `@go?s=`

```html
... <row>  <panel>  <html>   <p>This is an <i><b>HTML panel</b></i> providing links to system reports.</p>  <ul>   <li>   <p><a href="@go?s=Errors in the last 24 hours">Errors in the last 24 hours</a></p>   </li>   <li>   <p><a href="@go?s=Errors in the last hour">Errors in the last hour</a></p>   </li>   <li>   <p><a href="@go?s=Indexing workload">Indexing workload</a></p>   </li>   <li>   <p><a href="@go?s=License Usage Data Cube">License Usage</a></p>   </li>  </ul> </html> </panel> ...</row>
```

**Configure drilldown**

Use drilldown to link to a search, another dashboard, or an external website. You can also use drilldown to trigger contextual changes in the same dashboard.
See Use drilldown for dashboard interactivity for more information.

Form examples

A form is similar to a dashboard, but provides an interface for users to supply values to one or more search terms, typically using text boxes, dropdown menus, or radio buttons. A form shields users from the details of the underlying search – it allows users to focus only on the terms for which they are searching and the results. The results can be displayed in tables, event listings, or any of the visualizations available to dashboards.

This topic contains basic examples that show how to create forms. Refer to the Splunk Dashboard Examples app for additional examples that use more robust source data. The examples show how to use tokens to pass values in forms. See Token usage in dashboards for details on token implementation.

Basic form example

The user input to a form defines tokens for the selected values of the input. A search in the form uses the tokens to specify the values to use in the search. The search accesses the value for the token using the ‘$...$’ as a delimiter for the token value.

For example, the following code snippet defines a dropdown that uses the sourcetype_tok token to represent the selection by the user. It also defines the choices for the dropdown.

```xml
<input type="dropdown" token="sourcetype_tok">
  <label>Select a source type</label>
  <default>splunkd</default>
  <choice value="splunkd">splunkd</choice>
  <choice value="splunk_web_access">splunk_web_access</choice>
  <choice value="splunkd_ui_access">splunkd_ui_access</choice>
</input>
```

The search in the form references the token. In the following code snippet, $sourcetype_tok$ represents the value from the dropdown choice.

```xml
<search>
  <query>
    index = _internal sourcetype=$sourcetype_tok$
    | timechart count by sourcetype
  </query>
  <earliest>-7d</earliest>
  <latest>-0d</latest>
</search>
```
Here is the simple XML implementing the form.

```xml
<form>
  <label>Form example: source type time chart</label>

  <!--autoRun means the search runs as soon as it is loaded. -->
  <!-- Do not need a submit button -->
  <fieldset autoRun="true" submitButton="false">
    <input type="dropdown" token="sourcetype_tok">
      <label>Select a source type</label>
      <default>splunkd</default>
      <choice value="splunkd">splunkd</choice>
      <choice value="splunk_web_access">splunk_web_access</choice>
      <choice value="splunkd_ui_access">splunkd_ui_access</choice>
    </input>
  </fieldset>

  <row>
    <panel>
      <chart>
        <search>
          <query>
            index = _internal sourcetype=$sourcetype_tok$
            | timechart count by sourcetype
          </query>
          <earliest>-7d</earliest>
          <latest>-0d</latest>
        </search>
      </chart>
    </panel>
  </row>
</form>
```

**Form with time inputs example**

You can add one or more time inputs to a form. If you add a single time input, a token for the time input is not necessary. The time input drives the data for all
searches in the form.

However if you add additional time inputs to a form, specify a token for each time input. The searches in the form reference the tokens to indicate which time input to use.

The following code snippet creates a time input that defines a token for local use.

```xml
<input type="time" token="time_tok" searchWhenChanged="true">
<label></label>
<default>
  <earliest>-24h@h</earliest>
  <latest>now</latest>
</default>
</input>
Use the earliest and latest modifiers to the time input token when accessing the local time input.

```xml
<search>
  <query>
    index=_internal sourcetype=$sourcetype_tok$
    | stats count as sourcetype
  </query>
  <earliest>$time_tok.earliest$</earliest>
  <latest>$time_tok.latest$</latest>
</search>
```

The following example uses a global timer that drives the Source Type Timechart panel. The Source Type Event Counter panel contains a local time for that panel only.

```xml
<form>
  <label>Form example: add time pickers</label>
  <fieldset autoRun="true" submitButton="false">
    <input type="dropdown" token="sourcetype_tok">
      <label>Select a source type</label>
      <default>splunkd</default>
      <choice value="splunkd">splunkd</choice>
  </fieldset>
</form>
```
<input type="time" searchWhenChanged="true">
<label>Select time range</label>
<default>
    <earliest>-7d@h</earliest>
    <latest>now</latest>
</default>
</input>

</fieldset>

<row>
    <panel>
        <title>Source type time chart</title>
        <chart>
            <search>
                <query>index = _internal sourcetype=$sourcetype_tok$
                    | timechart count by sourcetype</query>
            </search>
        </chart>
    </panel>
    <panel>
        <title>Source type event counter</title>
        <input type="time" token="time_tok" searchWhenChanged="true">
        <label></label>
        <default>
            <earliest>-24h@h</earliest>
            <latest>now</latest>
        </default>
        </input>
    </panel>
</row>
</form>

<!-- Local timer. Use tokens to access selected time. -->
<input type="time" token="time_tok" searchWhenChanged="true">
<label></label>
<default>
    <earliest>-24h@h</earliest>
    <latest>now</latest>
</default>
</input>

<search>
    <query>
        index=_internal sourcetype=$sourcetype_tok$
        | stats count as sourcetype
    </query>
</search>

<!-- Use the earliest and latest modifiers to the time input token -->
<earliest>$time_tok.earliest$</earliest>
<latest>$time_tok.latest$</latest>
Static and dynamic inputs to forms

The following form inputs require multiple choices for selection by the user. You can statically define the inputs or use a search to dynamically populate the inputs to a form.

- Check box
- Dropdown
- Multiselect
- Radio

The search in the following example compares static and dynamic definition for choices. The dropdown uses a populating search to define the choices.

- Populating <search>
  Returns fields to use for the label and value of the choices.

- <fieldForLabel> <fieldForValue>
  Child elements to the <input> element. These specify the fields to use to populate choices for the dropdown.

```xml
<form>
  <label>Populate an input with a search</label>
  <description>Events Filtered by User and Sourcetype</description>
  <!-- Do not need a Search Button. Inputs search when changed -->
  <fieldset autoRun="true" submitButton="false">
    <!-- Static definition of choices -->
    <input type="radio" token="username_tok" searchWhenChanged="true">
    <label>Select a User:</label>

    <!-- Define the default value -->
    <default>All</default>
  </fieldset>
</form>
```
<!-- Hard-code the choices -->
<choice value="*">All</choice>
<choice value="-">-</choice>
<choice value="admin">Admin</choice>
<choice value="nobody">Nobody</choice>
<choice value="splunk-system-user">Splunk System User</choice>
</input>

<!-- Dynamic definition of choices -->
<input type="dropdown" token="sourcetype_tok"
searchWhenChanged="true">
<label>Select a Sourcetype:</label>
<prefix>sourcetype="</prefix>
<suffix">"</suffix>
</input>

<!-- Define the default value -->
<default>splunkd</default>

<!-- Hard-code the choice for "All" -->
<choice value="*">All</choice>

<!-- Define the other choices with a populating search -->
<search>
<query>
index=_internal | stats count by sourcetype
</query>
</search>
</fieldset>


</row>
</panel>

<!-- Use tokens from the <input> elements in the panel title -->
<title>
Input selections: (radio) "$username_tok$", (dropdown) $sourcetype_tok$
</title>

<chart>

<!-- search for the visualization, references the input tokens--> 
<search>
<query>
index=_internal user=$username_tok$ $sourcetype_tok$ | timechart count
</query>
<earliest>-24h@h</earliest>
<latest>now</latest>
Create a form with a global search

You can create a form that uses a global search that drives the data in the various panels. This scenario is another form of post-process search. You should be careful about using post-process searches because of various limitations. In many cases, a post-process search is not always the most efficient way to use search resources. Read carefully the topic Post-process searches. It discusses Post-process limitations and other factors to consider before implementing a post-process search.

The following example shows a form with a global search.

The global search uses a transforming search command to avoid the 10,000 event limit for the number of events that you can pass to a post-process search:

```xml
<search id="global_search">
  <query>
    index=_internal source=*splunkd.log | stats count by component, log_level
  </query>
</search>

The values for the dropdown choices contain the post-process searches:

```xml
<fieldset autoRun="true" submitButton="false">
  <input type="dropdown" token="stats_tok" searchWhenChanged="true">
    <label>Select count by:</label>
    <default>Log level</default>
    <choice value="stats sum(count) AS count by log_level">Log level</choice>
    <choice value="search log_level=error | stats sum(count) AS count by log_level">Error level</choice>
</fieldset>
```
by component">Component</choice>
</input>

The panel in the form accesses the selected choice using the token from the dropdown:

```
<search base="global_search">
  <query>
    $stats_tok$
  </query>
</search>
```

Here is the complete code for a form with a global search:

```
<form>
  <label>Form with global search</label>
  <search id="global_search">
    <query>
      index=_internal source=*splunkd.log | stats count by component, log_level
    </query>
  </search>

  <fieldset autoRun="true" submitButton="false">
    <input type="dropdown" token="stats_tok" searchWhenChanged="true">
      <label>Select count by:</label>
      <default>Log level</default>
      <choice value="stats sum(count) AS count by log_level">Log level</choice>
      <choice value="search log_level=error | stats sum(count) AS count by component">Component</choice>
    </input>

    <input type="time">
      <default>Last 7 days</default>
    </input>
  </fieldset>

  <row>
    <panel>
      <chart>
        <option name="charting.chart">bar</option>
        <search base="global_search">
          <query>
            $stats_tok$
          </query>
        </search>
      </chart>
    </panel>
  </row>
</form>
```
Using a third party XML editor

In most cases, you use the Splunk Web dashboard editor to edit Simple XML. See About the dashboard editor to learn more.

If you are using Splunk Enterprise, you can also use a third-party editor to work with dashboard source code files in your deployment.

Splunk Cloud users cannot use a third-party editor because access to dashboard source code files is not available. If you have Splunk Cloud, use the dashboard editor in Splunk Web.

Source code files for dashboards and forms

Dashboard and form source code files can include the following.

- Simple XML
- JavaScript
- CSS
- Static HTML and image files imported by reference

File usage requirements

File system write access
You must have write access to the Splunk deployment file system to access the Simple XML files as well as supporting CSS and JavaScript files. If you do not have write access, check with an administrator.

Dashboard source file permissions
After copying dashboard source files, make sure that you can read and write to them. Read and write permissions on the files are defined separately from dashboard user access permissions.

File directories and locations

Use the local directory for source code files

When you edit Simple XML in the dashboard editor, source code file changes are written to the /local directory. Put dashboard source code files that you edit with a third-party editor in the /local directory.
**Caution:** Do not put Simple XML source files in the `/default` directory. Files in the `/default` directory are overwritten on deployment and app updates.

For more information on directories and file precedence, see Configuration file precedence.

**File location and permissions**

Source code file location depends on the file type and permissions.

**Simple XML and prebuilt panel source files**

The `/views` directory of an app contains the following files.

- Simple XML files
- Panel files available by reference in a dashboard. See Create and add a panel by reference for more information.
- Legacy Advanced XML files

Put Simple XML and panel source code files in the following locations for each permission type.

<table>
<thead>
<tr>
<th>Permission type</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared in app</td>
<td><code>$SPLUNK_HOME/etc/apps/&lt;app&gt;/local/data/ui/views/&lt;file_name&gt;</code></td>
</tr>
<tr>
<td>Private</td>
<td><code>$SPLUNK_HOME/etc/users/&lt;user&gt;/&lt;app&gt;/local/data/ui/views/&lt;file_name&gt;</code></td>
</tr>
</tbody>
</table>

**HTML files**

The `/html` directory in an app contains source files for dashboards converted to HTML.

Put HTML files in the following locations for each permission type.

<table>
<thead>
<tr>
<th>Permission type</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared in app</td>
<td><code>$SPLUNK_HOME/etc/apps/&lt;app&gt;/local/data/ui/html/&lt;dashboard_file_name&gt;</code></td>
</tr>
<tr>
<td>Private</td>
<td></td>
</tr>
</tbody>
</table>
Show source code file changes in Splunk Web

To display changes to dashboard source code files, refresh configurations on your Splunk deployment by using the debug/refresh endpoint.

http://localhost:8000/debug/refresh

After refreshing the instance, reload the edited dashboard.

Importing CSS, JavaScript, and other static files

A dashboard can import CSS and JavaScript files as well as image files and static HTML files. These files are in the following location. The files cannot be in a subdirectory.

$SPLUNK_HOME/etc/apps/<app_name>/appserver/static/

By default, this directory contains the following two files:

- dashboard.css
- dashboard.js

You can edit default files at this location or add additional CSS and JavaScript files. You can also add any HTML files that you want to reference from a dashboard.

Import JavaScript and CSS files

Use the script and stylesheet attributes in <dashboard> or <form> element to import a JavaScript or CSS file from the default location for an app. You can also reference script and CSS files from other apps.

Examples

Import files from the same app

<dashboard script="myScript.js" stylesheet="myStyles.css">
Import files from another app

<dashboard script="myApp:myScript.js" stylesheet="myApp:myStyles.css">
  ...
</dashboard>
Drilldown and Dashboard Interactivity

Use drilldown for dashboard interactivity

You might want to share additional data insights when users click on data points, table rows, or other visualization elements in a dashboard. Use drilldown to build this interactivity into your dashboards.

How drilldown works

Drilldown is a tool for configuring responses to user clicks on visualizations in a dashboard or form. Drilldown behavior is configured within individual visualizations. You can have separate drilldown configurations for each visualization in a dashboard. Depending on the visualization type, you can also enable drilldown on specific elements in a visualization, such as a table row or cell.

Drilldown actions

The drilldown actions that you configure happen when a user clicks the visualization element where the drilldown is enabled.

Link to a target
Drilldown can link a source dashboard or form to an external target that opens on a user click. The target can be a secondary search, another dashboard or form, or a website.

If you link to a website, make sure that you use a validly encoded URL. For example, any question marks (?) that should be recognized as question marks in a URL link should be changed to %3F. You can also create tokens for encoded URL characters that can be used in multiple places in your XML.

Trigger interactive behavior in the current dashboard
Drilldown can also trigger contextual changes in the same dashboard or form. For example, you can show or hide content depending on a clicked value.

Using tokens to customize a drilldown

Tokens are like programming variables. A token name represents a value that can change, such as a user selection in a form input. You can use a token name to access the value that it represents. In drilldown, you can use tokens to capture
contextual values from the current dashboard or values from clicked elements. You might also define custom tokens to help implement interactive behavior.

You can pass token values to a target search, dashboard, or URL by configuring a drilldown to set tokens in the target to the captured source values. Setting token values in this way lets you show customized content in the target.

You can also use token values to trigger interactive changes in the current dashboard, such as content display or more specific search results. Configure elements in the current dashboard to listen for and respond to these changes.

**Tokens available in drilldown**

Several predefined token types representing dashboard events are available within a drilldown context. You can use these tokens to access clicked fields, search events, and other dynamic values.

See Token usage in dashboards for details on working with the following token types.

*Form input change events*

Form inputs use a token to represent the value that users select in the input. If your drilldown target is a form, you can pass a value from a source dashboard to the input token in the target form so that users see content customized for the selected value.

To determine the token name for a form input, check the Simple XML source code for that input.

*Search events*

Predefined tokens represent search progress and completion events. Include search event handlers inside the `<search>` element for a visualization to get search job or result properties. You can use tokens to pass these values to the `<drilldown>` element.

*Tokens set on page load*

You can use an `<init>` element to set token values when a dashboard loads in the browser. You can access token values from the `<init>` element in a `<drilldown>`.
**Chart navigation and selection events**

You can access token values representing user pan and zoom or selection events in some chart types. See Chart controls for more information on working with these tokens.

**Predefined click event tokens**

Some tokens that you can use for drilldown are predefined in Splunk software. You can use these tokens to capture user actions or other values from a dashboard. For example, you can use the predefined $\text{click.value2}$ token to capture a clicked table cell value.

See Predefined drilldown tokens in the Simple XML Reference for a list of predefined tokens available for each visualization.

**Custom tokens**

In addition to predefined tokens, you can create custom tokens to help create dynamic or conditional display behavior. These tokens can represent other values that change, such as search results.

**Choose a drilldown action**

Choose a drilldown action depending on the type of interactive behavior that you want and the data insights that you are sharing with users.

To view the most common use cases and examples of how to use a drilldown actions, note that you can access the Splunk Dashboards Quick Reference guide by clicking the link in Getting started.

<table>
<thead>
<tr>
<th>Action</th>
<th>Type</th>
<th>Behavior and configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link to a search</td>
<td>Link to a target</td>
<td>Open a search page in the browser. A secondary search generates automatically to show results for the clicked value. You can also create a custom search.</td>
</tr>
<tr>
<td>Link to a different dashboard or form</td>
<td>Link to a target</td>
<td>Open a target dashboard or form in the browser. Use tokens to pass values to the target and show content customized to the clicked value or other values from the source.</td>
</tr>
<tr>
<td>Link to a URL</td>
<td>Link to a target</td>
<td>Open an external website in the browser. Pass token values from the source to the URL as query string parameters.</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>Manage token values in the current dashboard or form</td>
<td>Trigger interactivity in the current dashboard</td>
<td>Set, unset, or filter token values when a user clicks on an element in a dashboard or form. Instead of linking to a different location, use token value changes to configure interactive behavior in the same dashboard. For example, you can use <code>depends</code> or <code>rejects</code> attributes in the dashboard to control panel show or hide behavior when a token is set.</td>
</tr>
</tbody>
</table>

## Drilldown defaults and customization

Some drilldown components have default settings. Depending on the component, you can use the drilldown editor or Simple XML to customize them.

<table>
<thead>
<tr>
<th>Drilldown component</th>
<th>Default configuration</th>
<th>Where to customize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled?</td>
<td>If you are building a new visualization or dashboard, drilldown is disabled by default. If you are migrating existing dashboards to software version 6.6 or later, your prior drilldown settings, including drilldown enablement by default, are retained.</td>
<td>Enable or disable drilldown using the drilldown editor or in Simple XML.</td>
</tr>
<tr>
<td>Element in the visualization where drilldown is enabled</td>
<td>Varies by visualization. For example, you can enable drilldown on table rows or on single table cells to capture more specific clicked values. Check the Simple XML reference for defaults and options.</td>
<td>Use Simple XML to adjust the drilldown location.</td>
</tr>
<tr>
<td>Browser tab where linked searches,</td>
<td>In the drilldown editor, the option to open in a new tab is selected by default.</td>
<td>In the drilldown editor, you can opt out of opening the drilldown target in a new</td>
</tr>
</tbody>
</table>

163
In Simple XML, drilldown opens in the same tab by default. In Simple XML, add the target="blank" attribute to a `<link>` to open the target in a new tab.

**Default settings and source code synchronization**

Drilldown is disabled by default in new dashboard content. To disable drilldown, an `<option name="drilldown">none</option>` Simple XML element is added to visualizations that you save to a dashboard.

To avoid synchronization issues, do not delete this `<option>` from your dashboard source code. You can use the drilldown editor to change drilldown configurations or edit the `<option name="drilldown">` element without deleting it.

**Access the drilldown editor**

You can use the drilldown editor to enable or configure drilldown actions. Some advanced configurations, such as conditional linking, are available only in Simple XML.

**Steps**

1. In the dashboard where you want to configure drilldown, click **Edit**.
2. Find the panel where you are adding or updating drilldown. Click the additional options icon at the right. Select **Edit Drilldown**.

3. Use the editor to enable and configure drilldown actions.

For details on configuring specific drilldown actions in the drilldown editor and in Simple XML, see the options and linked topics in **Choose a drilldown action**.
Link to a search

You can create a drilldown that links users to search results for a clicked value. Users can view events to get more information on the value that they click.

How linking to a search works

Linking to a search lets users explore additional fields, values, and other data related to the value they click. You can use a default search or customize the search that opens on a user click.

Use the default search

A secondary search generates automatically to show more information about the clicked value. This search is similar to the search driving the source visualization but generates results that are more specific to the clicked value.

Customize the search

You can customize the secondary search to generate different results. For example, you might want to show results for a separate data set or include different fields or commands in the search string.

You can use predefined tokens to include the clicked value in the search.

Example

This column chart shows event counts by sourcetype over the last week.

The chart is generated using the following search string.

```
index = _internal | stats count by sourcetype
```

Drilldown linking to a default search is enabled on the chart. If a user clicks the `mongod` sourcetype column in the chart, a secondary search opens. This search
removes aggregations and generates an events list for the `mongod` sourcetype value.

Users can explore details for the `mongod` sourcetype that were not available in the column chart.

**Configure the drilldown in the drilldown editor**

You can use the drilldown editor to enable drilldown and configure linking to a search.

To create conditional or other advanced behaviors in a drilldown, use Simple XML. An error message appears if you access the drilldown editor in panels with existing advanced configurations.

**Prerequisites**

Some default drilldown settings are new in software version 6.6.0. Review drilldown defaults and customization in Use drilldown for dashboard interactivity.

**Steps**

1. From the dashboard where you want to configure drilldown, click **Edit** to open the dashboard editor.
2. Find the panel where you are configuring drilldown. Click the additional options button and select **Edit drilldown**.
3. Select **Link to search**.
4. Select a search type.
   - "Auto" generates a default search to remove aggregations and filter for values from the clicked element.
   - Select "Custom" to input a search string and time range.
5. (Optional) Opt to open the search in a new browser tab.
6. Click **Apply** to apply the drilldown settings.
7. Click **Save** to save your dashboard changes.

### Configure the drilldown in Simple XML

Click **Edit** to open the dashboard editor and click **Source** to access Simple XML source code.

**Drilldown defaults and customization**

Some default drilldown settings are new in software version 6.6.0. Review drilldown defaults and customization in Use drilldown for dashboard interactivity.

**Enable the drilldown**

Find the `<option name="drilldown">none</option>` element in the visualization. Change the option to enable and focus the drilldown. For example, in a table visualization, use `<option name="drilldown">cell</option>` to enable drilldown on table cells.

Once enabled, the drilldown links to a default search in the same browser tab.

**Search syntax**

**Default search**

Use the Simple XML `<option>` for the visualization where you are enabling drilldown. For example, the following source code in a bubble chart adds a drilldown linking to a search.
Check the Simple XML Reference to review the `<option>` name and syntax to use for each visualization.

**Custom search**

In your dashboard source code, add the `<drilldown>` element to customize the linked search.

This example uses the `target` attribute to open the search in a new browser tab. The drilldown links to the `search` page and uses the `q` parameter to pass in the custom search string.

```xml
<drilldown>
  <link target="_blank">search?q=index=_internal | stats count by sourcetype</link>
</drilldown>
```

**Link to custom search example**

You can customize a drilldown to change the results that users see when the secondary search opens.

**Default search**

This search generates a table aggregating customer actions on a retail website.

```xml
source="my_retail_data_source" | stats count by action
```

Drilldown linking to a default search is enabled in the table. The default search removes the aggregation by sourcetype and filters events for the selected sourcetype column. If a user clicks on the `addtocart` action, the following secondary search opens.

```xml
source="my_retail_data_source" action="addtocart"
```
The default search filters results for the selected action. You might want to show more specific details. To override this default behavior, create a custom secondary search in the drilldown editor or in Simple XML.

**Customizing the search**

Customize the drilldown search to show users more information about the products involved in customer actions. The following search string uses the \$click.value2\$ predefined token to capture the action that users click and filter product counts for it.

```
source="my_retail_data_source" action=$click.value2$ | stats count by productId
```

**Customizing the search in the drilldown editor**

In the drilldown editor, you can change the default search to a custom search. Input the search string and configure time range parameters as needed.

**Customizing the search in Simple XML**

You can use Simple XML to create the same custom search behavior. In the table visualization element, add the following source code.

```
<drilldown>
  <link target="_blank">search?q=source="my_retail_data_source" action=$click.value2$ | stats count by productId</link>
</drilldown>
```
The drilldown uses the $click.value2$ predefined token to capture the action that users click and use it in the search.

You can add custom time range parameters to the search. Use the <![CDATA[]]> wrapper or HTML character entities to escape the & ampersand or other special characters.

This example sets the time range for the last twenty-four hours.

<drilldown>
  <link target="_blank">
    <![CDATA[
      search?q=source="my_retail_data_source" action=$click.value2$ | stats count by productId&earliest=-24h@h&latest=now
    ]]>  
  </link>
</drilldown>

For more syntax details, see the <link> element in the Simple XML Reference.

**Link to a dashboard**

Set up a drilldown to link users to a target dashboard or form. This drilldown action lets you show users dashboard content related to the values that they click.

**How linking to a dashboard works**

A table shows customer activity, such as purchases or views, on a retail website. The table aggregates events to show which products and product categories have recent activity.
You can use a drilldown to link users to a dashboard showing more data about the product category that they click.

For example, the target dashboard might have single value visualizations showing purchase and removal counts for this category. It might also include a Choropleth map showing relative purchase counts by state. The following drilldown opens this target dashboard in a new browser window.

Use parameters to customize target content

Show custom content in the target by passing values to it from the source. You can configure a drilldown to capture a clicked or other value in the source dashboard and pass it as a parameter to the target.

As an example, you might have drilldown enabled on a table visualization. The predefined token gives you access to the value in a clicked table cell. You can configure the drilldown to set a token in the target to the captured value. On a user click, drilldown opens the target with a parameter set to the clicked value.

Setting tokens in target forms

Prefix form token names with form. if you are setting token values in a target form. For example, use the following configuration to set the token in the target form to the clicked table cell value from the source.

form.host = $click.value2$

Dashboard token names do not need a prefix. To set the token in a target dashboard to the clicked table cell value, use host = $click.value2$.

Configure the drilldown in the drilldown editor

You can use the drilldown editor to configure linking to an external dashboard or URL. The drilldown editor also lets you pass parameters between the source and target.

To create conditional or other advanced behaviors in a drilldown, use Simple XML. An error message appears if you access the drilldown editor in panels with existing advanced configurations.
Prerequisites
Some default drilldown settings are new in software version 6.6.0. Review drilldown defaults and customization in Use drilldown for dashboard interactivity.

Steps

1. From the dashboard where you want to configure drilldown, click Edit to open the dashboard editor.
2. Find the panel where you are configuring drilldown. Click the additional options button and select Edit drilldown.
3. Select Link to dashboard.
4. Select the target app and dashboard.
5. (Optional) Click Advanced to input parameter names and values to set in the target dashboard. You can use values captured from the source, such as $click.value$ to configure parameters in the target.
6. Click Apply to apply the drilldown settings.
7. Click Save to save your dashboard changes.

Configure the drilldown in Simple XML

Start building a dynamic drilldown by putting a <drilldown> element in a table or chart.
**Specify a drilldown target**

Inside the `<drilldown>` element, use a `<link>` element to indicate the drilldown target and to customize content in a target dashboard or form.

```
<drilldown>
   <link>...</link>
</drilldown>
```

The `<link>` element contains a path to the target and any token values that you are passing from the source to the target. These examples show you the syntax for specifying the target path and passing values.

<table>
<thead>
<tr>
<th>Target and behavior</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link to a dashboard in your Splunk deployment.</td>
<td>Use a relative path that includes the dashboard or form id. &lt;link&gt;[relative path]/[dashboard or form id] &lt;/link&gt;</td>
</tr>
<tr>
<td>Link to a form in your Splunk deployment. Show customized content in the form by passing a token value captured from the source. Use the token value to populate a form input.</td>
<td>Add a ? symbol after the relative path. Set tokens in the target to values passed from the source. This example sets a token in a target form to a value from the source. Prefix tokens in the target form with form., as shown here. &lt;link&gt;[relative path]/[dashboard or form id]?form.[target_token_name]=[$source_value$] &lt;/link&gt;</td>
</tr>
<tr>
<td>Pass the <code>&lt;earliest&gt;</code> and <code>&lt;latest&gt;</code> time range</td>
<td>Add &amp;earliest=$earliest$&amp;latest=$latest$ to the target path and token values. Use &lt;!--[CDATA[ ... ]]&gt; wrapper to make sure that the &amp; symbol is interpreted correctly. &lt;link&gt;</td>
</tr>
</tbody>
</table>
Additional information

- For more details on the `<link>` and `<drilldown>` elements, see the Simple XML Reference.
- See Token usage in dashboards for information on using and customizing tokens.

**Conditional linking**

You might want to configure conditional linking to different targets depending on the specific elements that users click in the source dashboard or form. To do this, add a `<condition>` element to the `<drilldown>`. The `<condition>` element contains the conditional `<link>` target and values to use.

A table field or chart series attribute in the `<condition>` indicates the field or series value to evaluate for conditional linking.

**Example**

A dashboard includes a table with columns A, B, and C.

If a user clicks a value in column A, open a form with a token set to the captured value.

```xml
<drilldown>

<condition field="A">
  <link>
    [relative_path]/[target_form_id]?form.[target_token]=$[value_from_source]$
  </link>
</condition>

[...]

</drilldown>
```
**Capture values from multivalue fields**

Multivalue fields can appear multiple times in an event. Each time that the field appears in an event, it can have a different value. You can configure a drilldown to link to specific targets depending on the value that users click.

When setting up a drilldown from a table, you typically use $click.name$ or $click.name2$ to capture the value that users click in a column or row. However, when working with multivalue fields, use $click.value2$ to capture the selected value for the drilldown. Use a `<condition>` element with a `field` attribute to limit the column selection to the multivalue field.

See Configure multivalue fields in the *Knowledge Manager Manual* for more information on working with multivalue fields in your data.

**Open the target in another browser tab**

By default in Simple XML, drilldown targets open in the same browser tab as the source dashboard or form. You can add a `target="blank"` attribute to the drilldown element to make the target open in a new browser tab.

**Example source code**

```xml
<dashboard>
  <label>Example linking drilldown</label>

  <row>
    <panel>
      <table>
        <title>Sourcetypes by source (Drilldown to a form)</title>
        <search>
          <query>
            index="_internal" | stats dc(sourcetype) by sourcetype, source
          </query>
          <earliest>-60m</earliest>
          <latest>now</latest>
        </search>
        <option name="count">15</option>
        <option name="showPager">true</option>

        <drilldown target="blank">
          <!-- Access the input on the target form, which is in the same app
          -->
          <!-- sourcetype.tok is the token for an input to the target form
          -->
```
Simple XML Examples

Dashboard linking to a form

Target form path
The form_for_drilldown form is the target for the following example. The relative path for this form is

/app/search/form_for_drilldown

Target token configurations
Before linking to a target dashboard or form, review the tokens that it has available. Make sure that the target and its tokens are configured to work with the values that you are passing to it in the drilldown action.

This target form has a dropdown input that lets users select a sourcetype value. The input uses the sourcetype token to represent the selected value. This token is used in a search that generates a chart showing results for this sourcetype.

Target form source code

<form>
  <label>Target form for drilldown</label>
  <fieldset autoRun="true" submitButton="false">
    <input type="dropdown" token="sourcetype">
      <label>Select a source type</label>
      <default>splunkd</default>
      <search>
        <query>
          index = _internal | stats count by sourcetype
        </query>
      </search>
    </input>
  </fieldset>
</form>
Source dashboard
Set up a drilldown that links from a source dashboard to the target form. Use tokens to capture values from the source and set values in the target. When users click on a table row in the dashboard, the form opens to show customized content.

Drilldown configuration
This drilldown uses the `<link>` element to indicate the form to open and to set the `sourcetype` token value in the form.

```
<drilldown>
<link>
<![CDATA[
  /app/search/form_for_drilldown?form.sourcetype=$row.sourcetype$&earliest=$earliest$&latest=$latest$
]]>
</link>
</drilldown>
```

In the `<link>` element, the following drilldown components configure linking and token setting when users click on a table row in the dashboard.

<table>
<thead>
<tr>
<th>Component</th>
<th>Drilldown behavior that this component configures</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target form path</td>
<td>Indicates the form to open when the user clicks on a table row</td>
<td><code>/app/search/form_for_drilldown</code></td>
</tr>
<tr>
<td>Token names and values</td>
<td>Tokens customize form content based on the table row that users click in the dashboard.</td>
<td>To pass token values from the source dashboard to the target form, query string parameters are included in the path after the ? symbol.</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>form.sourcetype=$row.sourcetype$</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>When a user clicks a table row in the dashboard, pass the <code>sourcetype</code> value from this row to the form. Set the <code>form.sourcetype</code> token in the form to the <code>$row.sourcetype$</code> value from the table row that the user clicked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>earliest=$earliest$&amp;latest=$latest$</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set the <code>earliest</code> and <code>latest</code> time range modifiers in the form to the <code>$earliest$</code> and <code>$latest$</code> values from the source dashboard.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The <code>&lt;![CDATA[...]]&gt;</code> tag makes sure that the &amp; character is interpreted correctly.</td>
</tr>
<tr>
<td>Complete dashboard source code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
<dashboard>
  <label>Dashboard linking to a form</label>
  <row>
    <table>
      <search>
        <query>
          index="_internal" group="per_sourcetype_thruput" | chart sum(kbps) over series
        </query>
        <earliest>-60m</earliest>
        <latest>now</latest>
      </search>
      <title>Top sourcetypes (drilldown example)</title>
      <option name="count">15</option>
      <option name="showPager">true</option>
    </table>
    <drilldown>
      <link>
        <! [CDATA[
          /app/search/form_for_drilldown?form.sourcetype=$row.sourcetype$&earliest=$earliest$&latest=$latest$
        ]]>
      </link>
    </drilldown>
  </row>
</dashboard>
```
Link to a URL

Link to an external website and share relevant content with your dashboard users.

You can link to a URL to help users find more information about the values that they click, such as relevant user posts on Splunk Answers or a product page on a retail website.

How linking to a URL works

Using drilldown to link to a URL is similar to linking to a dashboard or form. When users click on an element in the source dashboard, drilldown can open an target external website in a browser window.

Use parameters to customize target content

To customize the content that opens in the browser, you can include query string parameters with the URL that you use. You can configure a drilldown to capture a clicked or other value in the source dashboard and pass it as a parameter to the target.

As an example, you might have drilldown enabled on a table visualization. The $click.value2$ predefined token gives you access to the value in a clicked table cell. You can configure the drilldown to set a token in the target to the captured $click.value2$. On a user click, drilldown opens the target with a parameter set to the clicked value.

Configure the drilldown in the drilldown editor

You can use the drilldown editor to configure linking to an external URL. The drilldown editor also lets you include query string parameters for the target URL.

To create conditional or other advanced behaviors in a drilldown, use Simple XML. An error message appears if you access the drilldown editor in panels with
existing advanced configurations.

**Prerequisites**
Some default drilldown settings are new in software version 6.6.0. Review drilldown defaults and customization in Use drilldown for dashboard interactivity.

**Steps**

1. From the dashboard where you want to configure drilldown, click **Edit** to open the dashboard editor.
2. Find the panel where you are configuring drilldown. Click the additional options button and select **Edit drilldown**.
3. Select **Link to URL**.
4. Type the full URL for the target, including an `http://` or `https://` prefix. Append any query string parameters to the URL. For example, use `http://buttercupgames.com?product=$click.value$` to pass in a clicked product value from the source dashboard.
5. Click **Apply** to apply these drilldown settings.
6. Click **Save** to save these dashboard changes.

**Configure the drilldown in Simple XML**

You can configure a drilldown to link to a URL, pass in parameters, and create conditional linking behavior in Simple XML.

**Syntax**

Use the full URL for the resource that you want to show in the browser. You can add optional query string parameters to customize the content that users see.

```
<drilldown>
  <link>
    [target_URL]?q=[$source_value$]
  </link>
</drilldown>
```
**Conditional linking**

You might want to configure conditional linking to different targets or pass in different query string parameters depending on contextual values like search results or user clicks. To do this, add `<condition>` elements to the `<drilldown>`.

To evaluate user click locations, you can use the table field or chart series attribute in the `<condition>`.

You can also use a `<condition match=" ">` statement to evaluate contextual values and trigger linking or other actions conditionally. See Define conditional matching in Token usage in dashboards for more details.

**Example: Link to different targets**

This drilldown links from a table to different target URLs depending on the column that users click.

If a user clicks a value in column A, the drilldown passes the clicked field value to the URL as a query string parameter. If users click a value in column B, the drilldown passes this value to the URL.

```xml
<drilldown>
  <condition field="A">
    <link>[target_URL]?q={$[value_from_field_A]}$</link>
  </condition>

  <condition field="B">
    <link>[other_target_URL]?q={$[value_from_field_B]}$</link>
  </condition>
</drilldown>
```

**Example: Use custom logic to evaluate search results**

The following drilldown sets a token inside a search event handler to capture the `$result.count$` property representing a failed login event count. It uses a `<condition match=" ">` element to evaluate the number of failed logins. If there are more than 5,000 events, the drilldown links to an internal runbook for troubleshooting failed logins.

```xml
<single>
  <search>
    <query>source="recent_login_events" type=failed_login | stats count</query>
    <earliest>-24h@h</earliest>
    <latest>now</latest>
    <done>
```

181
Capture values from multivalue fields

Multivalue fields can appear multiple times in an event. Each time that the field appears in an event, it can have a different value. You can configure a drilldown to link to specific targets depending on the value that users click.

When setting up a drilldown from a table, you typically use $click.name$ or $click.name2$ to capture the value that users click in a column or row. However, when working with multivalue fields, use $click.value2$ to capture the selected value for the drilldown. Use a <condition> element with a field attribute to limit the column selection to the multivalue field.

See Configure multivalue fields in the Knowledge Manager Manual for more information on working with multivalue fields in your data.

Additional information

- To learn more about editing Simple XML, see Editing Simple XML.
- For more details on the <drilldown> and <link> elements, see the Simple XML Reference.

Example

Link to search results on Splunk Answers

Link users who click an element in a chart to relevant search results on the Splunk Answers community forum.

<link>
http://answers.splunk.com/search.html?q=$click.value$
</link>

This drilldown includes the following components in the <link> element.
<table>
<thead>
<tr>
<th>Component</th>
<th>Drilldown behavior that this component configures</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL for the external website</td>
<td>This URL points to a Splunk Answers search page.</td>
</tr>
<tr>
<td>Parameters setting token names and values</td>
<td>The $\text{click.value}$ predefined token captures the clicked value from the chart. This value passes to the Answers search URL as a query string parameter. In this example, the $q$ Answers search term parameter gets the clicked value and a search generates. When the user clicks a value and the Answers site loads, users see search results for this value.</td>
</tr>
</tbody>
</table>

**Complete form source code**

```xml
<form>
  <label>Form Search</label>
  <fieldset>
    <!-- Use the html tag to specify text to display -->
    <html>
      <p>Enter a sourcetype in the field below. This view returns the most recent 1000 events for that sourcetype.</p>
      <p>In the Matching Events, click in the series column to open the value clicked in a new form</p>
      </html>
    <!-- The default input is a text box with no initial value -->
    <input token="sourcetype" />
    <!-- Include a time picker -->
    <input type="time">
    <default>Last 30 days</default>
  </fieldset>
  <row>
    <panel>
      <!-- output the results as a 50 row events table -->
      <table>
        <title>Matching events</title>
        <table>
          <th>Thumbnails</th>
          <...>
        <table>
      </row>
    </panel>
  </row>
</form>
```
Manage token values in the current dashboard

Use drilldown to update token values and trigger responsive behaviors in the current dashboard. Instead of linking to a search or another dashboard, you can build a drilldown that manages token values when users click an element in the dashboard. Depending on the dashboard configuration, these token value changes can prompt different dynamic behaviors in it.

How managing token values works

Managing token values is part of creating responsive drilldown behavior in the current dashboard. Before you configure token value updates in the drilldown, plan the behavior that you want in the dashboard. For example, you can show or hide content on a user click. You can also set a token value that is part of a search string to generate a more customized visualization in the dashboard.

In the drilldown, you can set, unset, or use eval to change token values on a user click. Setting or updating token values does not result in dynamic dashboard behavior by itself. To create responsive changes, you configure elements in the dashboard to listen for these token updates.
As an example, you might want to show a relevant panel when a user clicks on a particular table in your dashboard. You can use the drilldown editor or Simple XML to configure a drilldown in this table to set the custom `show_content` token. You can then edit the dashboard source code to add a `depends` attribute to the panel so that it displays only if the `show_content` token is set.

The use cases and examples in this topic show you different options for responsive behavior in the current dashboard.

- Generate custom content
- Show or hide content
- Configure conditional behavior

**Manage and respond to token values in the drilldown editor and Simple XML**

You can use the drilldown editor to `set`, `unset`, or `eval` token values. After creating these configurations, you can use Simple XML to set up responses to the token values in the same dashboard.

**Prerequisites**

- Before changing token values, plan the responsive behavior that you want in the current dashboard. Typically, you create or use existing tokens whose values you `set`, `unset`, and `eval` in a drilldown to invoke responsive behavior. Make sure that tokens are available and placed to configure the behavior that you want. You can create new tokens or reference tokens in dashboard elements as needed.

- Some default drilldown settings are new as of version 6.6.0. Review drilldown defaults and customization in Use drilldown for dashboard interactivity.

**Steps**

1. From the dashboard where you want to configure drilldown, click **Edit** to open the dashboard editor.
2. Find the panel where you are configuring drilldown. Click the additional options button and select **Edit drilldown**.
3. Select **Manage token values on this dashboard**.
4. Select and configure token actions. You can include one or more token actions in the drilldown.
   ♦ To **set** a token, type the token name and new value to set.
   ♦ To **eval** a token value, type the token name and the eval expression that determines the token value.
   ♦ To **unset**, or remove, a token value, type the token name.
5. Click **Apply** to apply the drilldown settings.
6. Click **Source** to open the source code editor.
7. Use Simple XML to configure the behavior responding to the token value updates. For example, use the **depends panel** attribute to make a token setting determine whether a panel displays.
8. Click **Save** to save your dashboard changes.

**Manage and respond to token values in Simple XML**

You can use Simple XML to manage and respond to token values. Before configuring the token value changes, plan the responsive behavior that you want in the current dashboard. Check your dashboard source code to make sure that the tokens that you need are available. You can create or edit tokens to work with the behavior you are building.

As an example, if you want to show customized visualization content, make sure that the search string driving the visualization includes a token that you can use to customize results. You can configure the drilldown to `<set>` that token so that the search filters for a set value.

**Example visualization search string with a token to filter results**

\[
\text{index} = \_\text{internal} \mid \text{stats count by } \$\text{count\_field}$
\]

**Example drilldown to set this token to a clicked field name**

This drilldown uses the predefined `$\text{click\_value}$` token to capture the clicked
value. It sets the custom $count_field$ token to that value. The search string can then filter results for the clicked field.

```xml
<drilldown>
  <set token="count_field">$click.value$</set>
</drilldown>
```

**Configuration details**

From the dashboard where you are enabling drilldown, click Edit to open the dashboard editor and select Source to access the dashboard source code.

Find the element where you are adding or updating drilldown. Add a `<drilldown>` element and `set`, `unset`, or `eval` child elements to configure how token values change on a user click.

See the following examples for more details on creating responsive behavior in Simple XML.

- Generate custom content
- Show or hide content
- Configure conditional behavior

**Generate customized content**

You can use token updates to generate customized content when a user clicks a dashboard element. For example, use tokens in a search string to show a visualization that reflects clicked values.

**Example**

A Choropleth map shows recent sales activity in the United States. When users click on a state, a single value visualization shows users a sales total for the selected state.

This screenshot shows the dashboard after the user clicks on California in the
Choropleth map.

**Configuring the dynamic behavior**

When a user clicks on a state in the map, a token is set. The token captures the selected state value.

The token representing the clicked state on the map is used in the single value search string to filter results. The search generates a visualization specific to the selected state.

**Dashboard source code**

In the Choropleth map, a `drilldown` sets the `selected_state` token to the clicked location.

```xml
<map>
    <title>Sales events in the United States this week</title>
    [...]
    <drilldown>
        <set token="selected_state">$click.value$</set>
    </drilldown>
    [...]
</map>
```

In the same dashboard, the single value search uses this token to generate a sales total for the selected state. This search is similar to the one driving the Choropleth map, but it filters and aggregates events using the `selected_state` value.

The single value `<title>` also uses the `selected_state` token.

```xml
<single>
    <title>Sales events for $selected_state$</title>
</single>
```
Show or hide content

To show or hide content on a user click, use the tokens that you are updating in \textit{depends} or \textit{rejects} attributes in dashboard elements. These attributes let you control show or hide behavior when the tokens are set or unset.

\textit{Configuring dynamic display}

You might want to show a panel only when users click on an element in a dashboard. Add a \textit{depends} attribute to the panel in order to require the \textit{show\_panel} token to be set for the panel to display.

<panel depends="$show\_panel$">

Use a contextual drilldown to \texttt{set} the \textit{show\_panel} token when a user clicks on an element in the dashboard.

<drilldown>
  <set token="show\_panel">true</set>
</drilldown>

Prior to the the click event, the token is unset and the panel does not display. When the user clicks on the element where the contextual drilldown is configured, the \textit{show\_panel} token is set and the panel displays.

\textit{Example}

This dashboard shows a table with event counts by sourcetype. When users click on a particular table row, an additional visualization appears. The visualization shows events for the selected sourcetype.
This screenshot shows the dashboard after the user clicks the `splunkd_conf` sourcetype in the table.

**Controlling panel display**
The dynamic behavior in this example includes showing a panel and generating customized search results when users click a table row. To start, the `<drilldown>` sets a token for controlling panel display. The events list panel uses a `depends` attribute to require the `show_panel` to be set for the panel to appear.

**Customizing the dashboard content**
Setting a token in the `<drilldown>` also helps to generate custom content in the dashboard. In this example, the `selected_value` token is set to the clicked sourcetype value.

The search driving the events list visualization uses the `selected_value` token to generate results for the clicked sourcetype value. The panel title is also customized using the `selected_value` token.

**Dashboard source code**

```xml
<dashboard>
  <row>
    <panel>
      <table>
        <title>Event counts by sourcetype</title>
        <search>
          <query>index=_internal | stats count by sourcetype</query>
        </search>
        <drilldown>
          <set token="show_panel">true</set>
          <set token="selected_value">$click.value$</set>
        </drilldown>
      </table>
    </panel>
    <panel depends="$show_panel$">
      <event>
        <title>Recent events for $selected_value$</title>
      </event>
    </panel>
  </row>
</dashboard>
```
Configure conditional behavior

You might want to show different dashboard content depending on which element or value a user clicks. You can use `<condition>` elements inside the `<drilldown>` to define conditional token setting. This can help you to customize responsive behavior in the dashboard.

**Example**

A table shows event counts by `sourcetype` and `log_level`.

If a user clicks on a `sourcetype` value in the table, the dashboard shows a single value aggregating events for the selected sourcetype.

If a user clicks on a `log_level` value, an events list for the selected logging level appears.
Setting up the conditional drilldown behavior
The contextual `<drilldown>` includes `<condition>` elements to set token values differently depending on where users click.

To handle the different click locations, there is a `<condition>` element for each field, or column, in the table. For example, this `<condition>` defines token setting if a user clicks on a value in the `sourcetype` column.

```
<condition field="sourcetype">
  <set token="selected_sourcetype">$click.value2$</set>
  <set token="show_single_value">true</set>
  <unset token="show_event_list"></unset>
</condition>
```

The drilldown actions in this `<condition>` set up conditional content customization and display.

<table>
<thead>
<tr>
<th>Conditional behavior</th>
<th>Source code in the <code>&lt;condition&gt;</code></th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate customized search results</td>
<td><code>&lt;set token=&quot;selected_sourcetype&quot;&gt;$click.value2$&lt;/set&gt;</code></td>
<td>The <code>selected_sourcetype</code> token value represents the clicked sourcetype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The <code>$click.value2$</code> predefined token captures the clicked sourcetype.</td>
</tr>
<tr>
<td>Control panel display</td>
<td>Table cell value. The token value is part of a search generating a customized visualization.</td>
<td>To control which panels display when users click a sourcetype, set the <code>show_single_value</code> token and <code>&lt;unset&gt;</code> the <code>show_event_list</code>. Combined with <code>depends</code> attributes, these token updates trigger responsive hide and show behavior.</td>
</tr>
</tbody>
</table>

The `<condition>` configured to handle user clicks in the `log_level` column is similar.

**Dashboard source code**

```
<dashboard>
  <label>Contextual drilldown</label>
  <row>
    <panel>
      <table>
        <title>Event counts by sourcetype and log level</title>
        <search>
          <query>index=_internal | stats count by sourcetype, log_level</query>
        </search>
        <drilldown>
          <condition field="sourcetype">
            <set token="selected_sourcetype">$click.value2$</set>
            <set token="show_single_value">true</set>
            <unset token="show_event_list"/>
          </condition>
          <condition field="log_level">
            <set token="selected_log_level">$click.value2$</set>
            <set token="show_event_list">true</set>
            <unset token="show_single_value"/>
          </condition>
        </drilldown>
      </table>
    </panel>
  </row>
</dashboard>
```
Token usage in dashboards

Tokens are like programming variables. A token name represents a value that can change, such as a user selection in a form input. You can use tokens to access and pass these values to create more interactive dashboards.

Some tokens are predefined in Splunk software to provide environment, contextual, or user click event information. For example, you can use the $click.value2$ predefined token in a drilldown to access the table cell value that a user clicks.

You can also create custom tokens to implement interactive behavior. For example, define a $show_panel$ token that you can set or unset to control panel display.
To view the most common token commands at a glance, note that you can access the Splunk Dashboards Quick Reference guide by clicking the link in Getting started.

**Use cases**

You can use tokens to create interactive dashboard behavior in many contexts.

<table>
<thead>
<tr>
<th>Usage context</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search strings</td>
<td>Customize a search string by including tokens to represent dynamic values. When the search runs, it uses the token value.</td>
</tr>
<tr>
<td>Search event handlers</td>
<td>Search event handlers represent search state changes, such as when a search is done, in progress, or cancelled. You can add an event handler to a <code>&lt;search&gt;</code> element and use predefined tokens inside the handler to access search job metadata or first results row data. You can pass token values from a search handler to other dashboard elements to control behavior or content.</td>
</tr>
<tr>
<td>Form inputs</td>
<td>Form inputs use tokens to represent the values that users select or type. When you add an input to a form, a token generates automatically for it. You can customize the token name if needed. When forms have multiple time picker inputs, tokens connect individual time pickers with one or more visualizations in the form. A user time range selection controls the search time range for the visualization using that token. When using the <code>map</code> command in a form, use double dollar signs <code>$$</code> to specify a variable string. For example, <code>$$count$$</code>.</td>
</tr>
<tr>
<td>Set tokens on page load</td>
<td>Show customized initial content when a dashboard loads by setting token values in the <code>&lt;init&gt;</code> element.</td>
</tr>
<tr>
<td>Drilldown</td>
<td>Use tokens to configure drilldown behavior. Predefined and custom tokens let you customize content in linked searches, dashboards, or URLs. You can also use tokens to create interactive behavior in the same dashboard.</td>
</tr>
<tr>
<td>Define chart pan and zoom</td>
<td>Use predefined tokens to define pan and zoom areas for charts.</td>
</tr>
</tbody>
</table>
Using tokens in a search

Tokens capture and pass values in a dashboard. Token values can come from various sources, including form inputs and predefined token values for visualizations. Searches can access token values.

In a search, token name syntax uses $...$ delimiters. For example, if you define a form input token as field_tok, you can specify the token in a search as $field_tok$. Here is an example.

```
<search>
index=_internal source=*splunkd.log | stats count by $field_tok$
</search>
```

See Token filters for advanced syntax to access token values.

Define search tokens

You can set search tokens for a dashboard to display search job metadata or to control dashboard behavior.

There are many ways to use search tokens. Here are some example use cases.

- Including a search result count in a visualization title.
- If a search returns no results, run a different search or hide the panel.
- Hide or show panels if a search fails.

There are also various advanced options for working with search tokens. Options include the following:

- Show the time range of the search below the visualization element using HTML.
- Build a custom HTML element and insert search results as tokens.
- Define token values based on the result of token-specific `eval` expressions.

Search event elements and job properties

There are several search event handlers that you can use in Simple XML dashboards.
Within a search event handler, you can access specific job properties with tokens. For example, here are some commonly used job metadata tokens.

- \$job.earliestTime\$: Initial job start time.
- \$job.latestTime\$: Latest time recorded for the search job.
- \$job.resultCount\$: Number of results a search job returned.
- \$job.runDuration\$: Time, in seconds, for the search to complete.
- \$job.messages\$: List of error and/or debug messages generated by the search job.

To learn about more search job properties, see View search job properties in the Search Manual.

**Search tokens for dynamic display example**

Here is an example of the `<search>` element for a dashboard that hides a panel if no search results are returned.

```xml
<search id="search_logic">
  <query>$index_switcher$ | top sourcetype</query>
  <earliest>-60m@m</earliest>
  <latest>now</latest>
  <progress>
    <!-- match attribute for condition uses eval-like expression (see Splunk search language 'eval' command) -->
    <!-- logic: if resultCount is 0, then show a static html element, and hide the chart element -->
    <condition match="'job.resultCount' == 0">
```

197
For more examples, see the Splunk Dashboard Examples App.

Custom logic for dashboards

Add custom logic to a dashboard with the `<condition match=" "> and `<eval>` elements.

For both `<condition>` and `<eval>` elements, all data available from an event as well as the submitted token model is available as a variable within the eval expression.

Token syntax

As of software version 6.4, you can use either `$. . . $` delimiters or single quote delimiters for tokens in an `<eval>` or `<condition match=" ">` statement. For example, both of the following options are valid.

- `<condition match="$job.resultCount$ > 0">`
- `<condition match="'job.resultCount' > 0">`

Define conditional matching

Use the `<condition match=" ">` element to define conditional behavior. The following example controls a token value according to the result count job property.

```xml
<condition match=" $job.resultCount$ == 0">
  <set token="show_table_query">true</set>
</condition>
```

You can also use a dashboard eval expression to define a condition to match. Here is an example using `<condition match=" ">` to set a token value depending on whether the selected time range spans more than one day.

```xml
<condition match="relative_time(now(), earliest) - relative_time(now(), latest) > 86400">
```

198
Troubleshoot job property access
Job properties are not available throughout a dashboard by default. If conditional logic statements in your source code are not being applied as expected to a job property, add a search event handler to access the job property in a custom token. Use the custom token in the conditional logic statement instead.

In this example, the $search_results$ token gets the $job.resultCount$ job property value in a <search> event handler. The conditional logic statement evaluates the $search_results$ token and sets the $show_panel$ token accordingly.

[...]  
<search>  
  <query>index=_internal </query>  
  <earliest>-24h@h</earliest>  
  <latest>now</latest>  
  <done>  
    <set token="search_results">$job.resultCount$</set>  
  </done>  
</search>  
<drilldown>  
  <condition match=" $search_results$ != 0">  
    <set token="showPanel">true</set>  
  </condition>  
</drilldown>  
[...]

Using strings in a conditional statement
If you are using a <condition match= " " > statement to evaluate a string value, such as a sourcetype name from the first results row, put escaped quotation marks around the string value. This prevents the dashboard parser from handling the quotation marks as special characters.

The following example sets up conditional token setting that depends on the sourcetype field value in the first results row. If the sourcetype field value in the first results row is mongod, the "show_table" token is set to true.

To specify the "mongod" string in the conditional match statement, replace the quotation marks with the equivalent HTML character entities.
<condition match="'result.sourcetype'=='mongod'">
    <set token="show_table">true</set>
</condition>

For more information about using special characters in dashboard source code, see Editing Simple XML.

**Define token filtering and formatting**

You can use `eval` expression logic to define token filtering and formatting. For example, you can set a token value to the result of an `eval` expression.

**Dashboard `<eval>` expression functionality**

The dashboard `eval` expression has the same syntax and semantics as the `eval` expression syntax for SPL queries. Most of the same `eval` expression functionality is the same between the dashboard `eval` expression and the SPL version of `eval`. However, there are some important exceptions.

**Unavailable dashboard `eval` expression functions**

- `commands(X)`
- `searchmatch(X)`
- `exact(X)`
- Cryptographic hash functions:
  - `md5(X)`
  - `sha1(X)`
  - `sha256(X)`
  - `sha512(X)`
  - `sigfig(X)`
  - `spath(X,"Y")`

**`eval` expression functions with different behavior for dashboards**

- `relative_time(X,Y)`: Uses client time zone.
- `strftime(X,Y)`: Uses client time zone.
- `strptime(X,Y)`: Uses client time zone.

It is also important to note that regular expressions in dashboard `eval` expressions use the syntax and semantics of the JavaScript regular expression engine. This is not the same engine used for SPL `eval` expressions. If you are
using regular expressions in search tokens, check that syntax and semantics match those for JavaScript.

To learn more about eval expression functions, see eval in the Search Reference.

Custom logic examples

You can use an eval expression in <condition> event handler elements. Here is an example.

<condition match="[eval expression]">
  . . . [conditional actions] . . .
</condition>

You can also compute a token's value based on the result of an eval expression. Here is an example.

<eval token="new_token">[eval expression]</eval>

Define tokens for form inputs

All form inputs have a token attribute that defines a token for the user-selected value for the input. Form inputs also have child <prefix> and <suffix> elements that further modify the value of the token. For multiselect options, there are additional elements that can modify the value of the token. See Define tokens for multiselect inputs.

This code snippet defines a token for a drop-down list. The selected choice for the dropdown provides the value of the token.

<input type="dropdown" token="sourcetype_tok">
  <label>Select a source type</label>
  <default>splunkd</default>
  <choice value="splunkd">splunkd</choice>
  <choice value="splunk_web_access">splunk_web_access</choice>
  <choice value="splunkd_ui_access">splunkd_ui_access</choice>
</input>

See Form input example.
**Define tokens for multiselect inputs**

A multiselect input uses the `<prefix>`, `<suffix>`, `<valuePrefix>`, `<valueSuffix>`, and `<delimiter>` elements to build the multiselection search for the selected choices. The multiselection search, which is the value of the token for the input, ensures that the input passes all selected values to the search for the form.

The following code snippet shows how to build a value for the multiselect token. If a user selects both `splunkd` and `splunk_web_access` from the multiselect input, the token value is the following search fragment:

```
(sourcetype ="splunkd") OR (sourcetype ="splunk_web_access")
```

The search fragment derives from:

```
<prefix> + <valuePrefix> + [choice value] + <valueSuffix> + <delimiter>
... + <suffix>

  (sourcetype ="splunkd"
   _OR_
   )
```

```
<input type="multiselect" token="sourcetype_tok">
<label>Select one or more source types</label>

  <choice value="splunk_web_access">splunk_web_access</choice>
  <choice value="splunkd">splunkd</choice>
  <choice value="splunk_ui_access">splunk_ui_access</choice>
  <choice value="splunkd_access">splunkd_access</choice>

<!-- Build multi-selection search: (sourcetype ="value1" OR sourcetype ="value2" OR ...) -->
  <prefix>{</prefix>
  <valuePrefix>sourcetype ="</valuePrefix>
  <valueSuffix>"</valueSuffix>
  <delimiter> OR </delimiter>
  <suffix>)</suffix>
</input>
```

See Multiselect input example.

**Define tokens for time inputs**

If you have a form with panels that use different time pickers, use tokens for the time input to indicate the time picker to use for each panel. To access the earliest and latest values from a time picker, use the following modifiers to the token:

```
A time input that does not define a token is global. The values selected from such a time picker applies to all visualizations that do not otherwise specify a time picker.

See Time input example.

**Define tokens for conditional operations with form inputs**

You can define tokens for conditional operations for form inputs. The value of the token changes according to the condition you specify. For example, you can modify searches or select different visualizations to display based on the conditional value of a token.

Conditional operations include:

- Modify searches based on the token value.
- Hide or display panels and the content of panels based on a condition.
- Select a view to open based on a token value.

Conditional operations are available for form inputs and dynamic drilldown. Form inputs use various combinations of the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;change&gt;</td>
<td>Container element for the conditions that you define.</td>
</tr>
<tr>
<td>&lt;condition&gt;</td>
<td>Sets the condition based on the value of the input selection. In the Conditional input example, this is the value of the selected choice for the drop-down list.</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>Specify a link to a destination based on a condition.</td>
</tr>
<tr>
<td>&lt;set&gt;</td>
<td>Defines various values for a token. The &lt;search&gt; element in the dashboard consumes the value of this token.</td>
</tr>
<tr>
<td></td>
<td>In the Conditional input example, defines value for the token <code>earliest_tok</code>.</td>
</tr>
<tr>
<td>&lt;unset&gt;</td>
<td>Removes a token that was previously set.</td>
</tr>
</tbody>
</table>
This is useful for conditional operations that depend on a token being set.

See the example at Conditional operations with form inputs.

**Predefined tokens for accessing labels and values of form inputs**

Splunk Enterprise provides predefined tokens to access the label and value of form inputs. Tokens are available for the following inputs:

- check box
- drop-down list
- multiselect
- radio buttons

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>Contains the specified name of a form input choice.</td>
</tr>
<tr>
<td>value</td>
<td>Contains the value of a form input choice.</td>
</tr>
</tbody>
</table>

These tokens are useful to customize a search or place the label of the selected choice in a title or description of a panel or visualization.

See Access labels and values of form inputs.

**Set tokens on page load**

Add an `<init>` element to a dashboard or form to reuse content or create a template. The token values inside this element are set when the dashboard page loads.

**Guidelines**

Within a `<dashboard>` or `<form>` element, place content to set on page load inside the following tags.

```html
<init>
</init>
```

- You can use the following event handlers to specify token settings within the `<init>` tags.
  - `<condition>`
  - `<eval>`
  - `<link>`
• PDF scheduling is disabled for dashboards and forms that include an \<init> element.

• Token settings made within the \<init> element override any settings made in URL query string parameters.

• Token setting on page load is only supported for Simple XML dashboards. If you convert a dashboard to HTML, token settings within the \<init> element are disabled.

**Example**

This form sets an app name token on page load. The token value is used in a panel label and a search with the \$s$ filter to wrap the value in quotation marks.

```xml
<form>
    <label>Application Monitoring: Exchange</label>
    <init>
        <set token="app_name">my_app_name</set>
    </init>
    <row>
        <panel>
            <title>Activity Monitoring: $app_name$</title>
            <search>
                <query>index=main app=$app_name|s$</query>
            </search>
        </panel>
    </row>
</form>
```

**Use global tokens to access environment information**

Access details about the user, Splunk platform instance, and environment using global tokens. The following tokens are available.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$env:user$</td>
<td>Current user’s user name</td>
</tr>
<tr>
<td>$env:user_realname$</td>
<td>Current user full name.</td>
</tr>
<tr>
<td>Token</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$env:user_email$</td>
<td>Current user email address.</td>
</tr>
<tr>
<td>$env:app$</td>
<td>Current app context</td>
</tr>
<tr>
<td>$env:locale$</td>
<td>Current locale</td>
</tr>
<tr>
<td>$env:page$</td>
<td>Currently open page</td>
</tr>
<tr>
<td>$env:product$</td>
<td>Current instance product type</td>
</tr>
<tr>
<td>$env:instance_type$</td>
<td>Indicates whether the current instance is Splunk Cloud or on-premises deployment</td>
</tr>
<tr>
<td>$env:is_cloud$</td>
<td>Indicates if the current instance is Splunk Cloud. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_enterprise$</td>
<td>Indicates if the current instance is a Splunk Enterprise deployment. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_hunk$</td>
<td>Indicates if the current instance is a Hunk deployment. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_lite$</td>
<td>Indicates if the current instance is a Splunk Light deployment. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_lite_free$</td>
<td>Indicates if the current instance is using a Splunk Light free license. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_free$</td>
<td>Indicates if the current instance is using a Splunk Enterprise free license. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:version$</td>
<td>Current instance product version</td>
</tr>
</tbody>
</table>

**Define tokens for dynamic drilldown**

**Predefined tokens for dynamic drilldown**

Splunk Enterprise provides predefined tokens for dynamic drilldown. The predefined tokens capture values according to the location a user clicks in a visualization.

Predefined token availability and captured values vary according to visualization type. See the Simple XML Reference for a complete list of all predefined tokens.

**Define tokens for conditional operations with the <drilldown> element**

You can use tokens for conditional drilldown behavior such as:

- Set token values, based on a condition.
Select a value for multivalue fields in a visualization.
A multivalue field is a field that appears more than once with different values.

Select a view to open based on a token value.

Hide or show panels based on conditions.

Conditional operations are available for both form inputs and conditional drilldown. Defining tokens for conditional drilldown uses various combinations of the following tags:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;drilldown&gt;</td>
<td>Define link destinations for fields in a dashboard or form. You can also use with &lt;condition&gt; to set tokens for custom actions.</td>
</tr>
<tr>
<td>&lt;condition&gt;</td>
<td>Limit the scope of drilldown actions to specific fields.</td>
</tr>
<tr>
<td>&lt;selection&gt;</td>
<td>Use with the &lt;set&gt; element to set the time window for the pan and zoom features of charts. Applies to charts of type area, column, or line. See Chart controls and the &lt;selection&gt; entry in the Simple XML Reference.</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>Specify a link to a destination for drilldown.</td>
</tr>
<tr>
<td>&lt;set&gt;</td>
<td>Defines various values for a token.</td>
</tr>
<tr>
<td>&lt;unset&gt;</td>
<td>Removes a token value that was previously set.</td>
</tr>
</tbody>
</table>

Use the <set> element to define tokens

Use the <set> element to define tokens for conditional use. You can use the value of another token when defining a token with the <set> element. For example, the following code snippet defines the sourcetype_tok token. This token captures the value clicked from a <table> element for the field sourcetype.

```xml
<drilldown>
  <condition field="sourcetype">
    <set token="sourcetype_tok">$click.value2$</set>
  </condition>
</drilldown>
```
You can use the `sourcetype_tok` token in a search:

```plaintext
index=_internal sourcetype=$sourcetype_tok$ | timechart count by sourcetype
```

**Use the `<condition>` element to select a value for multivalue fields in a visualization**

Multivalue fields are fields that appear multiple times in an event and have a different value for each appearance. See Configure multivalue fields in the *Knowledge Manager* manual.

If you have a dashboard that displays multivalue fields, use the `<condition>` element to specify a drilldown location specific to the value of a clicked field. The following example links to different destinations based on the specific value for the field. The `<link>` element consumes different predefined tokens for each condition.

```xml
<drilldown>
  <condition field="badges">
    <link>
      /app/foursquare_vegas/vegas_badge_1?form.badge=$click.value2$
    </link>
  </condition>

  <condition field="venue">
    <link>
      /app/foursquare_vegas/vegas_venue_1?form.venue=$row.venue$
    </link>
  </condition>

  <condition field="links">
    <link>
      http://www.yelp.com/search?find_desc=$row.venue$&find_loc=Las+Vegas,+NV
    </link>
  </condition>
</drilldown>
```

**Syntax to consume tokens**

Use `$...$` delimiters to access the value of a token. For example, the following search for a visualization accesses the `field_tok` token. A form input previously defined the `field_tok` token:

```plaintext
index=_internal source=*splunkd.log | stats count by $field_tok$
```

208
## Token filters

Token filters ensure that you correctly capture the value of a token.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Wrap value in quotes</strong></td>
<td>Ensures that quotation marks surround the value referenced by the token. Escapes all quotation characters, &quot;&quot;, within the quoted value.</td>
</tr>
<tr>
<td>$token_name</td>
<td>s$</td>
</tr>
<tr>
<td><strong>HTML format</strong></td>
<td>Ensures that the token value is valid for HTML formatting.</td>
</tr>
<tr>
<td>$token_name</td>
<td>h$</td>
</tr>
<tr>
<td><strong>URL format</strong></td>
<td>Ensures that the token value is valid to use as a URL.</td>
</tr>
<tr>
<td>$token_name</td>
<td>u$</td>
</tr>
<tr>
<td><strong>Specify no character escaping</strong></td>
<td>Prevents the default token filter from running. No characters in the token are escaped.</td>
</tr>
<tr>
<td>$token_name</td>
<td>n$</td>
</tr>
</tbody>
</table>

The following code snippet uses the |s filter to place quotation marks around the value returned from a token:

```<search><query>
    index=_internal sourcetype=$sourcetype_tok|s$ | timechart count by sourcetype
</query>
</search>
```

If the value of sourcetype_tok is `access_combined`, it builds the following search string:

```index=_internal sourcetype="access_combined" | timechart count by sourcetype```

### Escape the $ token delimiter character

If you include static text that contains the $ character, use $$ to escape the token delimiter value.
**Combine literal values with token values**

You can combine literal values with the value returned from a token. Use with the `<set>` element to set conditional actions based on token values.

The following template combines the captured value from the predefined token, `click.value`, with static text. It places the value of `NewToken` in quotation marks.

```
<set token="NewToken">sourcetype=$click.value|s$</set>
```

If the value of `click.value` is `access_combined`, then the value of `NewToken` is the following search fragment:

```
sourcetype="access_combined"
```

You can use the `prefix` and `suffix` attributes to the `<set>` element to specify static text for a token value. The following example sets the value for `NewToken`. It is equivalent to the template example:

```
<set token="NewToken" prefix="sourcetype="&quot;" suffix="&quot;">$click.value$</set>
```

**Access tokens to show or hide user interface components**

You can use token values to conditionally show or hide user interface components. The following elements contain the attributes `depends` and `rejects`. Use the `<set>` and `<unset>` elements to set the token values that these attributes consume.

- `<row>`
- `<panel>`
- `<chart>`
- `<event>`
- `<html>`
- `<map>`
- `<single>`
- `<table>`
- `<input>`

For example, show the `<chart>` element only when the `showChart` token has been set.

```
<chart depends="$showChart$">
```

210
Define tokens for pan and zoom chart controls

Splunk Enterprise uses predefined tokens to implement the zoom feature on a chart. Using the zoom feature, you can select a portion of a data series in a chart that opens in a separate chart. See Pan and zoom chart controls.

Set the values of the predefined tokens within a `<selection>` element that is a child element of a chart. Use the token values in the original chart to display a new chart that zooms to the selection.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>start</code></td>
<td>Captures the value of the x-axis at the beginning and end of a selection in a chart. Valid only in the context of the chart. Assign the values to tokens that you define to access the values elsewhere in a dashboard.</td>
</tr>
<tr>
<td><code>end</code></td>
<td></td>
</tr>
<tr>
<td><code>start.&lt;field&gt;</code></td>
<td>Captures the values for the y-axis values for a selection. <code>&lt;field&gt;</code> represents a series displayed in the chart. Valid only in the context of the chart. Assign the values to tokens that you define to access the values elsewhere in a dashboard.</td>
</tr>
<tr>
<td><code>end.&lt;field&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>

See Pan and zoom chart controls for an example that shows how zoom to a selection in a time chart.

**Tokens with SplunkJS Stack**

If you are using SplunkJS Stack with JavaScript extensions, see Tokens and Data Binding on the Splunk Developer Portal to learn how to use tokens with JavaScript.

**Examples of token usage**

**Form input example**

This example shows the basic usage of tokens in form inputs. It uses a drop-down list to select the source type for the time chart. See Define tokens for form inputs.

The `<input>` element defines the `sourcetype_tok` that is consumed by the search...
for the visualization.

```xml
<form>
  <label>Form example: source type time chart</label>
  <fieldset autoRun="true" submitButton="false">
    <input type="dropdown" token="sourcetype_tok">
      <label>Select a source type</label>
      <default>splunkd</default>
      <choice value="splunkd">splunkd</choice>
      <choice value="splunk_web_access">splunk_web_access</choice>
      <choice value="splunkd_ui_access">splunkd_ui_access</choice>
    </input>
  </fieldset>
  <row>
    <panel>
      <chart>
        <search>
          <query>
            index = _internal sourcetype=$sourcetype_tok$
            | timechart count by sourcetype
          </query>
          <earliest>-7d</earliest>
          <latest>-0d</latest>
        </search>
      </chart>
    </panel>
  </row>
</form>

**Multiselect input example**

This example shows how to build a search string for a form input using static text and token values. This is useful for building multiselect options. See Define tokens for multiselect inputs.
The example uses the `<prefix>`, `<suffix>`, `<valuePrefix>`, `<valueSuffix>`, and `<delimiter>` elements to build the multiselect search string. When a user selects `splunkd` and `splunk_web_access`, it generates the following search string:

(sourcetype ="splunkd" OR sourcetype ="splunk_web_access")
**Time input example**

This example shows how to use both a global and local time picker in a form. It also shows how to access the predefined modifiers to a time input token. See Define tokens for time inputs.

The example shows a form with both a global time picker and local time picker. The `<chart>` element contains the local time picker and uses modifiers to the `local_time_input_tok` token to access the earliest and latest values.
<!-- Do not define token for global timer -->
<input type="time" searchWhenChanged="true">
  <label>Select time range</label>
  <default>
    <earliest>-7d@h</earliest>
    <latest>now</latest>
  </default>
</input>
</fieldset>

<row>
  <panel>
    <title>Global timer</title>
    <chart>
      <search>
        <query>index=_internal $source_tok$ | timechart count</query>
      </search>
    </chart>
  </panel>

  <panel>
    <title>Local timer</title>
    <input type="time" searchWhenChanged="true" token="local_time_input_tok">
      <label>Select time range</label>
      <default>
        <earliest>-24h@h</earliest>
        <latest>now</latest>
      </default>
    </input>
    <chart>
      <search>
        <query>index=_internal $source_tok$ | timechart count
          <earliest>$local_time_input_tok.earliest$</earliest>
          <latest>$local_time_input_tok.latest$</latest>
        </query>
      </search>
    </chart>
  </panel>
</row>
</form>
Conditional operations with form inputs

This example shows how to use conditional operations with form inputs. See Define tokens for conditional operations with form inputs.

The example uses the <change>, <condition>, and <set> elements to conditionally set the label for the selected time and to set the earliest time token. The search consumes the earliest time token to set the bounds for the search. This example uses the label and value predefined tokens for input choices. See Predefined tokens for accessing labels and values of form inputs.

Note: All input elements, with the exception of the time input, require a token attribute to be present. In the example, the input element defines the token, period_tok. However, this token is never consumed by the search.

<form>
  <label>Use tokens with conditional input choices</label>
  <fieldset submitButton="false">
    <input type="radio" token="period_tok">
      <label>Select a time range</label>
      <choice value="-24h@h">Last 24 Hours</choice>
      <choice value="-7d@h">Last 7 Days</choice>
      <choice value="-30d@h">Last 30 Days</choice>
      <default>Last 24 Hours</default>

      <!-- set condition based on the label defined by <choice> -->
      <!-- Within each condition, specify a custom label for display -->
      <!-- Capture the selected value in the token, earliest_tok -->
      <change>
        <condition label="Last 24 Hours">
          <set token="date_label">Yesterday</set>
          <set token="earliest_tok">$value$</set>
        </condition>
        <condition label="Last 7 Days">
          <set token="date_label">Last week</set>
        </condition>
      </change>
  </fieldset>
</form>
Access labels and values of form inputs

This example shows how to use tokens to access the labels and values of form inputs. See Predefined tokens for accessing labels and values of form inputs.

The example uses the label of the selected radio button in the title of the visualization. It uses the value of the selected radio button to determine the bounds of the search.
<form>
  <label>Use tokens with input choices to capture input labels and values</label>
  <fieldset submitButton="false">
    <input type="radio" token="period_tok">
    <label>Select a time range</label>
    <choice value="-24h@h">Last 24 Hours</choice>
    <choice value="-7d@d">Last 7 Days</choice>
    <choice value="-30d@d">Last 30 Days</choice>
    <default>Last 24 Hours</default>
  </fieldset>
  <change>
  <!-- use predefined input tokens to set -->
  <!-- tokens for the selected label and value -->
  <set token="date_label">$label$</set>
  <set token="earliest_tok">$value$</set>
  </change><row>
  <panel>
    <title>Conditional Inputs</title>
    <chart>
      <!-- Display selected label in the title -->
    </chart>
    <search>
      <query>index = _internal | timechart count by sourcetype</query>
      <!-- use the value of earliest_tok -->
      <!-- to set the time range -->
    </search>
    <option name="charting.axisY.scale">log</option>
  </panel>
</row>
</form>
Chart controls

This topic describes advanced behavior for viewing data in charts.

Pan and zoom chart controls

The pan and zoom feature allows you to highlight chart details and optionally view the details in a separate panel. Pan and zoom is available for the following charts:

- Column
- Line
- Area

The following examples show how to access the pan and zoom charting features.

Pan and zoom behavior

The following dashboard shows a chart displaying source types over a seven day period. The Y-axis uses a logarithmic scale to provide a more meaningful graphic. The panel specifies the following search.

index=_internal | timechart count by sourcetype

The screen capture below shows a selection for the results for two days.

The resulting chart zooms in to the selection and now displays details of the selected area.
• Use the left and right arrows along the X-axis to move the selection window earlier or later.
• Click **Reset Zoom** to return to the original chart.

**Zoom to another chart**

You can specify pan and zoom behavior to display results in a separate chart. The following example uses the same base example illustrated above in Pan and zoom behavior. The chart on the left lists all source types and also shows the selection for a single day. The other chart lists only the *splunk_web_access* source type for the selected time range.

You can drag an edge of the time range in the left chart to expand the time range. You can also move the selected time range to the left or right to specify an earlier or later time range.

The chart at the bottom shows the values for the tokens that implement the pan and zoom behavior.

**Implementation details**

To display zoom results in a separate chart, first edit the base chart in simple XML. Use the `<selection>` element to set token values for the selection time range.

**Note:** See **Token usage in dashboards** for information on tokens. The section Define tokens for pan and zoom chart controls provides details for tokens specific to pan and zoom behavior.
Predefined tokens that capture the values of the X-axis at the beginning and end of the selection time range. In this example, capture the time at the beginning and end of a time chart. The value is in epoch time.

Captures the values of the Y-axis for the specified series at the beginning and end of the selection. In this example, the value is the number of events for the field `splunk_web_access`.

The `start` and `end` tokens are valid only in the context of the chart. Assign the values to tokens that you define so you can access the values throughout the dashboard.

In the target chart, use `$selection_earliest$` and `$selection_latest$` to access the selection time range.
The HTML panel shows the values captured by the $start$ and $selection$ tokens.

<html>
<h3>Token values for the splunk_web_access selection</h3>
<table border="0" cellpadding="12" cellspacing="0">
<tr>
<td>
<p><b>Time range (epoch time)</b></p>
<p><b>$selection_earliest$</b>: $selection_earliest$</p><br/>
<p><b>$selection_latest$</b>: $selection_latest$</p>
</td>
<td>
<p><b>Count at the begining and end of time range.</b></p>
<p><b>$start_splunk_web_access$</b>: $start_splunk_web_access$</p><br/>
<p><b>$end_splunk_web_access$</b>: $end_splunk_web_access$</p>
</td>
</tr>
</table>
</html>

**Chart overlay**

Use chart overlays to represent two different series on a single chart. You can highlight one series of search results as a line graph on top of a column chart, area chart, or another line chart.

When using overlays, you can specify the overlaid values on a single axis or dual axis. With single axis, you plot the overlaid value and the search results against the same Y-axis. For dual axis, you specify a second Y-axis to represent the overlaid values.

**Chart overlay example (single axis)**

This example shows the count of splunk_web_access source type events over a time chart on a weekly basis for one month. Overlaid on this chart is the weekly average count of these events.
Here is the search to create this chart:

```
index=_internal sourcetype=splunk_web_access | timechart span=1week count | eventstats avg(count) as average | eval average=round(average,0)
```

You can create the overlay using the Visualization Editor.

1. From the dashboard, click **Edit > Edit Panels**.
2. Add a panel specifying the following:
   - **Content Title**: Chart Overlay (Single Axis)
   - **Search String**: The search string listed above.
   - **Time Range**: 30 days.
3. For the chart overlay panel, click the **Edit Properties** icon. Click **Chart Overlay**.
4. Click in the **Overlay** field. Select **average** from the fields available for selection as an overlay.
5. For **View as Axis**, click **Off**.
   This example does not specify a second Y-axis.

**Chart overlay example (dual axis)**

This example overlays the event count of the `splunk_web_access` source type against the total for all source types. The chart plots the Web Access totals against a separate Y-axis.

Here is the search to create this chart:

```
You can create the overlay using the Visualization Editor.

1. From the dashboard, click Edit > Edit Panels.
2. Add a panel specifying the following:
   - **Content Title**: Chart Overlay (Dual Axis)
   - **Search String**: The search string listed above.
   - **Time Range**: 30 days.
3. For the chart overlay panel, click the Edit Properties icon. Click Chart Overlay.
4. Click in the Overlay field. Select Web Access from the fields available for selection as an overlay.
5. For View as Axis, click On to specify a second Y-axis.
6. For Title, click Custom. Type Web Access in the adjacent text field to specify a title for the second axis.
7. For Scale, click Inherit to inherit the selection for the scale from the first Y-axis.
Share and Manage Dashboards

Generate dashboard PDFs

Dashboard PDF generation includes the following options.

- Generate and save a dashboard PDF.
- Print dashboard PDFs.
- Schedule PDF email delivery.

There are some limitations to PDF generation. See Limitations to PDF generation for details.

For information on sending scheduled report PDFs as email attachments, see Schedule reports in the Reporting Manual.

Generate and print dashboard PDFs

Generate a dashboard PDF

1. From the dashboard, select Export > Export PDF. The generated PDF appears in a browser window.
2. View, download, or print the PDF from the browser window.

Print a dashboard PDF

1. From the dashboard, select Export > Print. The default print driver for your browser opens with print settings.

Real-time searches and integrated PDF generation

PDF generation has special time range handling for real-time searches. PDFs for real-time searches, reports, or dashboards show results for the search time window relative to PDF generation time. As an example, when you generate a PDF for a real-time search with a five minute time window, the PDF shows search results for the past five minutes.

PDFs for dashboard panels with "real-time all time" search time ranges show results for the search over all time.
Schedule PDF delivery

Authorized users can schedule PDF delivery for dashboards. To set up PDF delivery, select Export > Schedule PDF delivery. Ensure that email notification settings are configured prior to scheduling PDF delivery.

For more information, see Configure email notification settings in the Alerting Manual.

Note: Scheduled PDF delivery is not available for dashboards that include forms or for dashboards converted to HTML.

Use tokens in scheduled dashboard delivery

Splunk software provides tokens that you can use to include information generated by a search in the fields of an email. For scheduled PDF delivery, you can use tokens in the following fields of an email:

- Subject
- Message
- Footer

Access the value of a token with the following syntax:

$<token-name>$

For example, place the following token in the subject field of a scheduled PDF delivery to reference the app containing the dashboard.

Search results from $app$

Tokens available for email notifications

You can use tokens to populate email notifications dynamically with search, server, or dashboard information.

See Use tokens in email notifications for details on dashboard metadata and other available tokens.
Schedule PDF delivery of a dashboard

Schedule PDF delivery of a dashboard:

1. For the dashboard you want to schedule, select Export > Schedule PDF Delivery.
2. Select the Schedule PDF Delivery check box to enable PDF delivery.

3. Select a schedule.
   If you select Run on Cron Schedule see cron examples.
4. Specify email details.
   You can use tokens in the Subject and Message fields.
   ♦ To, CC, and BCC email recipients.
      Specify a comma-separated list of email recipients.
   ♦ Priority
      Enforcement of priority depends on your email client.
   ♦ Subject
   ♦ Message
6. Click Save to save the schedule delivery settings.

To Discontinue a scheduled email delivery of a dashboard PDF

1. For the dashboard you want to discontinue PDF delivery, select Export > Schedule PDF Delivery.
2. Deselect the Schedule PDF Delivery.
3. Click Save to save the schedule delivery settings.
Scheduled views reports

When you schedule a PDF for delivery, a report called a scheduled view is created. Scheduled views don't appear in Searches, Reports, and Alerts under Settings. They only appear as a stanza in savedsearches.conf. The naming convention for scheduled views is _ScheduledView_<dashboard_name>, where dashboard_name is the name of the corresponding dashboard. It's a best practice to monitor the number of searches that are running to generate these dashboards, especially if you are experiencing problems with concurrent search limits for your deployment.

When a dashboard is deleted, the corresponding scheduled view is orphaned, but still present. By default, orphaned scheduled views are deleted every 24 hours. If you want to change the frequency of the setting, or disable it, you can use the following settings in limits.conf.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_reaper = &lt;boolean&gt;</td>
<td>Controls whether the scheduled views reaper runs. The reaper deletes orphaned scheduled views.</td>
<td>true</td>
</tr>
<tr>
<td>reaper_freq = &lt;integer&gt;</td>
<td>Controls how often, in seconds, the scheduled views reaper runs.</td>
<td>86400 (24 hours).</td>
</tr>
</tbody>
</table>

Never make changes to the configuration files in the default directory. For information on how to edit configuration files, see About configuration files.

Specify a cron schedule for PDF delivery

You can use cron notation to define a custom delivery schedule. Select the Cron option to input a schedule.

Cron parameters

When specifying a cron expression, only five cron parameters are available, not six. The sixth parameter for year, common in other forms of cron notation, is not available.

The cron parameters, * * * * *, correspond to minute hour day month day-of-week.

Example expressions
Here are some example cron expressions.

*/5 * * * *       Every 5 minutes.
*/30 * * * *      Every 30 minutes.
0 */12 * * *      Every 12 hours, on the hour.
*/20  * * * 1-5   Every 20 minutes, Monday through Friday.
0 9 1-7 * 1       First Monday of each month, at 9am.

**Additional configurations for PDF printing**

Splunk Enterprise users can specify the following configurations for PDF printing.

- Maximum number of table rows to print
- Timeout setting for generating a PDF
- Whether to include a logo
- Enable usage of non-Latin fonts

**Note:** If you are using Splunk Cloud and want these settings changed, file a Support ticket.

**Configure the number of rows in a table**

By default, 1000 rows are generated for a simple results table in a dashboard panel. If you have a dashboard with a table that has more than 1000 rows, the initial 1000 rows are rendered for the PDF, printing the results across several pages if necessary.

Splunk Enterprise users can override the default number of rows generated for PDF in the `limits.conf` file.

**To configure the maximum number of rows in a table that can be printed in PDF:**

1. Open `$SPLUNK_HOME/etc/system/local/limits.conf` for editing. Create this file if it does not already exist.
2. Specify the following property in the `[pdf]` stanza:

   ```
   [pdf]
   max_rows_per_table = <unsigned int>
   ```

   **Note:** This setting configures PDF settings for all tables in your Splunk deployment.
Configure the timeout setting for generating a PDF

The default timeout for generating a PDF is 3600 seconds, as specified in limits.conf. A complex search that is slow to completion might need additional time to generate the PDF.

To configure the timeout for generating a PDF:

1. Open $SPLUNK_HOME/etc/system/local/limits.conf for editing.
   Create this file if it does not already exist.
2. Specify the number of seconds to wait to generate a PDF. This property is in the [pdf] stanza:

   [pdf]
   render_endpoint_timeout = <unsigned int>

   **Note:** This setting configures PDF generation timeout settings for all PDFs in your Splunk deployment.

Configure whether to include the Splunk logo for a PDF

By default, the Splunk logo is included in a generated PDF. You can override the default setting in alert_actions.conf.

To not include the Splunk logo in a generated PDF:

1. Open $SPLUNK_HOME/etc/system/local/alert_actions.conf for editing.
   Create this file if it does not already exist.
2. Specify the following property in the [email] stanza:

   [email]
   reportIncludeSplunkLogo=0

   **Note:** This setting configures settings for all generated PDFs in your Splunk deployment.

Enable usage of non-Latin fonts in PDFs

Splunk software comes prepackaged with a collection of Latin fonts, and also a set of CID fonts for handling Japanese, Korean, Simplified Chinese, and Traditional Chinese.

You can control how Splunk software loads the CID fonts by making changes to the *reportCIDFontList* parameter in alert_actions.conf. Specify fonts in a
space-separated list. If multiple fonts provide a glyph for a given character code, the glyph from the first font in the list is used.

The `reportCIDFontList` parameter is in the [email] stanza. Make any changes for font usage here:

```sh
$SPLUNK_HOME/etc/system/local/alert_actions.conf
```

Here are the CID fonts supported by default:

```sh
gb cns jp kor
```

These reference Simplified Chinese, Traditional Chinese, Japanese, and Korean respectively.

To skip loading any CID fonts, in the local version of `alert_actions.conf`, leave the value of `reportCIDFontList` blank.

If you want your PDFs to use another non-Latin font (such as Cyrillic or Greek) ask an administrator add the Unicode font to `$SPLUNK_HOME/share/splunk/fonts`. Create the `fonts` directory if it doesn't already exist.

**Note:** When multiple fonts are installed, they are sorted by name in alphabetical order. For example, If you have Cyrillic and Greek installed, Splunk software always chooses Cyrillic unless you change the name of the files in `$SPLUNK_HOME/share/splunk/fonts` so that Greek comes first.

**Limitations to PDF generation**

Integrated PDF generation functionality has a few limitations.

- PDFs in languages, such as Hebrew, where text should appear in right to left order, are rendered with text in left to right order.
- PDFs for dashboards with multiple panels in a row might generate with only a single panel per row.
- You cannot generate PDFs of dashboards that are built using advanced XML or HTML. PDF generation works only with dashboards built with simple XML.
- You cannot generate PDFs for forms.
- PDF generation ignores charting customizations that are not supported by the JSCart charting library. The finished PDF displays the panels as rendered in JSCart with the unsupported customizations removed.
Configure dashboard permissions

Learn how to manage dashboard and panel search permissions.

Managing dashboard permissions

A dashboard is a knowledge object. Depending on your user role and capabilities, you might have different options for configuring dashboard permissions.

User role options

If you have the user role and its default capabilities, you can do the following.

- Create dashboards that are private to you.

Power user role options

If you have the power role and its default capabilities, you can do the following.

- Provide other users with read and write access to the dashboard.

Admin role options

If you have the admin role and its default capabilities, you can do the following.

- Create dashboards that are private, visible in a specific app context, or visible in all apps.
- Provide other users and roles with read and write access to the dashboard.

Managing panel search permissions

The search driving a dashboard panel is a separate knowledge object from the dashboard. The search has its own configureable permissions.

Panel search permissions

A panel search can run using the permissions of the user who created the search (the search owner), or a user who views the dashboard (a search user). The search user context can impact results and what different users see in a dashboard panel.
Depending on the results data access that you want to provide, you can adjust the permissions context for the search in the Reports listing page. Locate the search on this page and select Edit > Edit Permissions to change whether the search runs with the owner or user context.

For additional information on setting up permissions for dashboards and other knowledge objects see Manage knowledge object permissions in the Knowledge Manager manual.

**Specify permissions for a new dashboard**

When you create a new dashboard from the Search or Dashboard pages, you can configure permissions. Choose one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private</td>
<td>Only you have permission to view and edit the dashboard. The dashboard is not visible to other users.</td>
</tr>
<tr>
<td>Shared in app</td>
<td>The dashboard is available to other users in the app context where it was created. For example, if you create the dashboard in the Search and Reporting app, the dashboard is visible to other users in this context. Depending on their permissions, other users can edit the dashboard.</td>
</tr>
</tbody>
</table>

**Update dashboard permissions**

After creating a dashboard you can change the permissions.

1. Navigate to the Dashboards page.
2. Locate the dashboard whose permissions you are updating.
3. Under Actions, select Edit > Edit Permissions
4. Depending on your role and capabilities, specify the following details.

   **Display for**
   - Choose Owner to make the dashboard private. Select App to share the dashboard in the current app context or share the dashboard in All apps on the Splunk platform instance.

   **Read and write permissions**
   - Configure read (viewing) and writing (editing) privileges for other system users and/or roles.
Clone and manage dashboards

You can copy dashboards or select a dashboard to appear on the app home screen.

You can also adjust caching for deployments with large numbers of dashboards.

Clone a dashboard

Create a copy of a dashboard from the **Dashboards** page or using the dashboard editor.

**Steps**

<table>
<thead>
<tr>
<th>From</th>
<th>What to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboards page</td>
<td>1. Locate the dashboard that you want to set as the home dashboard.</td>
</tr>
<tr>
<td></td>
<td>2. Select <strong>Edit &gt; Clone</strong>.</td>
</tr>
<tr>
<td></td>
<td>3. (Optional) Update the cloned dashboard title and id.</td>
</tr>
<tr>
<td></td>
<td>Provide a description.</td>
</tr>
<tr>
<td></td>
<td>4. Click <strong>Clone dashboard</strong>.</td>
</tr>
<tr>
<td>Dashboard editor</td>
<td>1. Click the . . . button and select <strong>Clone</strong>.</td>
</tr>
<tr>
<td></td>
<td>2. (Optional) Update the cloned dashboard title and id.</td>
</tr>
<tr>
<td></td>
<td>Provide a description.</td>
</tr>
<tr>
<td></td>
<td>3. Click <strong>Clone dashboard</strong>.</td>
</tr>
</tbody>
</table>

Set a home dashboard

Configure a dashboard to appear on the app home page.

**Steps**

<table>
<thead>
<tr>
<th>From</th>
<th>What to do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboards listing</td>
<td>1. Locate the dashboard that you want to set as the home dashboard.</td>
</tr>
<tr>
<td>page</td>
<td>2. Select <strong>Edit &gt; Set as home dashboard</strong>.</td>
</tr>
</tbody>
</table>
Dashboard editor

1. Click the … button and select **Set as home dashboard**.

Adjust UI caching to handle large numbers of dashboards

Splunk Enterprise deployments with several hundred or more dashboards might have slower UI performance.

To improve UI performance, increase the default `max_view_cache_size` setting in the `web.conf` configuration file. For example, for an instance with 700 dashboards, you can increase this setting to 1000.

For more information, see the `web.conf` spec file.
Simple XML Reference

Simple XML Reference

You can use Simple XML source code to build dashboards and forms. Use the dashboard source code editor to access and edit Simple XML. In some cases, Simple XML provides customization options not available in the dashboard user interface editor.

Overview

Elements

Simple XML source code has parent and child elements, starting with a <dashboard> or <form> root element. Use elements to structure a dashboard or form. For example, you can use a <row> element to organize one or more <panel> child elements in a dashboard. A <panel> can contain visualization elements such as a <table> or a <map>.

Attributes and options

Most Simple XML elements have attributes and options that you can use to configure dashboard appearance and behavior.

In visualization elements, you can use options to configure formatting, interactivity, and other components, such as legend placement or the colors to use in a chart.

Attributes and options have types, like number or text. Some attributes and options have default settings and additional requirements, such as an accepted value range.

Using the Simple XML Reference

Get reference information about Simple XML elements for building dashboards and forms.
### dashboard or form

The `<dashboard>` or `<form>` root element appears at the top of dashboard Simple XML source code. A `<form>` root element indicates a dashboard that contains one or more inputs.

#### Dashboard element hierarchy

Dashboard definitions in Simple XML use the following parent and child element hierarchy.

```xml
<dashboard>
  <init> (0..1)
  <label> (0..1)
  <description> (0..1)
  <search> (0..1)
  <row> (1..n)
    <panel> (0..n)
      <search> (0..n)
        <chart> | <event> | <html> | <map> | <single> | <table> | <viz> (1..n)
        <search> (0..n, for each visualization element)
  </row>
</dashboard>
```

#### Form element hierarchy

Form definitions in Simple XML use the following parent and child element hierarchy. Unlike dashboards, forms have a `<fieldset>` element that contains `<input>` elements.

```xml
<form>
  <init> (0..1)
</form>
```
Attributes

Use the following attributes to configure `<dashboard>` and `<form>` root elements.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hideChrome</td>
<td>boolean</td>
<td>false</td>
<td>Hide the Splunk bar, app bar, and footer. For this and all of the following hide&lt;element_name&gt; attributes, if they are specified as a URL query string parameter without a value, they are handled as &quot;true&quot;. For example, both of the following query strings are handled as &quot;true&quot;. <code>&lt;dashboard_url&gt;?hideChrome</code> <code>&lt;dashboard_url&gt;?hideChrome=&quot;true&quot;</code></td>
</tr>
<tr>
<td>hideAppBar</td>
<td>boolean</td>
<td>false</td>
<td>Hide the bar listing available apps.</td>
</tr>
<tr>
<td>hideEdit</td>
<td>boolean</td>
<td>false</td>
<td>Hide the dashboard editing interface. If enabled, use the Settings &gt; User interface &gt; Views or the Dashboards page to edit the dashboard.</td>
</tr>
<tr>
<td>hideFilters</td>
<td>boolean</td>
<td>false</td>
<td>Hide &lt;form&gt; inputs to increase panel display space. Not applicable to &lt;dashboard&gt;.</td>
</tr>
<tr>
<td>hideSplunkBar</td>
<td>boolean</td>
<td>false</td>
<td>Hide the bar that provides a link to the home page and settings pages.</td>
</tr>
<tr>
<td>hideTitle</td>
<td>boolean</td>
<td>false</td>
<td>Hide the text defined in the &lt;label&gt; and &lt;description&gt; elements.</td>
</tr>
<tr>
<td>isDashboard</td>
<td>boolean</td>
<td>true</td>
<td>For internal use.</td>
</tr>
<tr>
<td>isVisible</td>
<td>boolean</td>
<td>true</td>
<td>Indicates if the dashboard or form appears in the app Dashboards listing page and navigation menus.</td>
</tr>
<tr>
<td>onunloadCancelJobs</td>
<td>boolean</td>
<td></td>
<td>Indicate whether to cancel search jobs when a user navigates away from the dashboard or form.</td>
</tr>
<tr>
<td>refresh</td>
<td>integer</td>
<td>0</td>
<td>Specify a refresh interval, in seconds. The dashboard or form updates every refresh seconds.</td>
</tr>
</tbody>
</table>
form reloads after the interval.

Comma-separated list of custom .js files to load. The files must be in a folder or subfolder of the appserver/static directory.

$SPLUNK_HOME/etc/apps/<app_name>/appserver/static/

To reference custom .js files from another app, specify the the app name when referencing the file. For example, use the following reference.

```html
<dashboard script="myApp:myScript.js">
```

Comma-separated list of custom stylesheets to use for the dashboard. The stylesheet files must be in a folder or subfolder of the following directory.

$SPLUNK_HOME/etc/apps/<app_name>/appserver/static/

To reference a custom css file from another app, specify the the app name when referencing the file. For example, use the following reference.

```html
<dashboard stylesheet="myApp:myStyles.css">
```

### Attribute examples

Use the hideEdit attribute to hide the editing interface from a dashboard.

```html
<dashboard hideEdit="true">
...
</dashboard>
```

Use the refresh attribute to reload a form after 30 seconds.

```html
<form refresh="30">
...
</form>
```

### Child elements

A `<dashboard>` or `<form>` root element can contain one or more of the following child elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>stylesheet</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| `<description>` | Optional text description for a dashboard or a form. If a `<label>` is also used, the description appears below the label in a smaller font.            | `<dashboard>`  
|            |                                                                                                                                             | `<label>Sales this month</label>`  
|            |                                                                                                                                             | `<description>The following panels show monthly sales totals in the United States.</description>`  
|            |                                                                                                                                             | `<row>`  
|            |                                                                                                                                             | `...`  
|            |                                                                                                                                             | `</dashboard>`                                                                                                                         |
| `<fieldset>` | Used within a `<form>` to organize form inputs. Not applicable to `<dashboard>`.                                                                                                                        | `<form>`  
|            |                                                                                                                                             | `<label>Events by sourcetype</label>`  
|            |                                                                                                                                             | `...`  
|            |                                                                                                                                             | `<fieldset>`  
|            |                                                                                                                                             | `<input type="text" token="series">`  
|            |                                                                                                                                             | `<label>Enter a source type</label>`  
|            |                                                                                                                                             | `<default></default>`  
|            |                                                                                                                                             | `<initialValue>splunkd</initialValue>`  
|            |                                                                                                                                             | `</input>`  
|            |                                                                                                                                             | `</fieldset>`  
|            |                                                                                                                                             | `...`  
|            |                                                                                                                                             | `</form>`                                                                                                                              |
| `<init>`    | Element containing token updates to make on page load. For example, set a token value to show specific search results when a dashboard opens. See Token usage in dashboards for more information on setting tokens on page load. | `<dashboard>`  
|            |                                                                                                                                             | `<init>`  
|            |                                                                                                                                             | `<set token="selected_sourcetype">mongod</set>`  
|            |                                                                                                                                             | `</init>`  
|            |                                                                                                                                             | `...`  
<p>|            |                                                                                                                                             | <code>&lt;/dashboard&gt;</code>                                                                                                                         |</p>
<table>
<thead>
<tr>
<th><code>&lt;label&gt;</code></th>
<th>Optional text label for a dashboard, form, or form input.</th>
<th><code>&lt;dashboard&gt;</code>&lt;br&gt;<code>&lt;label&gt;Recent login activity&lt;/label&gt;</code>&lt;br&gt;...&lt;br&gt;<code>&lt;/dashboard&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;row&gt;</code></td>
<td>Organizes <code>&lt;panel&gt;</code> elements in a dashboard or form. See row for more details.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;search&gt;</code></td>
<td>Global search element in a dashboard or form. A dashboard or form can contain one or more global <code>&lt;search&gt;</code> elements. If a dashboard or form contains a global search, use a post-process search within a <code>&lt;panel&gt;</code> element to show results in the panel. See search for more details.</td>
<td></td>
</tr>
</tbody>
</table>
fieldset (form)

A `<fieldset>` contains `<form>` inputs, such as a checkbox or radio buttons.

Element hierarchy

```
<form>
  <fieldset autoRun="[Boolean]" submitButton="[Boolean]">
    <html> (0..n)
    <input type="[input type]" token="[search token]"> (1..n)
      <default> (0..1)
      <fieldForLabel> (0..1)
      <fieldForValue> (0..1)
      <initialValue> (0..1)
      <label> (0..1)
      <prefix> (0..1)
      <search> (0..1)
      <selectFirstChoice> (0..1)
      <suffix> (0..1)
  </fieldset>
</form>
```

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoRun</td>
<td>Boolean</td>
<td>False</td>
<td>Indicates whether to run the search when the page loads.</td>
</tr>
<tr>
<td>submitButton</td>
<td>Boolean</td>
<td>True</td>
<td>Indicates whether to display a Submit button.</td>
</tr>
</tbody>
</table>

Child elements

A `<fieldset>` contains one or more `<input>` elements.

Example

```
<fieldset autoRun="true" submitButton="false">
  <input type="text" token="series">
    <label>sourcetype</label>
    <default></default>
  <initialValue>splunkd</initialValue>
    <suffix>*</suffix>
  </input>
</fieldset>
```
**input (form)**

Input elements define the user interaction interface for a form. Selection elements, such as radio buttons or a checkbox, let users click or select choices. You can use a search to populate selection elements dynamically. Other inputs, such as time or text, let users type content or change a search time range.

Use attributes to configure user inputs and use child elements to define input content. Some child elements are shared across several input types. Others are specific to the input type.

**Input attributes**

Use these attributes in any form input.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>depends</td>
<td>Comma-separated list</td>
<td>All tokens in the list must be defined for the input to render. These tokens might be used in an input or drilldown context in the form.</td>
</tr>
<tr>
<td>id</td>
<td>Text (minimum two characters)</td>
<td>Input identifier. Only alphanumeric and underscore characters are valid. The id cannot begin with a number or the underscore character.</td>
</tr>
<tr>
<td>rejects</td>
<td>Comma-separated list</td>
<td>Prevent input rendering if one or more tokens in this list are defined. These tokens might be used in an input or drilldown context in the form.</td>
</tr>
<tr>
<td>searchWhenChanged</td>
<td>Boolean</td>
<td>Specifies to run the search when the selection changes. Defaults to false.</td>
</tr>
<tr>
<td>token</td>
<td>Token name</td>
<td>Token representing the user selected value from this input.</td>
</tr>
<tr>
<td>type</td>
<td></td>
<td>One of the following input types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• checkbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• dropdown</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• link</td>
</tr>
</tbody>
</table>
Use these child elements in `<input>` elements. There are a few usage exceptions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Usage exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;change&gt;</code></td>
<td>See change</td>
<td>Not available for multiselect inputs.</td>
</tr>
<tr>
<td><code>&lt;condition&gt;</code></td>
<td>See condition (input)</td>
<td>Not available for multiselect inputs.</td>
</tr>
<tr>
<td><code>&lt;default&gt;</code></td>
<td>Default value for the input. If the user does not select or input a value,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>this value is used. Overrides the <code>initialValue</code>.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;earliest&gt;</code> and</td>
<td>The earliest and latest times defining the <code>&lt;search&gt;</code> time range. Use relative time</td>
<td>Not available for text or time inputs.</td>
</tr>
<tr>
<td><code>&lt;latest&gt;</code></td>
<td>modifiers, as described in Specify time modifiers in your search, or use UNIX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>epoch time format for absolute time.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;fieldForLabel&gt;</code></td>
<td>In dynamically populated selection inputs, these elements indicate the search</td>
<td>Not available for text or time inputs.</td>
</tr>
<tr>
<td>and <code>&lt;fieldForValue&gt;</code></td>
<td>result fields to use for the label and value of each choice. Use with the <code>&lt;search&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>element that generates results for populating the input choices.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;initialValue&gt;</code></td>
<td>Initial value for the input. This value is used before the user selects or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>inputs a value.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;label&gt;</code></td>
<td>Label for the input.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;prefix&gt;</code></td>
<td>Prefix to prepend to the selected value. For example, you can add a quotation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mark to the beginning of a string value and use <code>&lt;suffix&gt;</code> to add a closing</td>
<td></td>
</tr>
</tbody>
</table>
quotation mark after the string value. The prefix can be a string or regular expression.

<table>
<thead>
<tr>
<th>element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;search&gt;</td>
<td></td>
<td></td>
<td>Search that dynamically populates choices for the input. Use the ref attribute of the &lt;search&gt; element to reference a search from a report. See &lt;search&gt;. Not available in text inputs.</td>
</tr>
<tr>
<td>&lt;suffix&gt;</td>
<td></td>
<td></td>
<td>Suffix to append to the selected value. The suffix can be a string or regular expression.</td>
</tr>
</tbody>
</table>

**Checkbox child elements**

In addition to shared input child elements, you can use the following child elements in a `<checkbox>`.

<table>
<thead>
<tr>
<th>element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;delimiter&gt;</td>
<td>text</td>
<td></td>
<td>A string that will be placed between each selected value. Typically, you specify &quot; OR &quot; or &quot; AND &quot; using upper case – do not specify the quote marks, but specify a space character before and after the text.</td>
</tr>
<tr>
<td>&lt;valuePrefix&gt;</td>
<td>text</td>
<td></td>
<td>String prefixed to the value of the input element. Can be a regular expression.</td>
</tr>
<tr>
<td>&lt;valueSuffix&gt;</td>
<td>text</td>
<td></td>
<td>String appended to the value of the input element. Can be a regular expression.</td>
</tr>
</tbody>
</table>

**Dropdown child elements**

In addition to shared input child elements, you can use the following child elements in a `<dropdown>`.

<table>
<thead>
<tr>
<th>element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;allowCustomValues&gt;</td>
<td>boolean</td>
<td>false</td>
<td>If true, enables the selection of custom values typed into the text field for the input.</td>
</tr>
<tr>
<td>&lt;choice value=[value]&gt;</td>
<td>text</td>
<td>value</td>
<td><strong>Required.</strong> Specifies the value to use for the choice.</td>
</tr>
</tbody>
</table>
Specifies choices for a radio or dropdown element. `<choice>` is the label to use for the specified value.

<table>
<thead>
<tr>
<th>element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;selectFirstChoice&gt;</code></td>
<td>boolean</td>
<td>false</td>
<td>Indicates if the first item listed is the default item for the input. If a value for <code>&lt;default&gt;</code> is present, <code>&lt;selectFirstChoice&gt;</code> is ignored.</td>
</tr>
<tr>
<td><code>&lt;showClearButton&gt;</code></td>
<td>boolean</td>
<td>true</td>
<td>Indicates if the clear button for the dropdown is present. When present, the user clicks the clear button to change the choice to the default value for the dropdown.</td>
</tr>
</tbody>
</table>

**link child elements**

In addition to shared input child elements, you can use the following child elements in a `<link>`.

<table>
<thead>
<tr>
<th>element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;choice value=[value]&gt;</code></td>
<td>text</td>
<td></td>
<td>value: Required. Specifies the value to use for the choice. Specifies choices for the link input element. <code>&lt;choice&gt;</code> is the label to use for the specified value.</td>
</tr>
<tr>
<td><code>&lt;selectFirstChoice&gt;</code></td>
<td>boolean</td>
<td>false</td>
<td>Indicates if the first item listed is the default item for the input. Overrides any value for <code>&lt;initialValue&gt;</code>. If a value for <code>&lt;default&gt;</code> is present, <code>&lt;selectFirstChoice&gt;</code> is ignored.</td>
</tr>
</tbody>
</table>

**multiselect child elements**

In addition to shared input child elements, you can use the following child elements in a `<multiselect>`.

<table>
<thead>
<tr>
<th>element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;allowCustomValues&gt;</td>
<td>boolean</td>
<td>false</td>
<td>If true, enables the selection of custom values typed into the text field for the input.</td>
</tr>
<tr>
<td>&lt;delimiter&gt;</td>
<td>text</td>
<td></td>
<td>A string that will be placed between each selected value. Typically, you specify &quot; OR &quot; or &quot; AND &quot; using upper case – do not specify the quote marks, but specify a space character before and after the text.</td>
</tr>
<tr>
<td>&lt;valuePrefix&gt;</td>
<td>text</td>
<td></td>
<td>String prefixed to the value of the input element. Can be a regular expression.</td>
</tr>
<tr>
<td>&lt;valueSuffix&gt;</td>
<td>text</td>
<td></td>
<td>String appended to the value of the input element. Can be a regular expression.</td>
</tr>
</tbody>
</table>

**radio child elements**

In addition to shared input child elements, you can use the following child elements in a `<radio>`.

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;choice value=...&gt;</td>
<td>text</td>
<td></td>
<td>value Required. Represents a radio button choice value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies choices for a radio or dropdown element. <code>&lt;choice&gt;</code> is the label for the specified value.</td>
</tr>
<tr>
<td>&lt;selectFirstChoice&gt;</td>
<td>boolean</td>
<td>false</td>
<td>Indicates if the first item listed is the default selected item for the input. If a value for <code>&lt;default&gt;</code> is present, <code>&lt;selectFirstChoice&gt;</code> is ignored.</td>
</tr>
</tbody>
</table>

**text child elements**

See shared input child elements for available elements and exceptions.
time child elements

See shared input child elements for available elements and exceptions.

change (form input)

Use a `<change>` element to define how a form responds to user input. When a user makes a selection or types in an input, the actions configured in the `<change>` element for that input happen.

The `<change>` element is available in the following `<input>` types.

- `<checkbox>`
- `<dropdown>`
- `<radio>`
- `<text>`
- `<time>`

Use tokens to define responses to user input

Inside a `<change>` element, you can define actions such as token setting or other token updates that help you manage dynamic behavior or display in the form.

Conditional responses

Use one or more `<condition>` elements inside a `<change>` element to define conditional responses to specific user input. See `<condition>` for more information.

Element hierarchy

```xml
<input>
  <change>
    ( <set> | <unset> | <link> | <eval> ) (1..n)
    <condition> (0..n)
      ( <set> | <unset> | <link> | <eval> ) (1..n)
  </change>
</input>
```

Attributes

None
Child elements

You can use the following child elements in a `<change>` element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;set&gt;</code></td>
<td>Set or update a token value.</td>
</tr>
<tr>
<td><code>&lt;unset&gt;</code></td>
<td>Unset a previously set token value.</td>
</tr>
<tr>
<td><code>&lt;link&gt;</code></td>
<td>Link to a target destination, such as another dashboard, form, or an external website.</td>
</tr>
<tr>
<td><code>&lt;eval&gt;</code></td>
<td>Filter or format a token value.</td>
</tr>
<tr>
<td><code>&lt;condition&gt;</code></td>
<td>Use <code>&lt;condition&gt;</code> elements within a <code>&lt;change&gt;</code> element to define conditional responses to user input.</td>
</tr>
</tbody>
</table>

Example

Use the `<change>` element to capture the selected label and value from an input.

```xml
<form>
  <label>Use tokens with input choices to capture input labels and values</label>
  <init>
    <set token="date_label">Last 7 Days</set>
    <set token="earliest_tok">-7d@d</set>
  </init>
  <fieldset submitButton="false">
    <input type="radio" token="period_tok">
    <label>Select a time range</label>
    <choice value="-24h@d">Last 24 Hours</choice>
    <choice value="-7d@d">Last 7 Days</choice>
    <choice value="-30d@d">Last 30 Days</choice>
    <default>Last 7 Days</default>
    <change>
      <!-- use predefined input tokens to set -->
      <!-- tokens for the selected label and value -->
      <set token="date_label">$label$</set>
      <set token="earliest_tok">$value$</set>
    </change>
  </input>
</fieldset>

</form>
```

249
condition (form input)

Use a `<condition>` inside a form input `<change>` element to define a conditional response to user input.

The `<condition>` element is not available for multiselect inputs.

For information on using a `<condition>` element as part of a drilldown, see `<condition>` (drilldown).

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

250
Specifies the input label element to which the condition applies.

* applies the condition to all input <label> elements.

Specifies a condition to evaluate for a match. For example, you can use <condition match="'job.resultCount' == 0"> to specify a condition to apply when a search returns no results.

Specifies the input value element to which the condition applies.

* applies the condition to all input <value> elements.

Child elements

Use the following child elements in a <condition> to define conditional token setting or other behavior. See eval, link, set, and unset for more details.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;set&gt;</td>
<td>Set a token value.</td>
</tr>
<tr>
<td>&lt;unset&gt;</td>
<td>Unset a previously set token value.</td>
</tr>
<tr>
<td>&lt;link&gt;</td>
<td>Link to a target destination, such as another dashboard, form, or an external website.</td>
</tr>
<tr>
<td>&lt;eval&gt;</td>
<td>Process or format a token value.</td>
</tr>
</tbody>
</table>

Element hierarchy

```
<change>
  <condition>
    ( <set> | <unset> | <link> | <eval> ) (1..n)
```

Example

Use conditional inputs to select preset time ranges for a search.

The token for the selected choice appears in the title for the chart. The conditional token for the selected value drives the data for the chart.
<form>
 <label>Use tokens with conditional input choices</label>
 <fieldset submitButton="false">
  <input type="radio" token="period_tok">
  <label>Select a time range</label>
  <choice value="-24h@h">Last 24 Hours</choice>
  <choice value="-7d@h">Last 7 Days</choice>
  <choice value="-30d@h">Last 30 Days</choice>
  <default>Last 7 Days</default>

  <!-- set condition based on the label defined by <choice> -->
  <!-- Within each condition, specify a custom label for display -->
  <!-- Capture the selected value in the token, earliest_tok -->
  <change>
   <condition label="Last 24 Hours">
    <set token="date_label">Yesterday</set>
    <set token="earliest_tok">$value$</set>
   </condition>
   <condition label="Last 7 Days">
    <set token="date_label">Last week</set>
    <set token="earliest_tok">$value$</set>
   </condition>
   <condition label="Last 30 Days">
    <set token="date_label">Last month</set>
    <set token="earliest_tok">$value$</set>
   </condition>
  </change>
 </fieldset>
 <row>
  <panel>
   <title>Conditional Inputs</title>
   <chart>

    <!-- Display selected label in the title -->
    <title>$date_label$</title>

   <search>
    <query>index = _internal | timechart count by sourcetype</query>
     <!-- use the value of earliest_tok -->
     <!-- to set the time range -->
     <earliest>$earliest_tok$</earliest>
     <latest>now</latest>
   </search>

   <option name="charting.axisY.scale">log</option>
   <option name="charting.axisTitleX.text">Time periods</option>
   <option name="charting.axisTitleY.text">Events</option>

  </chart>
 </panel>
</row>
</form>
**row**

A row contains one or more <panel> child elements in a dashboard or form. Use a row to configure panel grouping and layout.

**Attributes**

Use the following attributes within the <row> element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>depends</td>
<td>Comma-separated list of tokens</td>
<td></td>
<td>All tokens from the list of tokens must be defined to render this row in a dashboard.</td>
</tr>
<tr>
<td>rejects</td>
<td>Comma-separated list of tokens</td>
<td></td>
<td>All tokens from this list must be defined to prevent this row from rendering in a dashboard.</td>
</tr>
<tr>
<td>id</td>
<td>text</td>
<td></td>
<td>Row identifier. Only alphanumeric and underscore characters are valid. The id cannot begin with a number or the underscore character. The following terms are reserved for internal use and cannot be used for an id.</td>
</tr>
</tbody>
</table>
A `<row>` normally contains one or more `<panel>` elements. It’s also possible to have `<row>` that contains an `<html>` element. However a single `<row>` cannot contain both a `<panel>` and an `<html>`.

**Panel**

Inside a `<row>`, use a `<panel>` to contain one or more visualizations. You can use panels to manage visualization grouping and layout. You can also show or hide panels dynamically.

**Panel alignment**

Two or more visualization elements in a panel align vertically, except for single value visualizations. Two or more single value visualizations in the same panel are grouped horizontally.

**Panel types**

There are two available `<panel>` types in Simple XML.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Requirements</th>
<th>Editing options</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline</td>
<td>Contains one or more visualization elements.</td>
<td>N/A</td>
<td>Create and update an inline panel using the dashboard editing user interface or Simple XML source code.</td>
<td>N/A</td>
</tr>
<tr>
<td>Reference</td>
<td>Shows the contents of a prebuilt panel</td>
<td>Requires a <code>ref</code> attribute to indicate the prebuilt panel to</td>
<td>Reference panels cannot be edited using the panel</td>
<td></td>
</tr>
</tbody>
</table>
display.

Optionally, add an app reference.

editor.

Reference panels do not recognize <panel> child elements specified in Simple XML.

For more information on prebuilt panels, see Dashboard panels and Create and add a panel by reference.

Element hierarchy

**Inline panel**

```xml
<row>
    <panel> (0..n)
    <title> (0..1)
    <search> (0..n)
        <chart> | <event> | <html> | <map> | <single> | <table> | <viz> (1..n)
    </search>
</row>
```

**Reference panel**

```xml
<row>
    <panel ref="[panel name]" [app="[app name]"]> (0..n)
        <!-- Other <panel> child elements ignored -->
</row>
```

**Attributes**

Use the following attributes within the `<panel>` element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref</td>
<td>Text</td>
<td></td>
<td>Required for reference panels only. References the name of a prebuilt panel. Prebuilt panel names appear in Settings &gt; User Interface &gt; Panels.</td>
</tr>
<tr>
<td>app</td>
<td>Text</td>
<td></td>
<td>Optional for reference panels. Defaults to the app containing</td>
</tr>
</tbody>
</table>
the dashboard. References the name of the app containing the prebuilt panel. Panel app names appear in Settings > User Interface > Panels.

| depends | Comma-separated list of tokens | All tokens in this list must be defined to render this panel in a dashboard. |
| id | text | Panel identifier. Only alphanumeric and underscore characters are valid. The id cannot begin with a number or the underscore character. The following terms are reserved for internal use and cannot be used for an id. |
| rejects | Comma-separated list of tokens. | All tokens from this list must be defined to prevent this panel from rendering in a dashboard. |

Child elements

Use `<panel>` child elements to configure inline panels. Reference panels ignore `<panel>` child elements.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| `<title>` | Optional `<title>` panel title. A panel title can provide context for a group of visualizations. Each visualization element in the panel | `<form>`
|          |             | `<row>`
|          |             | `<panel>`<title>Video game` |
can also have its own title.

A panel can contain one or more of the following visualization types.
<chart> | <event> | <html> | <map> | <single> | <table> | <viz>

Panel example

This <panel> element shows single value visualizations and chart groupings.

<dashboard>
<label>Dashboard Panel Example</label>
<description></description>
<row>
<panel>
<chart>
<title>Chart grouping</title>
<search>
$query
index=_internal source="*splunkd.log"
( log_level=ERROR OR log_level=WARN OR log_level=FATAL OR log_level=CRITICAL )
| stats count as log_events
| rangemap field=log_events low=1-100 elevated=101-300
</query>
</search>
</chart>
</panel>
</row>
</dashboard>

Base search for a panel. Use a base panel <search> element with a post--process search in a panel visualization.

<search id="myBaseSearch">
$query index=sales ...</query>
<row>
<panel>
<title>Game sales</title>
<table>
<search base="myBaseSearch">
$query search action!=NONE | chart ...</query>
</search>
</table>
</panel>
</row>
</form>
</dashboard>
default=severe
</query>
<earliest>-7d@h</earliest>
/latest>now</latest>
</search>
<option name="charting.chart">radialGauge</option>
</chart>
<chart>
<search>
<query>
index=_internal source="*splunkd.log"
( log_level=ERROR OR log_level=WARN*
 OR log_level=FATAL OR log_level=Cритical )
| stats count as log_events
| rangemap field=log_events low=1-100 elevated=101-300
default=severe
</query>
<earliest>-7d@h</earliest>
/latest>now</latest>
</search>
<option name="charting.chart">markerGauge</option>
</chart>
</panel>
</row>
<row>
<panel>
<single>
<title>Single value grouping</title>
<search>
<query>
index=_internal source="*splunkd.log"
( log_level=ERROR OR log_level=WARN*
 OR log_level=FATAL OR log_level=Cритical )
| stats count as log_events
| rangemap field=log_events low=1-100 elevated=101-300
default=severe
</query>
<earliest>-7d@h</earliest>
/latest>now</latest>
</search>
<option name="beforeLabel">Found</option>
<option name="afterLabel">errors</option>
</single>
<single>
<search>
<query>
index=_internal source="*splunkd.log"
( log_level=ERROR OR log_level=WARN*
 OR log_level=FATAL OR log_level=Cритical )
| stats count as log_events
| rangemap field=log_events low=1-100 elevated=101-300
default=severe
</query>
<earliest>-7d@h</earliest>
/latest>now</latest>
</search>
<option name="beforeLabel">Found</option>
<option name="afterLabel">errors</option>
</single>
<search>
<query>
index=_internal source="*splunkd.log"
( log_level=ERROR OR log_level=WARN*
 OR log_level=FATAL OR log_level=Cритical )
| stats count as log_events
| rangemap field=log_events low=1-100 elevated=101-300
default=severe
</query>
<earliest>-7d@h</earliest>
/latest>now</latest>
</search>
<option name="beforeLabel">Found</option>
<option name="afterLabel">errors</option>
</single>
Working with visualization elements

Use one or more of the following elements to add a visualization to a `<panel>`.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;event&gt;</code></td>
<td>Event list</td>
</tr>
<tr>
<td><code>&lt;table&gt;</code></td>
<td>Statistics table</td>
</tr>
<tr>
<td><code>&lt;chart&gt;</code></td>
<td>Line, bar, area, bubble, and scatter charts</td>
</tr>
<tr>
<td><code>&lt;map&gt;</code></td>
<td>Cluster maps and Choropleth maps</td>
</tr>
<tr>
<td><code>&lt;single&gt;</code></td>
<td>Single value visualization</td>
</tr>
<tr>
<td><code>&lt;html&gt;</code></td>
<td>Custom html content</td>
</tr>
<tr>
<td><code>&lt;viz&gt;</code></td>
<td>Custom visualization. See Custom visualizations in Simple XML in <em>Developing Views and Apps for Splunk Web.</em></td>
</tr>
</tbody>
</table>

Using visualization attributes and options

Like other Simple XML elements, visualizations use attributes for identification and for managing display. There are several shared attributes that you can use in any visualization type. Some visualizations have additional attributes available.

Each visualization has several available `<option>` named properties for configuration. Shared options can be used in all visualizations. Other `<option>` properties are specific to one visualization type.

**Option syntax**

Use the following option tagging syntax within the visualization element.

```
<option name="[option_name]">[option_value]</option>
```

For example, use the following option to show the Export button in a panel.
### Shared attributes

Use the following attributes to identify and control display for any visualization element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>depends</td>
<td>Comma-separated list of tokens that must be set for the panel, row, or visualization to display.</td>
<td>All tokens from the list of tokens must be defined to render this visualization. For example, <code>&lt;chart depends=&quot;$show1$&quot;&gt;</code> indicates that the chart is shown only when the <code>$show1$</code> token is set.</td>
</tr>
<tr>
<td>rejects</td>
<td>Comma-separated list of tokens that must be set for this visualization to be hidden.</td>
<td>All tokens from this list must be defined to prevent this visualization from rendering. For example, <code>&lt;table rejects=&quot;$one$, $two$&quot;&gt;</code> indicates that the table is hidden when either the <code>$one$</code> or <code>$two$</code> token is set.</td>
</tr>
<tr>
<td>id</td>
<td>text</td>
<td>Visualization identifier. Only alphanumeric and underscore characters are valid. The <code>id</code> cannot begin with a number or the underscore character. The following terms are reserved for internal use and cannot be used for an <code>id</code>.</td>
</tr>
</tbody>
</table>
### Trellis layout options

Trellis layout is available for any visualization type except cluster maps and tables. For more information, see Use trellis layout to split visualizations.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trellis.enabled</td>
<td>Boolean</td>
<td>0</td>
<td>Enable or disable trellis layout. Defaults to 0 (disabled).</td>
</tr>
<tr>
<td>trellis.scales.shared</td>
<td>Boolean</td>
<td>1</td>
<td>Indicate whether to share scale for axes in line, area, column, and bar charts or value ranges in Choropleth map segments. Use one of the following values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1: Shared scales</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0: Independent scales</td>
</tr>
<tr>
<td>trellis.size</td>
<td>String</td>
<td>medium</td>
<td>Configure the visualization segment size. Segment size affects panel display density for the split visualization. Use one of the following values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• small</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• large</td>
</tr>
<tr>
<td>trellis.splitBy</td>
<td>Result field name</td>
<td>N/A</td>
<td>Indicate the search result field or aggregation name to use for splitting the visualization. Segments appear for each value in this field.</td>
</tr>
</tbody>
</table>
Search, Inspect, Refresh, and Export options

Use the following `<option>` settings to configure **Open in Search** behavior and manage other panel user interface elements.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>link.exportResults.visible</td>
<td>Boolean</td>
<td>Defaults to the value of link.visible</td>
<td>Show the Export button at the bottom of the panel.</td>
</tr>
<tr>
<td>link.inspectSearch.visible</td>
<td>Boolean</td>
<td>Defaults to the value of link.visible</td>
<td>Show the Inspect button at the bottom of the panel.</td>
</tr>
<tr>
<td>link.openPivot.visible</td>
<td>Boolean</td>
<td>Defaults to the value of link.visible</td>
<td>Show the Open in Pivot button at the bottom of the panel.</td>
</tr>
<tr>
<td>link.openSearch.search</td>
<td>search string</td>
<td>Defaults to the panel search</td>
<td>Specify a search to open when a user clicks the Open in Search button.</td>
</tr>
<tr>
<td>link.openSearch.searchEarliestTime</td>
<td>time modifier</td>
<td>Defaults to the panel search earliest time.</td>
<td>The earliest time to use for the alternative search specified by link.openSearch.search.</td>
</tr>
<tr>
<td>link.openSearch.searchLatestTime</td>
<td>time modifier</td>
<td>Defaults to the panel search latest time.</td>
<td>The latest time to use for the alternative search specified by link.openSearch.search.</td>
</tr>
<tr>
<td>link.openSearch.text</td>
<td>text</td>
<td>Open in Search</td>
<td>Specify a custom label for the Open in Search button.</td>
</tr>
<tr>
<td>link.openSearch.viewTarget</td>
<td>View name</td>
<td>search</td>
<td>Specify a target view to use when a user clicks the Open in Search button.</td>
</tr>
<tr>
<td>link.openSearch.visible</td>
<td>Boolean</td>
<td>Defaults to the value of link.visible</td>
<td>Show the Open in Search button at the bottom of the panel.</td>
</tr>
<tr>
<td>link.visible</td>
<td>Boolean</td>
<td>true</td>
<td>Show link buttons at the bottom of the panel.</td>
</tr>
<tr>
<td>refresh.time.visible</td>
<td>Boolean</td>
<td>true</td>
<td></td>
</tr>
</tbody>
</table>
Display the refresh time indicator in the panel.

Use the `<refresh>` property in the `<search>` element to configure refresh behavior.

**refresh.link.visible**

Boolean  
true

Display the refresh link in the panel.

Use the `<refresh>` property in the `<search>` element to configure refresh behavior.

**event**

Use the `event` element to add an events list to a `<panel>`.

**Attributes**

You can use any of the shared attributes for visualizations in an `<event>` element.

**Child elements**

You can use the following child element in an `<event>` list.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| `<fields>` | Comma separated list | Restrict search results to these fields. The order of | `<dashboard>`  
  `<label>Fields Example</label>`  
  `<row>`  
  `<panel>` |
the listed fields determines the order of fields in the event list.

```xml
<event>
  <search>
    <query>
      index=_internal | timechart count by sourcetype
    </query>
    <earliest>-7d@d</earliest>
    <latest>now</latest>
  </search>
  <fields>_time, splunkd, splunk_web_access, splunk_web_service</fields>
</event>
</panel>
</row>
</dashboard>

Options

In addition to the shared options for visualizations, you can use the following options to configure an event list.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Integer</td>
<td></td>
<td>The maximum number of rows to display.</td>
</tr>
<tr>
<td>list.drilldown</td>
<td>One of the following values.</td>
<td>full</td>
<td>Specify drilldown enabling for the events list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>full: Enables the entire entry for drilldown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>inner: Enables inner elements of the event listing for drilldown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>outer: Enables outer elements of the event listing for drilldown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>none:</td>
<td>Disables drilldown.</td>
</tr>
<tr>
<td>list.wrap</td>
<td>Boolean</td>
<td>true</td>
<td>Indicates whether to wrap the contents of the events list.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxLines</td>
<td>Integer</td>
<td></td>
<td>The maximum number of lines to display for each result/event.</td>
</tr>
<tr>
<td>raw.drilldown</td>
<td></td>
<td>full</td>
<td>Specify drilldown enabling in the raw event listing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>full</td>
<td>full: Enables the entire entry for drilldown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inner</td>
<td>inner: Enables inner elements of the event listing for drilldown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outer</td>
<td>outer: Enables outer elements of the event listing for drilldown.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>none</td>
<td>none: Disables drilldown.</td>
</tr>
<tr>
<td>rowNumbers</td>
<td>Boolean</td>
<td>false</td>
<td>Indicates whether to display row numbers.</td>
</tr>
<tr>
<td>showPager</td>
<td>Boolean</td>
<td>true</td>
<td>Toggle pagination on or off.</td>
</tr>
<tr>
<td>table.sortColumn</td>
<td>text</td>
<td></td>
<td>Specifies the column on which to sort for the table.</td>
</tr>
<tr>
<td>table.sortDirection</td>
<td></td>
<td>asc</td>
<td>Indicates the sort direction for items in the table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>asc</td>
<td>asc: Indicates the sort direction for items in the table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>desc</td>
<td>desc: Indicates the sort direction for items in the table.</td>
</tr>
<tr>
<td>table.drilldown</td>
<td></td>
<td>all</td>
<td>Indicates whether drilldown functionality is enabled for the table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all</td>
<td>all: Drilldown is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>none</td>
<td>none: Drilldown is disabled.</td>
</tr>
<tr>
<td>table.wrap</td>
<td>Boolean</td>
<td>true</td>
<td>Indicates whether text in the table wraps.</td>
</tr>
<tr>
<td>type</td>
<td></td>
<td>list</td>
<td>Indicates the format for displaying events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>list</td>
<td>list: Indicates the format for displaying events.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>drilldown</td>
<td>String</td>
<td>all</td>
<td>Enable or disable drilldown types.</td>
</tr>
<tr>
<td>entityName</td>
<td>String</td>
<td>events</td>
<td>Show events or results in the list.</td>
</tr>
<tr>
<td>segmentation</td>
<td>String</td>
<td>none</td>
<td>Specify which event elements are clickable for drilldown.</td>
</tr>
<tr>
<td>softWrap</td>
<td>Boolean</td>
<td></td>
<td>Configure event wrapping.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<dashboard>
  ...
  <row>
    <panel>
      <title>Events in the last 24 hours</title>
      <event>
        <search>
          <query>index=_internal</query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="count">20</option>
      </event>
    </panel>
  </row>
</dashboard>
```
The `<table>` element defines a statistics table visualization.

### Attributes

You can use any of the shared attributes in a `<table>`.

### Child elements

You can use the following child element in a `<table>`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;fields&gt;</code></td>
<td>Comma separated list</td>
<td>Restrict search results to these fields. The order of the listed fields</td>
<td><code>&lt;dashboard&gt;</code>&lt;br&gt;<code>&lt;label&gt;Fields Example&lt;/label&gt;</code>&lt;br&gt;<code>&lt;row&gt;</code>&lt;br&gt;<code>&lt;panel&gt;</code>&lt;br&gt;<code>&lt;table&gt;</code>&lt;br&gt;<code>&lt;search&gt;</code></td>
</tr>
</tbody>
</table>
determines table column order.

```xml
<query>
   index=_internal |
   timechart count by sourcetype
</query>
<earliest>-7d@d</earliest>
<latest>now</latest>
</search>
<fields>_time, splunkd, splunk_web_access, splunk_web_service</fields>
</table>
</panel>
</row>
</dashboard>

Options

In addition to the shared options for visualizations, you can use the following options to configure a table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>Integer</td>
<td>10</td>
<td>The maximum number of rows to display.</td>
</tr>
<tr>
<td>dataOverlayMode</td>
<td>One of the following values.</td>
<td>none</td>
<td>Indicates which type of overlay to display.</td>
</tr>
<tr>
<td>drilldown</td>
<td>One of the following values.</td>
<td>cell</td>
<td>Enables drilldown on row or cell level, or disables drilldown.</td>
</tr>
<tr>
<td>rowNumbers</td>
<td>Boolean</td>
<td>false</td>
<td>Toggle display of row numbers.</td>
</tr>
<tr>
<td>showPager</td>
<td>Boolean</td>
<td>true</td>
<td>Toggle pagination on or off.</td>
</tr>
<tr>
<td>totalsRow</td>
<td>Boolean</td>
<td>false</td>
<td>Set to true to add a column totals row to the table.</td>
</tr>
<tr>
<td>percentagesRow</td>
<td>Boolean</td>
<td>false</td>
<td>Set to true to add a column percentages row to the table.</td>
</tr>
<tr>
<td>wrap</td>
<td>Boolean</td>
<td>true</td>
<td>Enable wrapping of text in the results table.</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>---------------------------------------------</td>
</tr>
</tbody>
</table>

**Column formatting**

You can use color and other formatting in table columns to show context or highlight values. Table column formatting uses a different syntax from other visualization options. To learn about formatting table columns in Simple XML, see Table column Simple XML.

**Example**

Example of a table panel using an inline search, displaying five rows, and disabling row numbers:

```xml
<dashboard>
  <label>Dashboard with Table</label>
  <row>
    <panel>
      <table>
        <title>Top source types in the last 24 hours</title>
        <search>
          <query>
            index=_internal group=per_sourcetype_thruput | chart sum(kb) by series | sort -sum(kb)
          </query>
          <earliest>-24h</earliest>
          <latest>now</latest>
        </search>
        <option name="count">5</option>
        <option name="rowNumbers">0</option>
      </table>
    </panel>
  </row>
</dashboard>
```

**Table sparklines**

Table sparklines show recent behavior patterns for each table row. You can format a sparkline in Simple XML.
Single value sparklines work differently from table sparklines. See single value for more details.

Configuring sparkline formatting in Simple XML does not change sparkline appearance in PDF. Sparklines appear without additional formatting in PDF.

**Generating a sparkline**

To use the formatting options described here, your search must generate a sparkline in the visualization. To create a sparkline, use the `chart` or `stats` command `sparkline( )` function. See Add sparklines to search results in the Search Manual for more details.

**Format syntax and attributes**

The `<format>` element contains formatting rules for sparklines. Include the "sparkline" type attribute and the field that the sparkline is tracking.

```
<format type="sparkline" field="[field name]">
 ...
</format>
```

Put `<option>` child elements inside the `<format>` to configure sparkline appearance.

**Format element attributes**

Use the following attributes to configure the `<format>` element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td><strong>Required.</strong> Use &quot;sparkline&quot;.</td>
</tr>
</tbody>
</table>
| field  | Search result field that the sparkline is tracking | **Required.** Use the same field that the `sparkline( )` command in your search is using to generate the sparkline. For example, you might use this search to generate sparklines for the `trend` field.  
`... | chart count sparkline(count, 1h) as trend by sourcetype | ...`
|        |        | Use `trend` as the `field` value to format the sparklines.                    |

```
<format type="sparkline" field="trend">
 ...
</format>
```
Format options

Use <option> elements inside the <format> element to define sparkline appearance. Some options are common to any sparkline. Other options are specific to the sparkline chart type.

Common sparkline options

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chartRangeMax</td>
<td>Number</td>
<td>n/a</td>
<td>Specify an alternate maximum sparkline range value.</td>
</tr>
<tr>
<td>chartRangeMin</td>
<td>Number</td>
<td>n/a</td>
<td>Specify an alternate minimum sparkline range value.</td>
</tr>
<tr>
<td>height</td>
<td>CSS style</td>
<td>auto</td>
<td>Chart height. Use any valid CSS width. For example, 1.5em or 20px.</td>
</tr>
<tr>
<td>tooltipPrefix</td>
<td>text</td>
<td></td>
<td>Text to place before each field displayed in a tooltip.</td>
</tr>
<tr>
<td>tooltipSuffix</td>
<td>text</td>
<td></td>
<td>Text to append to each field displayed in a tooltip.</td>
</tr>
<tr>
<td>type</td>
<td>(bar</td>
<td>discrete</td>
<td>line)</td>
</tr>
</tbody>
</table>

Sparkline bar chart options

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>barSpacing</td>
<td>Number</td>
<td></td>
<td>Space between each bar, in pixels.</td>
</tr>
<tr>
<td>barWidth</td>
<td>Number</td>
<td></td>
<td>Width of each bar, in pixels.</td>
</tr>
<tr>
<td>colorMap</td>
<td>See description</td>
<td></td>
<td>Map field values to selected colors. For example, if you want all values of -2 to appear yellow, use colorMap: { '-2': '#ff0' }. You can also use a value array instead of a mapping to specify a color for each bar. For example if your chart has the values 1,3, and 1, you can use colorMap=[&quot;red&quot;, &quot;green&quot;, &quot;blue&quot;].</td>
</tr>
</tbody>
</table>
### Sparkline discrete chart options

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineColor</td>
<td>CSS style</td>
<td></td>
<td>Used by line and discrete charts to specify the color of the line drawn as a CSS values string.</td>
</tr>
<tr>
<td>lineHeight</td>
<td>Number</td>
<td>30% of graph height</td>
<td>Height of each line, in pixels.</td>
</tr>
<tr>
<td>thresholdColor</td>
<td>CSS color</td>
<td></td>
<td>CSS color to use in combination with thresholdValue.</td>
</tr>
<tr>
<td>thresholdValue</td>
<td>CSS color</td>
<td></td>
<td>Draw values less than this threshold using thresholdColor instead of lineColor.</td>
</tr>
</tbody>
</table>

### Sparkline line chart options

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fillColor</td>
<td>CSS color</td>
<td>false</td>
<td>Specify the color to fill the area under the graph as a CSS value. Set to false to disable fill.</td>
</tr>
<tr>
<td>highlightLineColor</td>
<td>CSS color</td>
<td>#f22</td>
<td>CSS color for the vertical line that appears through a value when moused over. Set to null to disable.</td>
</tr>
<tr>
<td>highlightSpotColor</td>
<td>CSS color</td>
<td>#f5f</td>
<td>Color for the spot that appears on a value when moused over. Set to null to disable.</td>
</tr>
<tr>
<td>lineColor</td>
<td>CSS style</td>
<td></td>
<td>Used by line and discrete charts to specify the color of the line drawn as a CSS values string.</td>
</tr>
<tr>
<td>lineWidth</td>
<td>Number</td>
<td>1</td>
<td>line width, In pixels.</td>
</tr>
<tr>
<td>maxSpotColor</td>
<td>CSS color</td>
<td></td>
<td>CSS color of the marker displayed for the maximum value.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>minSpotColor</td>
<td>CSS color</td>
<td>Set to false or an empty string to hide it. CSS color of the marker displayed for the minimum value.</td>
<td></td>
</tr>
<tr>
<td>normalRangeMax</td>
<td>Number (see description)</td>
<td>Use with <code>normalRangeMin</code> to define threshold values for a normal or expected value range. Bars appear in the sparkline to show which values are within and outside of this range.</td>
<td></td>
</tr>
<tr>
<td>normalRangeMin</td>
<td>Number</td>
<td>Use with <code>normalRangeMax</code> to define threshold values for a normal or expected value range. Bars appear in the sparkline to show which values are within and outside of this range.</td>
<td></td>
</tr>
<tr>
<td>spotColor</td>
<td>CSS color</td>
<td>CSS color of the final value marker. Set to false or an empty string to hide it.</td>
<td></td>
</tr>
<tr>
<td>spotRadius</td>
<td>Number 1.5</td>
<td>Radius, in pixels, of all spot markers.</td>
<td></td>
</tr>
<tr>
<td>valueSpots</td>
<td>See description</td>
<td>Points and colors for drawing value spots. Accepts a range. For example, to render green spots on all values less than 50 and red on values higher than 50, use <code>{:49: 'green, '50:: 'red'}.</code></td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>CSS style auto</td>
<td>Chart width. Specify any valid CSS width (for example, 1.5em or 20px).</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

This Simple XML source code generates a table with a bar chart sparkline.
chart

Use the `<chart>` element to add column, bar, line, area, bubble, or scatter charts to a `<panel>`. You can also use the `<chart>` element to add a gauge visualization to a `<panel>`. 
The search generating a chart can be an inline search or a saved report. A saved report can contain chart format settings. These settings are used by default, but you can use options and attributes in your Simple XML source code to override them.

For more information on saving reports, see Create and edit reports.

Attributes

Use any of the shared attributes for visualizations in a chart.

Options

In addition to the shared options for visualizations, you can use the following options for general chart configuration. See the Chart Configuration Reference for a complete list of chart options.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart</td>
<td>area</td>
<td>bar</td>
<td>column</td>
</tr>
<tr>
<td>charting.legend.placement</td>
<td></td>
<td>right</td>
<td></td>
</tr>
<tr>
<td></td>
<td>top</td>
<td>left</td>
<td>bottom</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>charting.&lt;option_name&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>height</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```xml
<form>
  . . .
  <row>
    <panel>
      <title>Chart example</title>
      <chart>
        <title>Top sourcetypes in the last week</title>
        <search>
          <query>index=_internal | timechart sum(kb) by series | head 5</query>
          <earliest>-4h@m</earliest>
          <latest>now</latest>
        </search>
        <option name="height">200</option>
        <option name="charting.chart">line</option>
        <option name="charting.axisY.scale">log</option>
        <option name="charting.chart.nullValueMode">connect</option>
      </chart>
    </panel>
  </row>
  . . .
</form>
```
**map**

Use the `<map>` element to add a Choropleth or cluster map to a `<panel>`.

**Attributes**

In addition to the shared attributes for visualizations, you can use the following attributes in a `<map>` element.

**General map options**

In addition to the shared options for visualizations, you can use the following options to configure Choropleth and cluster maps.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drilldown</td>
<td>all</td>
<td>none</td>
<td>all</td>
</tr>
<tr>
<td>mapping.fieldColors</td>
<td>One or more &lt;field&gt; : &lt;color_hexvalue&gt; mappings.</td>
<td></td>
<td>A comma-separated list of field hexadecimal color values (0xRRGGBB) for specific series.</td>
</tr>
<tr>
<td>mapping.map.center</td>
<td>(lat,long)</td>
<td></td>
<td>The initial center point of the map. Latitude values can range from -85 to 85, with values outside of this range being clipped. Longitude values can range from -180 to 180, with values outside of this range being wrapped to fall within it.</td>
</tr>
<tr>
<td>mapping.map.scrollZoom</td>
<td>Boolean</td>
<td>false</td>
<td>Indicates whether the map zooms when a user scrolls the map.</td>
</tr>
<tr>
<td>mapping.map.panning</td>
<td>Boolean</td>
<td>true</td>
<td>Indicates whether the map pans when dragged.</td>
</tr>
<tr>
<td>mapping.map.zoom</td>
<td>Number</td>
<td>2</td>
<td>The initial zoom level of the map.</td>
</tr>
<tr>
<td>mapping.map.fitBounds</td>
<td>(south-lat, west-long,</td>
<td></td>
<td>The initial bounds to fit within the map. Latitude values can range from -85 to 85, with values outside of this range being clipped. Longitude values can range from -180 to 180, with values outside of this range being wrapped to fall within it.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>north-lat, east-long</td>
<td></td>
<td>Outside of this range being clipped. Longitude values can range from -180 to 180, with values outside of this range being wrapped to fall within it. Values assigned to this property effectively override any values assigned to the center or zoom properties.</td>
<td></td>
</tr>
<tr>
<td>mapping.seriesColors</td>
<td>List</td>
<td>Default* Comma-separated list of hexadecimal color values (0xRRGGBB). The list indicates sampling colors for series with no specific fieldColors color assigned to them.</td>
<td></td>
</tr>
<tr>
<td>mapping.showTiles</td>
<td>boolean</td>
<td>true Specify whether to show or hide map tiles.</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.tileOpacity</td>
<td>text</td>
<td>1 Specifies the opacity of the tiles. Values can range from 0 (transparent) to 1 (opaque).</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.url</td>
<td>URL template</td>
<td>See description The URL to use for requesting tiles, based on the following template. http://(s).tile.openstreetmap.org/(z)/(x)/(y).png</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.subdomains</td>
<td>[string,. . .]</td>
<td>[a,b,c] List of subdomains over which to distribute tile requests. More subdomains allows more tiles to be requested simultaneously. See example below.</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.minZoom</td>
<td>Integer</td>
<td>0 The minimum zoom level of the tileset.</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.maxZoom</td>
<td>Integer</td>
<td>7 The maximum zoom level of the tileset. Use any non-negative integer to specify the maximum zoom level.</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.invertY</td>
<td>Boolean</td>
<td>False Whether to invert the y coordinate for tile requests. TMS servers use inverse y-axis numbering.</td>
<td></td>
</tr>
<tr>
<td>mapping.tileLayer.attribution</td>
<td>String</td>
<td>See description A copyright attribution to be displayed in the bottom right corner of the map. The default value: Map data (c) 2012 OpenStreetMap contributors.</td>
<td></td>
</tr>
<tr>
<td>mapping.type</td>
<td>marker</td>
<td>choropleth See description Map type to render. Defaults to marker (cluster map).</td>
<td></td>
</tr>
</tbody>
</table>
## Choropleth map options

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapping.choroplethLayer.colorBins</td>
<td>integer</td>
<td>5</td>
<td>Specifies the number of color bins to use.</td>
</tr>
<tr>
<td>mapping.choroplethLayer.colorMode</td>
<td>&quot;sequential&quot;</td>
<td>'auto'</td>
<td>Specifies the color mode to use for the choropleth shapes. Possible modes are 'sequential', 'divergent', or 'categorical'.</td>
</tr>
<tr>
<td>mapping.choroplethLayer.maximumColor</td>
<td>text</td>
<td>DB5800</td>
<td>Specifies the color to use for the highest value shapes.</td>
</tr>
<tr>
<td>mapping.choroplethLayer.minimumColor</td>
<td>text</td>
<td>2F2BA</td>
<td>Only used when the color mode is divergent. The color to use for the lowest value shapes.</td>
</tr>
<tr>
<td>mapping.choroplethLayer.neutralPoint</td>
<td>text</td>
<td>0</td>
<td>Only used when the color mode is divergent. The value where the color palette should switch from using the minimum color to the</td>
</tr>
</tbody>
</table>
### Cluster map options

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mapping.data.maxClusters</code></td>
<td>Integer</td>
<td>100</td>
<td>The maximum number of clusters to render in a cluster map.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Caution:</strong> Setting this option to a large number of clusters can significantly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>increase processing time.</td>
</tr>
</tbody>
</table>
mapping.markerLayer.markerOpacity Number 0.8 Opacity of cluster map markers. Values can range from 0 (transparent) to 1 (opaque).

mapping.markerLayer.markerMinSize Number 10 The minimum size of cluster map markers, in pixels.

mapping.markerLayer.markerMaxSize Number 50 The maximum size of cluster map markers, in pixels.

Examples

The following examples show you how to use different map configurations.

**mapping.data.maxClusters**

The following example sets the maximum number of clusters to 250.

```
<map>
  <option name="mapping.data.maxClusters">250</option>
</map>
```

**mapping.fieldColors and mapping.seriesColors**

The following example configures the "foo" and "bar" fields to be red (0xFF0000) and green (0x00FF00), respectively, and configures all other fields to be blue (0x0000FF).

```
<map>
  <option name="mapping.fieldColors">{foo:0xFF0000,bar:0x00FF00}</option>
  <option name="mapping.seriesColors">[0x0000FF]</option>
</map>
```
mapping.map.fitBounds example

The following example initializes the map view to a boundary around San Francisco.

```xml
<map>
  <option name="mapping.map.fitBounds">
    (37.5,-123,38,-122)
  </option>
</map>
```

mapping.tileLayer.*

The following example configures the client to request tiles from openstreetmap.org (this is the default configuration).

```xml
<map>
  <option name="mapping.tileLayer.url">http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png</option>
  <option name="mapping.tileLayer.subdomains">[a,b,c]</option>
  <option name="mapping.tileLayer.maxZoom">18</option>
  <option name="mapping.tileLayer.attribution">
    Map data (c) 2012 OpenStreetMap contributors, CC-BY-SA.
  </option>
</map>
```

map example using foursquare data

This example assumes you are indexing foursquare data as source foursquare.

```xml
<map>
  <title>Roma</title>
  <search>
    sourcetype=foursquare
    | geostats latfield=checkin.geolat longfield=checkin.geolong count by checkin.user.gender
  </search>
  <option name="mapping.data.maxClusters">500</option>
  <option name="mapping.markerLayer.markerMaxSize">20</option>
  <option name="mapping.map.fitBounds">(41.3,12.7,41.5,12.8)</option>
  <option name="mapping.seriesColors">[0x0060DD]</option>
  <option name="mapping.map.zoom">4</option>
</map>
```

Complete Choropleth map example
The `<single>` element defines a single value visualization in a `<panel>`. 
Attributes

You can use any of the shared attributes for visualizations in a `<single>` value element.

Options

In addition to the shared options for visualizations, you can use the following options to configure a single value visualization.

Use the following options to configure a `<single>` value visualization.

**Note:** Configurations made in the **Format** menu can override Simple XML settings.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorBy</td>
<td>One of the following values. value</td>
<td>Indicate whether to show color changes in the single value or the trend indicator. To use this option, the <code>useColors</code> option must be set to <code>true</code>. If you use value, the value range determines the color used. Color by trend is only available if the visualization search uses the <code>timechart</code> command. If you use trend, the indicator appears in black (no change), green (positive), or red (negative). Use the</td>
<td></td>
</tr>
<tr>
<td><strong>trendColorInterpretation</strong></td>
<td>Option to configure positive and negative value interpretation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| colorMode | One of the following values.  
block  
none | Specify whether the single value or the background shows the value range color.  
Use block to show color in the background with white text. Use none to show color in the single value with a white background. |
| drilldown | One of the following values.  
all  
none | Use all to enable drilldown on the single value. Use none to disable drilldown. |
| field | field name  
Defaults to the first field returned | Indicate which field's single value to display in the visualization. |
| numberPrecision | string  
0 | Specify how many decimal places to display. For decimal precision, indicate the number of places using 0, followed by up to four zeroes. For example, 0.0 or 0.00. |
| rangeColors | String array  
Defaults to standard range colors: red, orange, yellow, blue, and green.  
Use a string array to specify any number of hex color values. List the colors in an order corresponding to rangeValues values. The hex value format should be either 0xFFFFFF or FFFFFF.  
If there are more rangeColors values than ranges, excess color values |
are ignored. If there are more rangeValues values than rangeColors hex colors, the single value appears dark grey if it falls within a range without a color mapped to it.

<table>
<thead>
<tr>
<th><strong>rangeValues</strong></th>
<th><strong>Number array</strong></th>
<th><strong>none</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use an array of numbers to specify ranges for the single value. Ranges determine single value color if you are using color by value. Use the rangeColors option to specify colors that map to rangeValues values.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>showSparkline</strong></th>
<th><strong>boolean</strong></th>
<th><strong>true</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicate whether to show the single value sparkline, if one is available. A sparkline is only available for searches that include the timechart search command.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>showTrendIndicator</strong></th>
<th><strong>boolean</strong></th>
<th><strong>true</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Show or hide the single value trend indicator. A trend indicator is only available when the visualization query includes the timechart command.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>trendColorInterpretation</strong></th>
<th><strong>One of the following values.</strong></th>
<th><strong>standard</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether a field value greater than 0 is shown as a positive (standard) or negative (inverse) trend.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>trendDisplayMode</strong></th>
<th><strong>One of the following values.</strong></th>
<th><strong>absolute</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether the delta value is displayed as a percentage or an absolute count.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>trendInterval</td>
<td>percent</td>
<td>absolute</td>
</tr>
<tr>
<td>underLabel</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>useColors</td>
<td>Boolean</td>
<td>false</td>
</tr>
<tr>
<td>useThousandSeparators</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>unit</td>
<td>Brief text label</td>
<td>none</td>
</tr>
<tr>
<td>unitPosition</td>
<td>One of the following values.</td>
<td>after</td>
</tr>
</tbody>
</table>

**Deprecated options**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
<th>Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>additionalClass</td>
<td>CSS class name</td>
<td>CSS class name to add to the result container. Use <code>colorBy</code>, <code>rangeValues</code>, and <code>rangeColors</code> to configure color mapping for values.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>afterLabel</td>
<td>String</td>
<td>Visualization label Use <code>underLabel</code> to include a caption or unit to add a measurement unit to the single value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beforeLabel</td>
<td>String</td>
<td>Visualization label Use <code>underLabel</code> to include a caption or unit to add a measurement unit to the single value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>classField</td>
<td>String</td>
<td>Adds the value of the <code>classField</code> of the first result as an additional CSS class to the result container. Use <code>colorBy</code>, <code>rangeValues</code>, and <code>rangeColors</code> to configure color mapping for values.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>linkFields</td>
<td>String</td>
<td>Indicates the visualization element clickable for drilldown. The single value itself is clickable for drilldown. Use the <code>&lt;drilldown&gt;</code> element to configure drilldown.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
<td>------------------------------------------------------</td>
<td>------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>linkSearch</td>
<td>Search string</td>
<td>Search whose results are clickable for drilldown.</td>
<td>The single value itself is clickable for drilldown. Use the <code>&lt;drilldown&gt;</code> element to configure drilldown behavior.</td>
<td></td>
</tr>
<tr>
<td>linkView</td>
<td>Dashboard or form</td>
<td>Dashboard or form used with linkSearch search for drilldown.</td>
<td>Use the <code>&lt;drilldown&gt;</code> element to configure drilldown behavior, including linking to a dashboard or form.</td>
<td></td>
</tr>
<tr>
<td>refresh.auto.interval</td>
<td>integer</td>
<td>0</td>
<td>Specifies a visualization refresh interval. To disable panel refresh, specify 0 or a negative integer.</td>
<td>Configure dashboard or form refresh behavior using the <code>refresh</code> attribute or use the <code>&lt;search&gt;</code> element <code>&lt;refresh&gt;</code> child element to configure panel refresh behavior.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<dashboard>
  
  <panel depends="$show_single_value$">
    <title>Event count for $selected_sourcetype$</title>
    <single>
      
      
    </single>
  </panel>

  
  
</dashboard>`
Show inline HTML content.

Attributes

In addition to the shared attributes for visualizations, you can use the following attributes in an `<html>` element.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>encoded</td>
<td>boolean</td>
<td>false</td>
<td>For internal use only.</td>
</tr>
<tr>
<td>src</td>
<td>String</td>
<td></td>
<td>Put the contents of an HTML or image file into the <code>&lt;html&gt;</code> panel. Directory locations and Simple XML syntax are different for HTML files and image files.</td>
</tr>
</tbody>
</table>

**HTML file from the same app context**

```html
<html src="<file_name>.html">
</html>
```

**HTML file from a different app context**

```html
<html src="<other_app_name>:<file_name>.html">
</html>
```

**Image file**

```html
<img src="/static/app/<app_name>/images/<file_name>.png">
```
See the following instructions for details on where to save HTML and image files and how to reference files from different app contexts.

| tokens | boolean | true | If false, disables token replacement for the <html> panel. |

Use an HTML file in a dashboard panel

Steps

1. Put the HTML file in the following directory.
   `$SPLUNK_HOME/etc/apps/<appname>/appserver/static`

2. In the <html> panel, use this syntax to indicate a file from the current app context.
   ```html
   <html src="<file_name>.html">
   </html>
   ```

   If you are specifying an HTML file from another app context, use this syntax.
   ```html
   <html src="<other_app_name>:<file_name>.html">
   </html>
   ```

Use an image file in a dashboard panel

Steps

1. Put the image file in the following directory.
   `$SPLUNK_HOME/etc/apps/<appname>/appserver/static/images`
   
   If an /images directory does not already exist, create one and put the file in it.

2. Verify that the image file path is accessible by testing the following URL.
   `http://<host>:<port>/static/app/<app_name>/images/<image>`
   
   For example, use this URL to verify that the `my_image.png` file is accessible.
   `http://localhost:8000/static/app/search/images/my_image.png`
3. In the `<html>` panel, use this syntax to indicate a file from the current app context.

```html
<html>
<img src="/static/app/search/images/<file_name>.png">
</html>
```

**Options**

There are no child options for the `<html>` element.

**Example**

```html
<dashboard>
  <label>test_db</label>
  <row>
    <panel>
      <html>
      <!-- Use an image from the current app's /static/images directory -->
      <img src="/static/app/search/images/my_image.png"></img>
      </html>
    </panel>
    <panel>
      <!-- Use an HTML file from the webhook app. -->
      <html src="alert_webhook:my_html_file.html">
      </html>
      <!--Use an image from the webhook app static/images directory -->
      <html>
      <img src="/static/app/alert_webhook/images/my_other_image.png"></img>
      </html>
    </panel>
  </row>
</dashboard>
```

**search**

Use the `<search>` element to define a search in a dashboard, form, or panel. You can also use a `<search>` to define input choices dynamically.

**Search types**

The `<search>` element can define any of the following search types.

<table>
<thead>
<tr>
<th>Search type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inline</td>
<td></td>
</tr>
<tr>
<td><strong>Define a search directly in the dashboard, form, or panel. Use the <code>&lt;query&gt;</code> element to specify the search string. You can add time range modifiers and other options to configure the search.</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>Use a report search and visualization in a panel. Add the <code>ref</code> attribute and the report name to the <code>&lt;search&gt;</code> element.</strong> You can configure the visualization from the referenced report and add custom time range modifiers to the search. You cannot modify the search string in Simple XML. Edit the report to adjust the search string. When you edit a report search, panels with references to the report update automatically.</td>
<td></td>
</tr>
<tr>
<td><strong>Report reference</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Use <code>&lt;search&gt;</code> as a child element of a form input to populate choices for checkbox, dropdown, multiselect, or radio inputs. Indicate <code>&lt;fieldForLabel&gt;</code> and <code>&lt;fieldForValue&gt;</code> search result fields to populate the input choice labels and choice values.</strong> Do not use real-time searches for populating searches. Input choice labels and values do not update as real-time search results generate.</td>
<td></td>
</tr>
<tr>
<td><strong>Populating search for input choices</strong></td>
<td></td>
</tr>
<tr>
<td><strong>A base search generates transformed results for post-process searches to modify. A base search can be global, defined at the <code>&lt;dashboard&gt;</code> or <code>&lt;form&gt;</code> level. You can also use a base search at the <code>&lt;panel&gt;</code> level in Simple XML.</strong> Make sure that all base searches meet the following requirements.</td>
<td></td>
</tr>
<tr>
<td><strong>Base search</strong></td>
<td></td>
</tr>
</tbody>
</table>
| • Use transforming commands in the base search to generate transformed results. Non-transforming base searches can cause performance and timeout issues.  
• The base search must have an `<id>` attribute for post-process searches to reference. |
| **Post-process search** |
| A post-process search accepts transformed results from a base search and performs additional search processing on these results. Use the `base` and `id` attributes to reference the base search id for a post-process search. **Specify `<earliest>` and `<latest>` elements in the base search. A post-process search ignores time modifiers defined** |
within the post-process `<search>` element.

For more information on post-process searches, see Post-process searches in this manual.

**Element hierarchy**

You can put a `<search>` element into any of the following parent elements.

- `<form>`
- `<dashboard>`
- `<panel>`
- `<chart>`
- `<event>`
- `<map>`
- `<single>`
- `<table>`

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app</td>
<td>text</td>
<td></td>
<td>The name of an app. Use the <code>app</code> attribute with the <code>ref</code> attribute to reference a report that is not in the current app.</td>
</tr>
<tr>
<td>base</td>
<td>text</td>
<td></td>
<td>In a post-process search, references a base search in this dashboard, form, or panel. Use <code>base</code> with the <code>id</code> attribute to indicate the base search that the post-process search should use.</td>
</tr>
<tr>
<td>depends</td>
<td>List of one or more tokens</td>
<td></td>
<td>All of the tokens listed must be defined for the search to dispatch. If the search does not dispatch, the panel does not render.</td>
</tr>
<tr>
<td>id</td>
<td>Text (minimum two characters)</td>
<td></td>
<td>Search identifier. In a post-process search, reference the base search using this identifier. Make sure that search identifiers follow these requirements.</td>
</tr>
</tbody>
</table>

- Use only alphanumeric or underscore characters.
• The id cannot begin with a number or the underscore character.
• The following terms are reserved for internal use and cannot be used for an id.
  ◆ dashboard
  ◆ default
  ◆ footer
  ◆ header
  ◆ search
  ◆ submitted
  ◆ url

Reference to a report. Adds the report search and visualization to the dashboard or form.

If you are referencing a report in another app, use the app attribute to specify the app.

Stop the search from dispatching if one or more tokens in this list are defined. If the search does not dispatch, the panel does not render.

### Child elements

Use the following child elements within a `<search>`.

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<cache>` | For saved searches, use one of the following values. | scheduled | • true: Always use the results from a preexisting saved search job when possible.  
• false: Never use results from preexisting saved search jobs.  
• scheduled: Reuse any previously run scheduled saved search jobs.  
• [integer]: The number of seconds indicating the maximum saved search job |
results age. Only results that are newer than this number of seconds are used.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Required</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cancelled&gt;</td>
<td>N/A</td>
<td>N/A</td>
<td>Execute actions when a search is cancelled.</td>
</tr>
<tr>
<td>&lt;done&gt;</td>
<td>N/A</td>
<td>N/A</td>
<td>Execute actions based on finished search events. Includes job properties and first result row.</td>
</tr>
<tr>
<td>&lt;error&gt;</td>
<td>N/A</td>
<td>N/A</td>
<td>Execute actions when there is a search error event, such as an invalid query.</td>
</tr>
<tr>
<td>&lt;earliest&gt;</td>
<td>N/A</td>
<td>N/A</td>
<td>Optional time expressions that specify the earliest and latest time parameters for a search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Post-process searches ignore child <code>&lt;earliest&gt;</code> and <code>&lt;latest&gt;</code> elements. Instead, the <code>&lt;earliest&gt;</code> and <code>&lt;latest&gt;</code> elements from the base search are used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can specify the time as relative time or absolute time. For relative time, use relative time modifiers, as described in Specify relative time ranges in your search in the Search Manual. For absolute time, specify the time in UNIX epoch time format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note</strong>: UNIX epoch time format for absolute time in Simple XML is different from the SPL absolute time format used in queries.</td>
</tr>
<tr>
<td>&lt;progress&gt;</td>
<td></td>
<td></td>
<td>Execute an action on search progress events. Access job properties and the first results row.</td>
</tr>
<tr>
<td>&lt;query&gt;</td>
<td>text</td>
<td></td>
<td>Search string.</td>
</tr>
<tr>
<td><code>&lt;refresh&gt;</code></td>
<td>Integer or relative time expression</td>
<td>No refresh</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indicate a delay or interval time for inline or saved searches. This setting does not apply to post-process searches, which refresh automatically when their base search refreshes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integers are handled as seconds. Use SPL syntax for relative time expressions. For example, <code>1h5m</code> or <code>5m</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use the <code>&lt;refreshType&gt;</code> setting to specify refresh behavior in relation to search completion or dispatch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can use the <code>&lt;refresh.display&gt;</code> setting in a visualization to specify a refresh progress indicator.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>&lt;refreshType&gt;</code></th>
<th>interval or delay</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicate the starting time for counting down to a refresh. Use <code>delay</code> to start counting when the search is done.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use <code>interval</code> to count down when the search is dispatched. If the runtime of the search is longer than the configured time, the search job is cancelled and a new job is dispatched.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>&lt;sampleRatio&gt;</code></th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Event sampling ratio. To learn more, see Event sampling with reports and dashboard panels in the Search Manual.</td>
</tr>
</tbody>
</table>

**Examples**

**Base search from inline search**

```xml
<search id=[base ID]>
  <query>[search string]</query> (1)
  <earliest> (0..1)
```
Base search from report

<search id=[base ID] [ref=[report name]]>
  <earliest> (0..1)
  <latest> (0..1)
</search>

Post-process search

<search base=[base ID]> (0..n)
  <query>[post-process search string]</query> (1)
</search>

Dashboard with base search and two post-process searches

<dashboard>
  <label>Dashboard with post-process search</label>
  <description></description>
  <!-- Example uses stats transforming command -->
  <!-- This limits events passed to post-process search -->
  <search id="baseSearch">
    <query>
      index=_internal source=*splunkd.log | stats count by component, log_level
    </query>
    <earliest>-30d</earliest>
    <latest>now</latest>
  </search>
  <row>
    <panel>
      <chart>
        <title>Post-process 1: Event count by log level</title>
        <!-- post-process search -->
        <search base="baseSearch">
          <query>
            stats sum(count) AS count by log_level
          </query>
        </search>
      </chart>
    </panel>
    <panel>
      <chart>
        <title>Post-process 2: Error count by component</title>
        <!-- post-process search -->
        <search base="baseSearch">
          <query>
            search log_level=error | stats sum(count) AS count by
          </query>
        </search>
      </chart>
    </panel>
  </row>
</dashboard>
Dashboard with empty post-process search

<dashboard>
  <label>Dashboard with empty post-process search</label>
  <description>Example uses stats transforming command -->
  <!-- This limits events passed to post-process search -->
  <search id="baseSearch">
    <query>index=_internal source=*splunkd.log | stats count by component, log_level</query>
    <earliest>-30d</earliest>
    <latest>now</latest>
  </search>
  <row>
    <panel>
      <chart>
        <title>Count by component, log level (from post-process search)</title>
        <!-- post-process search -->
        <search base="baseSearch">
          <query>stats sum(count) AS count by log_level</query>
        </search>
        <option name="charting.axisY.scale">log</option>
      </chart>
    </panel>
    <panel>
      <chart>
        <title>Count by component (from base search)</title>
        <!-- empty post-process search -->
        <search base="baseSearch" />
        <option name="charting.chart">bar</option>
      </chart>
    </panel>
  </row>
</dashboard>
selection (area, column, and line charts)

Use a `<selection>` element in area, column, or line charts to configure pan and zoom time window parameters or to set other token values.

See Chart controls for more details on working with chart pan and zoom.

Parent elements

```
<chart>
<option name="charting.chart">area</option> | <option
name="charting.chart">column</option> | <option
name="charting.chart">line</option>
```

Use predefined tokens to capture the earliest and latest time of the time window and the earliest and latest values within that time window for a field.

Example

```
<selection>
  <set token="selection.earliest">$start$</set>
  <set token="selection.latest">$end$</set>
  <set token="start.\[fieldname\]">$start.\[fieldname\]$</set>
  <set token="end.\[fieldname\]">$end.\[fieldname\]$</set>
</selection>
```

You can also `<link>` to a target dashboard, form, or external website in a `<selection>` element.
Attributes

None

Example

A selection on the left chart zooms into the right chart with details for the selected area.

<dashboard>
  <label>Pan and Zoom</label>
  <row>
    <panel>
      <chart>
        <title>Pan and Zoom (All source types)</title>
        <search>
          <query>
            index=_internal | timechart count by sourcetype
          </query>
          <earliest>-7d@h</earliest>
          <latest>now</latest>
        </search>
        <option name="charting.axisX.scale">linear</option>
        <option name="charting.axisY.scale">log</option>
        <option name="charting.chart">line</option>
        <selection>
          <set token="selection.earliest">$start$</set>
          <set token="selection.latest">$end$</set>
          <set token="start.splunk_web_access">$start.splunk_web_access$</set>
          <set token="end.splunk_web_access">$end.splunk_web_access$</set>
        </selection>
        <option name="charting.axisTitleX.text">Last 7 Days</option>
      </chart>
    </panel>
    <panel>
      <chart>
        <title>Pan and Zoom (Web access source type)</title>
        <search>
          <query>
            index=_internal sourcetype=splunk_web_access | timechart count by sourcetype
          </query>
          <earliest>$selection.earliest$</earliest>
          <latest>$selection.latest$</latest>
        </search>
        <option name="charting.chart">column</option>
        <option name="charting.legend.placement">none</option>
      </chart>
    </panel>
  </row>
</dashboard>
<option name="charting.legend.masterLegend">null</option>
<option name="charting.axisX.scale">linear</option>
<option name="charting.axisY.scale">log</option>
<option name="charting.axisTitleX.text">Selected Time Range</option>
</chart>
</row>
<row>
<p><b>Token values for the splunk_web_access selection</b></p>
<table border="0" cellpadding="12" cellspacing="0">
<tr>
<td>
<p><b>Time range (epoch time)</b></p>
<p><b>$$selection.earliest$$</b>:
$$selection.earliest$$
<b>$$selection.latest$$</b>:
$$selection.latest$$</p>
</td>
<td>
<p><b>Count at the begining and end of time range.</b></p>
<p><b>$$start.splunk_web_access$$</b>:
$$start.splunk_web_access$$
<b>$$end.splunk_web_access$$</b>:
$$end.splunk_web_access$$</p>
</td>
</tr>
</table>
</html>
</panel>
</row>
</dashboard>
**drilldown**

Drilldown lets you build dynamic responses to user input or interactions. Use the following elements to configure drilldown behavior.

See Use drilldown for dashboard interactivity for details on using drilldown in dashboards and forms.

Use the `<drilldown>` element to start implementing drilldown behavior. The `<drilldown>` element contains child elements that define the response to a user click.

**Drilldown actions**

Inside a `<drilldown>` element, use `<set>`, `<unset>`, or `<eval>` to update token values as part of building dynamic behavior or display in the current dashboard or form. Use `<link>` to open a search, another dashboard, or an external website. You can pass token values to a linked target on a user click.

**Drilldown conditions**

Define conditional responses to different user actions by adding a `<condition>` element within the `<drilldown>`.

**Use actions or conditions as drilldown child elements**

You cannot combine actions and conditions as direct child elements of `<drilldown>`. If you are defining conditional drilldown behavior, you cannot include action elements directly inside the `<drilldown>` element. If you are defining actions directly inside the `<drilldown>` element, you cannot add `<condition>` elements directly inside the drilldown.

**Attributes**

None

**Element hierarchy**

Put a `<drilldown>` element inside any of the following visualization parent elements.

- `<chart>`
- `<event>`
- `<map>`
- `<single>`
- `<table>`
Child elements

Use the following child elements inside a `<drilldown>` to configure the actions that happen when users click on the element where drilldown is enabled.

```
<drilldown>
  ( <link> | <set> | <unset> ) (1..n) | <condition> (1..n)
</drilldown>
```

Examples

Pass a value to a form

```
<table>
<search>index=_internal</search>

<!-- Pass the clicked row's 'count'-column value to populate a destination form's 'foo' token. -->
<drilldown>
  <link>
    /app/search/simple_xml_form?form.foo=$row.count$
  </link>
</drilldown>
</table>
```

Pass parameters to a form

```
<table>
<search>index=_internal</search>

<!-- Pass the clicked cell's value, earliest time, and latest time to a destination form's token ('foo') and search parameters -->
<drilldown>
  <link>
    <![CDATA[
      /app/search/simple_xml_form?form.foo=$click.value2$&earliest=$earliest$&latest=$latest$
    ]]>  
  </link>
</drilldown>
</table>
```

Pass a value from a chart to a website

```
<chart>
  <search>
  
```

304
index=_internal | chart count by sourcetype
</search>
<option name="charting.chart">column</option>

<!--[-- $click.value$ captures the value clicked by the user -->
<!--[-- From the x-axis of a column chart and passes -->
<!--[-- it to the website as a query parameter -->
<drilldown>
  <link>
    http://splunkbase.splunk.com/integrated_search/?q=$click.value$
  </link>
</drilldown>
</chart>

**condition (drilldown)**

Define conditional responses to user clicks on specific fields in the dashboard or form. If the `<condition>` element is not present, drilldown actions apply to clicks on all fields.

See `<condition>` (input) for details on using a `<condition>` in a form input.

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>text</td>
<td>*</td>
<td>Specifies the search field on which to implement the drilldown, or to set or unset a token.</td>
</tr>
</tbody>
</table>

**Child elements**

See eval, link, set, and unset.

**Example**

See the example for `<set>` for using the `<condition>` tag to set a token for in-page drilldown.

**eval, link, set, and unset**

**Managing token values**

Update token values to create dynamic display and behavior in dashboards and forms. Use `eval, link, set, and unset` with `<drilldown>` or `<change>` elements to define the response to user interactions.
**Element hierarchy**

```xml
<input>
  <change>
    ( <set> | <unset> | <link> | <eval> ) (1..n)
    <condition> (0..n)
      ( <set> | <unset> | <link> | <eval> ) (1..n)
  </change>
</input>

<drilldown>
  ( <set> | <unset> | <link> | <eval> ) (1..n)
  <condition> (0..n)
    ( <set> | <unset> | <link> | <eval> ) (1..n)
</drilldown>
```

**eval**

Filter or format token values. See Custom logic for dashboard eval expressions for more information.

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>text</td>
<td>None</td>
<td>Token whose value is generated by the <code>&lt;eval&gt;</code> expression. In an <code>&lt;eval&gt;</code> expression, you can use either <code>$...$</code> delimiters or single quote delimiters for tokens. For example, both of the following options are valid.</td>
</tr>
</tbody>
</table>

- `$my_token$`
- `'my_token'`

**Example**

This example uses `<eval>` with the `<progress>` search event handler to compute and display job duration in the dashboard.

```xml
<dashboard stylesheet="eval_tokens.css">
  <label>Eval Tokens</label>
  <row>
    <panel>
      <title></title>
      <search id="search_logic">
        <query>index=_internal | top sourcetype</query>
      </search>
    </panel>
  </row>
</dashboard>
```
<earliest>0</earliest>
<latest>now</latest>
<progress>
<eval
token="duration">tostring(tonumber($job.runDuration$),"duration")</eval>
</progress>
</search>
<chart>
<title>Top sourcetypes for index=_internal</title>
<search base="search_logic" />
<option name="charting.chart">bar</option>
</chart>
<html>
<h3>Duration</h3>
<div class="custom-result-value">$duration$</div>
</html>
</panel>
</row>
</dashboard>

**link**

Link to a dashboard, form, or external website in response to a drilldown click or form input.

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>Text</td>
<td></td>
<td>Indicate the browser window where the drilldown target opens. Corresponds to the <code>target</code> attribute of the <code>&lt;a&gt;</code> HTTP tag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• &quot;_blank&quot;: Open the target in a new window.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• &quot;_self&quot;: Open the target in the current window.</td>
</tr>
</tbody>
</table>

307
• `<optional_string>`: Use an arbitrary string to open the target in a new window. Each subsequent time that a user click invokes this drilldown, the target opens in the same window.

### Deprecated attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Alternatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>Field name</td>
<td>(&lt;drilldown&gt; only) Specifies which values to capture in a table from the specified column or row. Cannot be specified together with the <code>series</code> attribute.</td>
<td>Specify fields with the <code>&lt;condition&gt;</code> tag.</td>
</tr>
<tr>
<td>series</td>
<td>Series name</td>
<td>(&lt;drilldown&gt; only) Specifies which values to capture in a chart from the specified series. Cannot be specified together with the <code>field</code> attribute.</td>
<td>Specify series with the <code>&lt;condition&gt;</code> tag.</td>
</tr>
</tbody>
</table>

### Link path syntax

The `<link>` element contains a path to the target and any token values that you are passing from the source to the target. Use one of the following syntax options.

<table>
<thead>
<tr>
<th>Target and behavior</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| Link to a dashboard in your Splunk deployment. | Use a relative path that includes the dashboard or form id.  
  `<link>`  
  `[relative path]/[dashboard or form id]`  
  `</link>` |
| Link to a form in your Splunk deployment. Show customized content in the form by passing a token value captured from the source. Use | Add a `?` symbol after the relative path. Set tokens in the target to values passed from the source. This example sets a token in a target form to a value from the source.  
  `<link>`  
  `[relative path]/[dashboard or form id]?form.[target_token_name]='][$source_value'`  
  `</link>` |
the token value to populate a form input.

Pass the `<earliest>` and `<latest>` time range modifiers from the source search to a search in the target.

Add `&earliest=$earliest$&latest=$latest$` to the target path and token values. Use `<![CDATA[ ... ]]>` wrapper to make sure that the `&` symbol is interpreted correctly.

```
<link><![CDATA[
[relative path]/[dashboard or form id]?form.[target_token_name]=[$source_value$]&earliest=$earliest$&latest=$latest$]]>
</link>
```

Use a URL and query argument to pass a value to a target web page.

```
<link>[target_URL]?q=$source_value$</link>
```

See also Token usage in dashboards to review available token filters.

**Example**

Use `<link>` with conditional inputs to open a new page.

```
<form>
  . . .
  <fieldset>
    <input type="dropdown" token="openNewPageToken">
      <label></label>
      <default>Select a page to open</default>
      <choice value="">Select a page to open</choice>
      <choice value="manager_page">Buttercup Games dashboard</choice>
      <choice value="splk_page">Splunk home page</choice>
    <change>
      <condition value="manager_page">
        <link target="_blank">
          /app/search/buttercup_games
        </link>
      </condition>
      <condition value="splk_page">
        <link target="_blank">
          309
        </link>
      </condition>
    </change>
  </input>
</fieldset>
```

309
Set or update token values that other dashboard or form elements can consume. You can set tokens as part of creating dynamic behavior or display, such as passing a user selected value from a form input to a search or managing panel hide/show behavior.

Element hierarchy

Form input

<change>
  <condition> (optional)
  <set>

Drilldown

<drilldown>
  <condition> (optional)
  <set>

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>Token name</td>
<td><strong>Required.</strong> Use the token whose value you are setting or updating.</td>
<td>&lt;set token=&quot;[token_name]&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This example includes the `</td>
<td>s$` token filter to indicate that the token value</td>
</tr>
<tr>
<td>prefix</td>
<td>text</td>
<td>should be handled as a string.</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>String to add to the beginning of the token value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This example includes a quotation mark prefix and suffix around the token value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;set token=&quot;[token_name]&quot;&gt; prefix=&quot;sourcetype=&quot;&quot; suffix=&quot;&quot;&quot;&gt;$click.value$&lt;/set&gt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>suffix</th>
<th>text</th>
<th>String to append to the value of the token.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This example includes a quotation mark prefix and suffix around the token value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;set token=&quot;[token_name]&quot;&gt; prefix=&quot;sourcetype=&quot;&quot; suffix=&quot;&quot;&quot;&gt;$click.value$&lt;/set&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Example

A click on the table sets a token which is consumed by the search of the chart visualization.

```xml
<dashboard>
  <label>In-page Drilldown</label>
  <row>
    <panel>
      <table>
        <title>Set sourcetype token on click</title>
        <search>
          <query>
            index=_internal | stats count by sourcetype
          </query>
          <earliest>-1h</earliest>
          <latest>now</latest>
        </search>
        <drilldown>
          <condition field="sourcetype">$click.value2$</condition>
          <set token="sourcetype">$click.value2$</set>
        </drilldown>
      </table>
    </panel>
  </row>
</dashboard>
```
Use `<unset>` to clear a previously set token value. When you unset a token, its value is empty until it is set again. Unsetting token values can help you with implementing dynamic display or other behavior in dashboards and forms.

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>Token name</td>
<td><strong>Required.</strong> Use the token whose value you are unseting.</td>
</tr>
</tbody>
</table>

### Example

Use `<set>` and `<unset>` with `<depends>` and `<rejects>` in dashboard panels to change the visualization that appears on different user clicks.

```xml
<dashboard>
  <label>Example for <set> and <unset></label>
  <row>
    <panel>
      <table>
        <title>Set sourcetype token</title>
        <search>
          <query>
            index=_internal | stats count by sourcetype
          </query>
          <earliest>-1h</earliest>
          <latest>now</latest>
        </search>
        <drilldown>
          <!-- For the sourcetype field clicked: -->
        </drilldown>
      </table>
    </panel>
  </row>
</dashboard>
```
<!-- Set token to display a chart -->
<!-- Unset token to display a table -->
<condition field="sourcetype">
  <set token="sourcetype">$row.sourcetype$</set>
  <set token="showChart">foo</set>
  <unset token="showTable"></unset>
</condition>
<!-- For any other field clicked: -->
<!-- Set token to display a table -->
<!-- Unset token to display a chart -->
<condition field="*">
  <set token="sourcetype">$row.sourcetype$</set>
  <set token="showTable">foo</set>
  <unset token="showChart"></unset>
</condition>
</drilldown>
</table>
</panel>

<!-- Hide the html panel when either token is present -->
<!-- Click in the original table to set either token -->
<panel>
<html rejects="$showTable$, $showChart$">
<h2>Details</h2>
<div style="padding: 50px; margin: 0 auto; width: 350px;">
<div class="alert alert-warning">
<i class="icon-alert"/>
Click on a row in the table on the left to show details.
</div>
</div>
</html>
<!-- if showChart token is set, display results here -->
<chart depends="$showChart$">
<title>Details for $submitted:sourcetype|s$</title>
<search>
  <query>
  index=_internal sourcetype=$sourcetype|s$
  | timechart count by sourcetype
  </query>
  <earliest>-1h</earliest>
  <latest>now</latest>
</search>
</chart>
<!-- if showCTable token is set, display results here -->
<table depends="$showTable$">
<title>Details for $submitted:sourcetype|s$</title>
<search>
  <query>
  index=_internal sourcetype=$sourcetype|s$
  | timechart bins=10 count by sourcetype
  </query>
  <earliest>-1h</earliest>
</search>
</table>
</panel>

313
Predefined drilldown tokens

Predefined tokens capture information when a user clicks different visualization elements. Depending on the visualization type, you can use different tokens to capture a click location or related data.

**chart**

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$click.name$</td>
<td>X-axis field or category name for the clicked location. Not available if the user clicks the chart legend.</td>
</tr>
<tr>
<td>$click.value$</td>
<td>X-axis field or category value for the clicked location. Not available if the user clicks the chart legend.</td>
</tr>
<tr>
<td>$click.name2$</td>
<td>Y-axis field or series name for the clicked location. Not available if the user clicks the chart legend.</td>
</tr>
<tr>
<td>$click.value2$</td>
<td>Y-axis field or series value for the clicked location. Not available if the user clicks the chart legend.</td>
</tr>
<tr>
<td>$row.&lt;fieldname&gt;$</td>
<td>Access any y-axis field value corresponding to the clicked location x-axis. Not available if the user clicks the chart legend.</td>
</tr>
<tr>
<td>$row.&lt;x-axis-name&gt;$</td>
<td>Access any x-axis field value corresponding to the clicked location. Not available if the user clicks the chart legend.</td>
</tr>
<tr>
<td>$earliest$</td>
<td>Earliest time for the clicked chart segment. If not applicable, uses the earliest time for the search.</td>
</tr>
<tr>
<td>$latest$</td>
<td>Latest time for the clicked chart segment. If not applicable, uses the latest time for the search.</td>
</tr>
</tbody>
</table>
### Event

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$click.name$</code></td>
<td>Field name for the clicked element in the event list. If a field name is not available for the clicked location, the <code>$click.name$</code> value defaults as follows.</td>
</tr>
<tr>
<td></td>
<td>• Clicking an element in the event: <code>_raw</code></td>
</tr>
<tr>
<td></td>
<td>• Clicking the event timestamp: <code>_time</code></td>
</tr>
<tr>
<td></td>
<td>• Clicking a tag: uses the tag name. For example, if <code>host</code> is tagged, uses <code>host</code>.</td>
</tr>
<tr>
<td><code>$click.value$</code></td>
<td>Field value for the clicked element in the event list.</td>
</tr>
<tr>
<td><code>$click.name2$</code></td>
<td>Identical to <code>$click.name$</code>.</td>
</tr>
<tr>
<td><code>$click.value2$</code></td>
<td>Identical to <code>$click.value$</code>.</td>
</tr>
<tr>
<td><code>$row.&lt;fieldname&gt;$</code></td>
<td>Access any field value in the clicked event. For example, to access the <code>host</code> field value, use <code>$row.host$</code>.</td>
</tr>
<tr>
<td><code>$earliest$</code></td>
<td>Earliest time for the clicked event. Equivalent to the <code>_time</code> field value. Defaults to the earliest search time.</td>
</tr>
<tr>
<td><code>$latest$</code></td>
<td>Latest time for the clicked event. Equivalent to one second after the <code>_time</code> field value. Defaults to the latest search time.</td>
</tr>
</tbody>
</table>

### Map

Predefined tokens are available for cluster and Choropleth maps. Some tokens are only available in cluster maps.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$click.name$</code></td>
<td>Field name for the clicked location. If multiple fields are associated with the location, uses the first field.</td>
</tr>
<tr>
<td><code>$click.value$</code></td>
<td>Field value for the clicked location. If multiple fields are associated with the location, uses the first field.</td>
</tr>
<tr>
<td><code>$click.name2$</code></td>
<td>Same as <code>$click.name$</code></td>
</tr>
<tr>
<td><code>$click.value2$</code></td>
<td>Same as <code>$click.value$</code></td>
</tr>
<tr>
<td><code>$row.&lt;fieldname&gt;$</code></td>
<td>Access field values related to the clicked location. Check the Statistics tab for available</td>
</tr>
</tbody>
</table>

315
<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$earliest$</td>
<td>Earliest time for the search generating the map.</td>
</tr>
<tr>
<td>$latest$</td>
<td>Latest time for the search generating the map.</td>
</tr>
<tr>
<td>$click.lat.name$</td>
<td>For cluster maps: latitude field name for the clicked location.</td>
</tr>
<tr>
<td>$click.lat.value$</td>
<td>For cluster maps: latitude field value for the clicked location.</td>
</tr>
<tr>
<td>$click.lon.name$</td>
<td>For cluster maps: longitude field name for the clicked location.</td>
</tr>
<tr>
<td>$click.lon.value$</td>
<td>For cluster maps: longitude field value for the clicked location.</td>
</tr>
<tr>
<td>$click.bounds.&lt;orientation&gt;$</td>
<td>For cluster maps: south, west, north, or east outer boundary for the clicked location. For example, use $click.bounds.east$ to get the eastern outer boundary.</td>
</tr>
</tbody>
</table>

### single value

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$click.name$</td>
<td>Name of the field that the single value represents. In a <code>&lt;condition&gt;</code> element configuring drilldown for a single value, <code>field</code> corresponds to $click.name$.</td>
</tr>
<tr>
<td>$click.value$</td>
<td>Field value that the single value represents</td>
</tr>
<tr>
<td>$click.name2$</td>
<td>Same as $click.name$.</td>
</tr>
<tr>
<td>$click.value2$</td>
<td>Same as $click.value$.</td>
</tr>
<tr>
<td>$row.&lt;fieldname&gt;$</td>
<td>Access any field value from the Statistics table row for the single value.</td>
</tr>
<tr>
<td>$earliest$</td>
<td>Earliest time for the search that generates the single value</td>
</tr>
<tr>
<td>$latest$</td>
<td>Latest time for the search that generates the single value</td>
</tr>
</tbody>
</table>
table

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$click.name$</td>
<td>Leftmost field (column) name in the table. Typically, _time is leftmost in a table that includes this field and where columns are ordered by default.</td>
</tr>
<tr>
<td>$click.value$</td>
<td>Leftmost field (column) value in the clicked table row.</td>
</tr>
<tr>
<td>$click.name2$</td>
<td>Clicked field (column) name. In a &lt;condition&gt; element configuring drilldown in a table, field corresponds to $click.name2$.</td>
</tr>
<tr>
<td>$click.value2$</td>
<td>Clicked field (column) value. Use this token to capture the specific table cell value that users click.</td>
</tr>
<tr>
<td>$row.&lt;fieldname&gt;$</td>
<td>Access any field (column) value from the clicked table row.</td>
</tr>
<tr>
<td>$earliest$</td>
<td>Earliest time for the clicked table row. If not applicable, uses the earliest time of the search.</td>
</tr>
<tr>
<td>$latest$</td>
<td>Latest time for the clicked table row. If not applicable, uses the latest time of the search.</td>
</tr>
</tbody>
</table>

trellis

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$trellis.name$</td>
<td>Split field name</td>
</tr>
<tr>
<td>$trellis.value$</td>
<td>Split field value</td>
</tr>
</tbody>
</table>

Deprecations and removals

Check the Deprecated features list in the Release Notes for additional information on deprecated or removed elements.

Chart configuration reference

Chart overview

The <chart> element is a panel visualization that is highly configurable.

<chart>

A panel that displays search data in a chart. Saved reports contain chart formatting parameters. Saved searches do not. For more information, see "Save
When you load a saved report in the chart panel, your saved report format is also loaded. However, chart formatting can be overridden inline using the chart options.

Charts use named options to specify chart-specific properties. This reference contains sections on all configurable properties of charts.

### Parent elements

```
<row>
    <panel>

<chart>
    <title> (0..1)
    <search> (0..1)
        <earliest> (0..1)
        <latest> (0..1)
    <drilldown> (0..n)
    <selection> (0..n, for charts of type area, line, and column only)
    <option name="[property]"> (0..n)
```

### General chart properties

These are properties that apply to all charts unless otherwise specified.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.backgroundColor</td>
<td>Hex color value.</td>
<td></td>
<td>Chart background color.</td>
</tr>
<tr>
<td>charting.chart</td>
<td>(area</td>
<td>bar</td>
<td>bubble</td>
</tr>
<tr>
<td>charting.data.count</td>
<td>Number</td>
<td>10000</td>
<td>The number of results to retrieve. Set to 0 to get all results.</td>
</tr>
<tr>
<td>charting.data.fieldListMode</td>
<td>(show_hide</td>
<td>hide_show)</td>
<td>hide_show</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>charting.data.fieldShowList</td>
<td>array of fields</td>
<td>—</td>
<td>The list of fields to explicitly show in the results. This property does not apply to Scatter and Bubble charts.</td>
</tr>
<tr>
<td>charting.data.fieldHideList</td>
<td>array of fields</td>
<td>—</td>
<td>The list of fields to explicitly hide from the results, in JSON array format. This property does not apply to Scatter and Bubble charts.</td>
</tr>
<tr>
<td>charting.drilldown</td>
<td>(all</td>
<td>none)</td>
<td>all</td>
</tr>
<tr>
<td>charting.fieldColors</td>
<td>Map of hex colors. See description.</td>
<td>—</td>
<td>The map of hexadecimal color values to use for each field. A map is a comma-delimited list of key/value pairs, enclosed in curly brackets.</td>
</tr>
</tbody>
</table>
braces.

Keys are separated from their values by a colon.

Example:

```javascript
{"foo\: bar": 0xffff00, foo: 0xff0000, "foobar": 0x000000}
```

Escape the following special characters in a key or string value with double quotes:

```
[]{}(),:
```

Escape existing double quotes or backslashes or colons with a preceding backslash.

See Specify custom colors for fields in charts for an example.

<table>
<thead>
<tr>
<th>charting.fieldDashStyles</th>
<th>Map of dash styles.</th>
<th>?</th>
</tr>
</thead>
</table>

A map of dash styles to use for each field in JSON object format.

For example:

```
{"Field1": "shortDash"}.
```

Possible dash styles are: (dash | dashDot | dot | longDash | longDashDot | longDashDotDot | shortDash | shortDot |
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.fontColor</td>
<td>Hex color value.</td>
<td>shortDashDot</td>
</tr>
<tr>
<td>charting.foregroundColor</td>
<td>Hex color value.</td>
<td>Chart foreground color.</td>
</tr>
<tr>
<td>charting.legend.labels</td>
<td>CSV of labels — A list of labels with which to pre-populate the legend.</td>
<td>This property does not apply to Scatter and Bubble charts.</td>
</tr>
<tr>
<td>charting.legend.labelStyle.overflowMode</td>
<td>(ellipsisEnd</td>
<td>ellipsisMiddle</td>
</tr>
<tr>
<td>charting.legend.masterLegend</td>
<td>n/a</td>
<td>If attribute is present, disables legend color synchronization with other panels in the dashboard.</td>
</tr>
<tr>
<td>charting.seriesColors</td>
<td>List of hex colors</td>
<td>See below*</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
<td>-----------</td>
</tr>
</tbody>
</table>

Note: The only valid value is an empty tag. If a value is specified, the attribute is ignored.

Use an array of hexadecimal values to define the colors of chart series.

Note: To apply static colors to specific fields use the charting.fieldColors property.

**height**

Number — Height, in pixels, of the chart.

Default value is 250, must be between 100 and 10000.

*Default value for charting.seriesColors:

```
[0x1e93c6, 0xf2b827, 0xd6563c, 0x6a5c9e, 0x31a35f, 0xed8440, 0x3863a0, 0xa2cc3e, 0xcc5068, 0x73427f, 0x11a88b, 0xeaa9600, 0xe0776d, 0xffb380, 0x9a3977, 0x91af27, 0x4453aa, 0x314d5b, 0x8444539, 0x00b290, 0xe2c188, 0xa344a41, 0x4444f6d, 0xe29847, 0x8c8910, 0x0b416d, 0x774772, 0x3d9988, 0xbdbd5e, 0x5f7396, 0x8444539]
```

**General Chart Properties: selected examples**

```xml
<dashboard>
    <label>Selected chart examples</label>
    <row>
        <panel>
            <chart>
                <title>A line chart</title>
                <search>
                    <query>
                        index=_internal source="*metrics.log"
                        group=per_sourcetype_thruput
                        | timechart sum(kb) by series
                    </query>
                </search>
            </chart>
        </panel>
    </row>
</dashboard>
```
Use the <tt>eval</tt> function in the search to transpose
the value of the `<tt>log_level</tt>` field into individual fields for `<tt>charting.fieldcolors</tt>`.

```html
<chart>
<title>Field colors example</title>
<search>
<query>
  index = _internal log_level=* | stats
  count(eval(log_level="ERROR")) as ERROR
  count(eval(log_level="WARN")) as WARN
  count(eval(log_level="INFO")) as INFO
  by sourcetype
</query>
<earliest>-7d@h</earliest>
<latest>now</latest>
</search>
<option name="charting.axisY.scale">log</option>
<option name="charting.chart">column</option>
<option name="charting.fieldColors">
  "ERROR": 0xFF0000, "WARN": 0xFF9900, "INFO":0x0066FF, "NULL":0xC4C4C0
</option>
<option name="charting.legend.placement">right</option>
</chart>
</panel>
</row>
</dashboard>
```

**Area, Bubble, Bar, Column, Line, and Scatter charts**

Properties specific to line, area, column, scatter, bubble, and bar charts, all of which contain an x-axis and y-axis.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>charting.annotation.categoryColors</code></td>
<td>Map of hex or rgb colors.</td>
<td>—</td>
</tr>
</tbody>
</table>

See description.

The map of hexadecimal or RGB color values to use for each event annotation category. This property is only used for charts with event annotation overlays, where the field `annotation_category` is defined.

A map is a comma-delimited list of key/value pairs, enclosed in curly braces. Keys are separated from their values by a colon.

Example:

```
{"foo: bar": 0xffff00, foo: 0xff0000, "foobar": 0x000000}
```

Escape the following special characters in a key or string value with double quotes:

- `[`, `]`, `{`, `}`, `(`, `)`, `\`

Escape existing double quotes or backslashes or colons with a preceding backslash.

To view an example of this property in use, see Event Annotations in Simple XML.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>charting.axisLabelsX.axisVisibility</code></td>
<td>(show</td>
<td>hide)</td>
</tr>
<tr>
<td><code>charting.axisLabelsY.axisVisibility</code></td>
<td>(show</td>
<td>hide)</td>
</tr>
<tr>
<td><code>charting.axisLabelsY2.axisVisibility</code></td>
<td>(show</td>
<td>hide)</td>
</tr>
<tr>
<td><code>charting.axisLabelsX.extendsAxisRange</code></td>
<td>Boolean true</td>
<td>Indicates whether the range of the axis should be extended to snap to whole major tick marks.</td>
</tr>
<tr>
<td><code>charting.axisLabelsY.extendsAxisRange</code></td>
<td>Boolean false</td>
<td></td>
</tr>
<tr>
<td><code>charting.axisLabelsX.integerUnits</code></td>
<td>Boolean false</td>
<td>Indicates whether the major unit is rounded to the nearest integer. <code>charting.axisLabelsY.integerUnits</code> and <code>charting.axisLabelsY2.integerUnits</code> are only supported for bar and line charts.</td>
</tr>
<tr>
<td><code>charting.axisLabelsX.majorLabelStyle.overflowMode</code></td>
<td>(ellipsisMiddle</td>
<td>ellipsisNone)</td>
</tr>
<tr>
<td><code>charting.axisLabelsX.majorLabelStyle.rotation</code></td>
<td>(-90</td>
<td>-45</td>
</tr>
<tr>
<td><code>charting.axisLabelsX.majorLabelVisibility</code></td>
<td>(auto</td>
<td>show</td>
</tr>
<tr>
<td><code>charting.axisLabelsY2.majorLabelVisibility</code></td>
<td>(auto</td>
<td>show</td>
</tr>
</tbody>
</table>
auto: Shows or hides individual major labels to maintain readability in the available space without overlapping.
show: Show all major labels, even if overlapping occurs.
hide: Hide all major labels.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.axisLabelsX.majorTickSize</td>
<td>number</td>
<td>6</td>
</tr>
<tr>
<td>charting.axisLabelsY.majorTickSize</td>
<td>number</td>
<td>6</td>
</tr>
<tr>
<td>charting.axisLabelsX.minorTickSize</td>
<td>number</td>
<td>6</td>
</tr>
<tr>
<td>charting.axisLabelsY.minorTickSize</td>
<td>Number</td>
<td>6</td>
</tr>
<tr>
<td>charting.axisLabelsY2.minorTickSize</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsX.majorTickVisibility</td>
<td>(auto</td>
<td>show</td>
</tr>
<tr>
<td>charting.axisLabelsY.majorTickVisibility</td>
<td>(auto</td>
<td>show</td>
</tr>
<tr>
<td>charting.axisLabelsY2.majorTickVisibility</td>
<td>(auto</td>
<td>show</td>
</tr>
</tbody>
</table>

The size, in pixels of the major tick marks.
charting.axisLabelsY2.majorTickSize is deprecated.

The size, in pixels of the minor tick marks.
charting.axisLabelsY2.minorTickSize is not supported for bubble and scatter charts.

Indicates whether major tick marks are visible.
charting.axisLabelsY2.majorTickVisibility depends on axis type.
Applies only to Area, Bar, Column, and Line charts. Indicates whether major tick marks are visible.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.axisLabelsX.majorUnit</td>
<td>(Positive integer</td>
<td>auto)</td>
</tr>
<tr>
<td>charting.axisLabelsX.majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsX.majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY-majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY-majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY-majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY2-majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY2-majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY2-majorUnit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsX.minorTickVisibility</td>
<td>(auto</td>
<td>show</td>
</tr>
<tr>
<td>charting.axisLabelsX.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsX.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY2.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY2.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisLabelsY2.minorTickVisibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.abbreviation</td>
<td>(none</td>
<td>auto)</td>
</tr>
<tr>
<td>charting.axisX.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY2.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY2.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY2.abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.includeZero</td>
<td>Boolean</td>
<td>false</td>
</tr>
<tr>
<td>charting.axisX.includeZero</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.includeZero</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.includeZero</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.maximumNumber</td>
<td>Number</td>
<td>auto</td>
</tr>
<tr>
<td>charting.axisX.maximumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.maximumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.maximumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.maximumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.minimumNumber</td>
<td>Number</td>
<td>auto</td>
</tr>
<tr>
<td>charting.axisX.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisX.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charting.axisY.minimumNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Default</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>charting.axisX.scale</td>
<td>Use a linear or logarithmic scale. Only scatter charts support a logarithmic scale.</td>
<td>linear</td>
</tr>
<tr>
<td>charting.axisY.scale</td>
<td></td>
<td>linear</td>
</tr>
<tr>
<td>charting.axisTitleX.text</td>
<td>Specifies the title of the axis.</td>
<td>—</td>
</tr>
<tr>
<td>charting.axisTitleY.text</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>charting.axisTitleY2.text</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>charting.axisTitleX.visibility</td>
<td>Indicates whether to show the title of the x-axis.</td>
<td>visible</td>
</tr>
<tr>
<td>charting.axisTitleY.visibility</td>
<td></td>
<td>visible</td>
</tr>
<tr>
<td>charting.axisTitleY2.visibility</td>
<td></td>
<td>collapsed</td>
</tr>
<tr>
<td>charting.chart.resultTruncationLimit</td>
<td>Override default limits for the number of data points rendered in a chart.</td>
<td>50000</td>
</tr>
<tr>
<td>charting.gridLinesX.showMajorLines</td>
<td>Indicates whether major grid lines on the X-axis are visible.</td>
<td>false</td>
</tr>
<tr>
<td>charting.gridLinesY.showMajorLines</td>
<td></td>
<td>true</td>
</tr>
<tr>
<td>charting.gridLinesY2.showMajorLines</td>
<td>Applies only to Area, Bar, Column, and Line charts. Indicates whether major grid lines on the second Y-axis are visible.</td>
<td>false</td>
</tr>
<tr>
<td>charting.gridLinesX.showMinorLines</td>
<td>Applies only to Area, Bar, Column, and Line charts. Indicates whether minor grid lines are visible.</td>
<td>False</td>
</tr>
<tr>
<td>charting.gridLinesY.showMinorLines</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>charting.gridLinesY2.showMinorLines</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>charting.layout.splitSeries</td>
<td>Applies only to Area, Bar, Column, and Line charts. Splits a multi-series chart into separate charts that are stacked from top to bottom, one for each series.</td>
<td>False</td>
</tr>
<tr>
<td>charting.layout.splitSeries.allowIndependentYRanges</td>
<td>When set to True, allows each series to have its own Y-range.</td>
<td>False</td>
</tr>
<tr>
<td>charting.legend.placement</td>
<td>Where to place the legend.</td>
<td>right</td>
</tr>
</tbody>
</table>
charting.legend.mode

(standard | seriesCompare) standard

Provides visual and behavior settings.

"Standard" is the default, "seriesCompare" is useful when comparing series (for example, this setting changes the styles for tooltips).

You can only use this property for line charts and area charts.

**Area chart properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.areaFillOpacity</td>
<td>0 - 1.0</td>
<td>.75</td>
<td>Configures the opacity of an area chart. 1.0 means the area chart is solid. 0 indicates the area chart is transparent.</td>
</tr>
<tr>
<td>charting.axisY2.enabled</td>
<td>boolean</td>
<td>false</td>
<td>Enables a second y-axis for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.fields</td>
<td>comma delimited list</td>
<td></td>
<td>Fields to be mapped to a second y-axis for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.includeZero</td>
<td>boolean</td>
<td>false</td>
<td>Indicates whether to include zero in the second y-axis range for chart</td>
</tr>
<tr>
<td>Key</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>charting.axisY2.maximumNumber</td>
<td>Number</td>
<td>auto</td>
<td>Sets the maximum number for the y-axis range for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.minimumNumber</td>
<td>Number</td>
<td>auto</td>
<td>Sets the minimum number for the y-axis range for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.scale</td>
<td>(inherit</td>
<td>linear</td>
<td>log)</td>
</tr>
<tr>
<td></td>
<td>inherit</td>
<td></td>
<td>Inherit from the first y-axis, or use a linear or logarithmic scale.</td>
</tr>
<tr>
<td>charting.chart.overlayFields</td>
<td>comma-delimited list</td>
<td></td>
<td>List of fields to use for a chart overlay.</td>
</tr>
<tr>
<td>charting.chart.showDataLabels</td>
<td>(all</td>
<td>minmax</td>
<td>none)</td>
</tr>
<tr>
<td></td>
<td>none</td>
<td></td>
<td><strong>all</strong>: Display all labels</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>charting.chart&gt;nullValueMode</td>
<td>(gaps</td>
<td>zero</td>
<td>connect)</td>
</tr>
<tr>
<td>charting.chart.showLines</td>
<td>Boolean</td>
<td>true</td>
<td>Indicates whether to show lines in area charts.</td>
</tr>
<tr>
<td>charting.chart.stackMode</td>
<td>(default</td>
<td>stacked</td>
<td>stacked100)</td>
</tr>
</tbody>
</table>

Bar chart properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.barSpacing</td>
<td>Number</td>
<td>1</td>
<td>Specifies, in pixels, the spacing between bars in a bar chart.</td>
</tr>
<tr>
<td>charting.chart.seriesSpacing</td>
<td>Number</td>
<td>—</td>
<td>Specifies, in pixels, the spacing between clustered series in a bar chart.</td>
</tr>
<tr>
<td>charting.chart.showDataLabels</td>
<td>(all</td>
<td>minmax</td>
<td>none)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>all</strong>: Display all labels</td>
</tr>
</tbody>
</table>
### Bubble chart properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.bubbleMaximumSize</td>
<td>Number</td>
<td>50</td>
<td>Specifies, in pixels, the maximum size of each bubble.</td>
</tr>
<tr>
<td>charting.chart.bubbleMinimumSize</td>
<td>Number</td>
<td>10</td>
<td>Specifies, in pixels, the minimum size of each bubble.</td>
</tr>
<tr>
<td>charting.chart.bubbleSizeBy</td>
<td>(area</td>
<td>area</td>
<td>Determines whether the area or the diameter determine the bubble size.</td>
</tr>
<tr>
<td></td>
<td>diameter)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Column chart properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.axisY2.enabled</td>
<td>boolean</td>
<td>false</td>
<td>Enables a second y-axis for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.fields</td>
<td>comma delimited</td>
<td>—</td>
<td>Fields to be mapped to a second y-axis for chart</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>charting.axisY2.includeZero</td>
<td>boolean</td>
<td>false</td>
<td>Indicates whether to include zero in the second y-axis range for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.maximumNumber</td>
<td>Number</td>
<td>auto</td>
<td>Sets the maximum number for the y-axis range for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.minimumNumber</td>
<td>Number</td>
<td>auto</td>
<td>Sets the minimum number for the y-axis range for chart overlays.</td>
</tr>
<tr>
<td>charting.axisY2.scale</td>
<td>(inherit</td>
<td>inherit</td>
<td>Scale to use for a second y-axis for chart overlays.</td>
</tr>
<tr>
<td>charting.chart.columnSpacing</td>
<td>Number</td>
<td>1</td>
<td>Specifies, in pixels, the spacing between columns.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>charting.axisY2.enabled</td>
<td>boolean</td>
<td>false</td>
<td>Enables a second</td>
</tr>
<tr>
<td>charting.chart.overlayFields</td>
<td>comma-delimited</td>
<td>—</td>
<td>List of fields to use for a chart overlay.</td>
</tr>
<tr>
<td>charting.chart.seriesSpacing</td>
<td>Number</td>
<td>—</td>
<td>Specifies, in pixels, the spacing between clustered series in a column chart.</td>
</tr>
<tr>
<td>charting.chart.showDataLabels</td>
<td>(all</td>
<td>minmax</td>
<td>none)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>all: Display all labels</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>minmax: Display labels only for the lowest and highest values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>none: Do not display labels.</td>
</tr>
<tr>
<td>charting.chart.stackMode</td>
<td>(default</td>
<td>stacked</td>
<td>stacked100)</td>
</tr>
</tbody>
</table>

**Line chart properties**
<table>
<thead>
<tr>
<th>charting.axisY2.fields</th>
<th>comma delimited list</th>
<th>Fields to be mapped to a second y-axis for chart overlays.</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.axisY2.includeZero</td>
<td>boolean</td>
<td>false</td>
</tr>
<tr>
<td>charting.axisY2.maxiumNumber</td>
<td>Number</td>
<td>auto</td>
</tr>
<tr>
<td>charting.axisY2.minimumNumber</td>
<td>Number</td>
<td>auto</td>
</tr>
<tr>
<td>charting.axisY2.scale</td>
<td>(inherit</td>
<td>linear</td>
</tr>
</tbody>
</table>
|                                  |                      |                                                         | Inherit from the first y-axis, or use a linear scale.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.nullValueMode</td>
<td>(gaps</td>
<td>zero</td>
</tr>
<tr>
<td>charting.chart.overlayFields</td>
<td>comma-delimited list</td>
<td>List of fields to use for a chart overlay.</td>
</tr>
<tr>
<td>charting.chart.showDataLabels</td>
<td>(all</td>
<td>minmax</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>all</strong>: Display all labels</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>minmax</strong>: Display labels only for the lowest and highest values.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>none</strong>: Do not display labels.</td>
</tr>
<tr>
<td>charting.chart.showMarkers</td>
<td>Boolean false</td>
<td>Indicates whether to draw markers in line charts.</td>
</tr>
<tr>
<td>charting.chart.stackMode</td>
<td>(default</td>
<td>stacked</td>
</tr>
<tr>
<td>charting.lineDashStyle</td>
<td>solid</td>
<td></td>
</tr>
</tbody>
</table>
Specifies the dash style for all line series in a chart.

**charting.lineWidth**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.lineWidth</td>
<td>float</td>
<td>2</td>
<td>Line width, in pixels, for all line series in the chart. You can provide decimal values if needed.</td>
</tr>
</tbody>
</table>

**Scatter chart properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.markerSize</td>
<td>Number</td>
<td>4</td>
<td>Indicates, in pixels, the size of markers.</td>
</tr>
</tbody>
</table>

**Gauge charts**

Properties specific to gauge charts:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.gaugeColors</td>
<td>[Hex,...]</td>
<td>[0x84E900, 0xFFE800, 0xBF3030]</td>
<td>An array of hexadecimal color values from which the range band colors are generated. Colors display in the order indicated in the array.</td>
</tr>
</tbody>
</table>
For example, you can reverse the default green-yellow-red sequence by changing the gaugeColors value to:

```
[0xBF3030, 0xFFE800, 0x84E900]
```

You can specify any number of colors. If the gauge has more or less range intervals than the number of rangeColors, colors are interpolated as necessary. This interpolation occurs regardless of whether you specify the range interval in the search language or the rangeValues parameter.

<table>
<thead>
<tr>
<th>charting.chart.majorUnit</th>
<th>Number</th>
<th>auto</th>
<th>Specifies, in pixels, the spacing of major tick marks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.rangeValues</td>
<td>array of number</td>
<td>—</td>
<td>A numeric array that represents the overall numerical range represented by the gauge, and the relative size of the color-coded subranges within that overall range.</td>
</tr>
</tbody>
</table>

For example, a range of:

```
[0, 30, 70, 100]
```

indicates that the gauge starts at zero, ends at 100, and has three subranges that are each identified by another filler color. If the search returns a value of 71, the filler rises to that value on the gauge and takes on the color assigned to the top range, which is 71-100.

**Note:** When you specify range values in simple XML, they
override range values that are specified through the search upon which the dashboard panel is based.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.showLabels</td>
<td>Boolean</td>
<td>True</td>
<td>Indicates whether to display labels.</td>
</tr>
<tr>
<td>charting.chart.showMajorTicks</td>
<td>Boolean</td>
<td>True</td>
<td>Indicates whether to display major tick marks.</td>
</tr>
<tr>
<td>charting.chart.showMinorTicks</td>
<td>Boolean</td>
<td>See description</td>
<td>Indicates whether to display minor tick marks. Defaults to False for radial gauge and True for filler and marker gauges</td>
</tr>
<tr>
<td>charting.chart.showValue</td>
<td>Boolean</td>
<td>True</td>
<td>Indicates whether the gauge displays its value.</td>
</tr>
<tr>
<td>charting.chart.style</td>
<td>(minimal</td>
<td>shiny</td>
<td>Specify the display style of the gauge.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>)</td>
<td>shiny: A graphically stylized version of the gauge with chrome, shading, and other features to mimic a real world gauge. minimal: A &quot;just the basics&quot; version of the gauge.</td>
</tr>
<tr>
<td>charting.chart.usePercentageRange</td>
<td>Boolean</td>
<td>False</td>
<td>Indicates whether to format the range values as percentages.</td>
</tr>
<tr>
<td>charting.chart.usePercentageValue</td>
<td>Boolean</td>
<td>False</td>
<td>Indicates whether to format the gauge values as percentages.</td>
</tr>
</tbody>
</table>

**Filler gauge specific properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.orientation</td>
<td>(x</td>
<td>y)</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x: horizontal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>y: vertical</td>
</tr>
</tbody>
</table>
**Marker gauge specific properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.orientation</td>
<td>(x</td>
<td>y)</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x: horizontal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>y: vertical</td>
</tr>
<tr>
<td>charting.chart.showRangeBand</td>
<td>boolean</td>
<td>true</td>
<td>Indicates whether to show the color ranges as a band on the left side of the marker gauge.</td>
</tr>
</tbody>
</table>

**Radial gauge specific properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.rangeArcAngle</td>
<td>Number</td>
<td>270</td>
<td>The length of the range arc, in degrees. Positive values are clockwise. Negative values are counterclockwise.</td>
</tr>
<tr>
<td>charting.chart.rangeStartAngle</td>
<td>Number</td>
<td>45</td>
<td>The angle, in degrees, to begin drawing the range arc. The range arc is clockwise and starts from the bottom of the gauge.</td>
</tr>
<tr>
<td>charting.chart.showRangeBand</td>
<td>boolean</td>
<td>true</td>
<td>Indicates whether to show the color ranges as a band at the top of the radial gauge.</td>
</tr>
</tbody>
</table>

**Pie charts**

Properties specific to pie charts:
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>charting.chart.sliceCollapsingLabel</td>
<td>Text</td>
<td>Other</td>
<td>The label for the consolidated slice.</td>
</tr>
<tr>
<td>charting.chart.sliceCollapsingThreshold</td>
<td>Number</td>
<td>0.01</td>
<td>The threshold at which smaller slices collapse into a consolidated slice.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid values are between 0 and 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 indicates no collapsing. 1 indicates all slices collapse into a single pie.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default value, 0.01, collapses slices smaller than 1% of the whole pie.</td>
</tr>
<tr>
<td>charting.chart.showLabels</td>
<td>Boolean</td>
<td>true</td>
<td>Indicates whether to display labels.</td>
</tr>
<tr>
<td>charting.chart.showPercent</td>
<td>Boolean</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>
Event Handler Reference

Use Simple XML event handler elements to create responsive dashboard behavior.

**What is an event handler?**

In web programming, event handlers let you listen for and define responses to state changes and user behavior, such as a mouse click or scroll. These changes or interactions are events. Event handlers in Simple XML work similarly.

**Event handling contexts**

Depending on the context, such as a form input or drilldown, the events that can happen and that are exposed in Simple XML elements vary. The use cases for handling events can also vary by context.

For example, in a form input, you can use the `<change>` element that listens for a user selection and sets a token value based on that selection. This token value update can trigger panel show or hide behavior in the form, customizing the content to the selection.

<table>
<thead>
<tr>
<th>Context</th>
<th>Events that happen in this context</th>
<th>Use case example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form inputs</td>
<td>User selections or input</td>
<td>Customize the form content in response to user input values.</td>
</tr>
<tr>
<td>Search states</td>
<td>A search has several state events, such as being in progress, cancelled, or done.</td>
<td>Use search event handlers to listen for search state changes and capture data or create dynamic behavior during different states.</td>
</tr>
<tr>
<td>Drilldown</td>
<td>User clicks on a visualization where drilldown is enabled</td>
<td>Use drilldown to share additional data insights with your dashboard users. For example, you can open a</td>
</tr>
<tr>
<td>Dashboard page load</td>
<td>Page loads in the browser window, displaying the dashboard</td>
<td>Set tokens on page load to manage initial display settings such as panel show or hide or search results for a visualization.</td>
</tr>
</tbody>
</table>

**Customizing responsive behavior**

In some elements, you can define conditional responses to state changes. For example, you might want to handle particular search result field values differently when a search completes.

**Tokens and event handlers**

Use predefined or custom tokens to capture dynamic values or to help you implement responsive behavior. Depending on the context, available predefined tokens vary.

If you are unfamiliar with tokens, see Token usage in dashboards.

**Form inputs**

Respond to user input in any of the following form input elements.

- `<checkbox>`
- `<dropdown>`
- `<link>`
- `<multiselect>`
- `<radio>`
- `<text>`
- `<time>`

**Event handler element**

Use the `<change>` element to define responses to user selections in the input. You can include a `<condition>` child element in `<change>` to handle user input values conditionally.

**Note:** The `<change>` element is not available for multiselect inputs.
For more details, see [[[ ]] in the Simple XML Reference.

### Form input event tokens

Respond to user selections in a form input. Use predefined tokens to access the selected `<choice>` element label or value.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>Selected <code>&lt;choice&gt;</code> element label.</td>
</tr>
<tr>
<td>value</td>
<td>Selected <code>&lt;choice&gt;</code> element value.</td>
</tr>
</tbody>
</table>

**Example** Use the `<change>` element to capture the selected label and value from an input.

```xml
<form>
  <label>Use tokens with input choices to capture input labels and values</label>
  <fieldset submitButton="false">
    <input type="radio" token="period_tok">
    <label>Select a time range</label>
    <choice value="-24h@h">Last 24 Hours</choice>
    <choice value="-7d@d">Last 7 Days</choice>
    <choice value="-30d@d">Last 30 Days</choice>
    <default>Last 24 Hours</default>
  </input>
  <change>
    <!-- use predefined input tokens to set -->
    <!-- tokens for the selected label and value -->
    <set token="date_label">$label$</set>
    <set token="earliest_tok">$value$</set>
  </change>
  </fieldset>
  <row>
    <panel>
      <title>Conditional Inputs</title>
      <chart>
        <!-- Display selected label in the title -->
      </chart>
    </panel>
  </row>
</form>
```
Search event handlers

Search event handlers allow you to enable event actions based on search results or search properties. Actions include linking to a page, setting or unsetting tokens, and executing an eval function.

Search event tokens

Search event handlers use predefined tokens to access the search results and search properties. The tokens available to each handler vary. In some cases, the event handler does not access a predefined token to enable an action.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>job.property</td>
<td>Access the value of the named job property or one of its secondary properties. For example, use</td>
</tr>
</tbody>
</table>
$job.request.earliest_time$ and $job.request.latest_time$
to access information about the search time range.

You can also view properties for a search from the Search Job Inspector. From the Search Page, after running a search select **Job > Inspect Job**.

See View search job properties in the *Search Manual* for a list of properties available.

| **result.field** | Access the value of the named field. The token accesses the value from the first row of returned results. |

### Search element syntax

```
<done | error | fail | cancelled | progress>
<condition match=(dashboard eval expression)> (0..n)
  (<eval> | <link> | <set> | <unset>) (1..n)
```

For detailed information on search event tokens, see Define search tokens.

**cancelled**

```
<cancelled>
```

Execute actions when a search is cancelled.

**Parent element**

```
<search>
```

```
<cancelled>
  <condition match="(dashboard eval expression)" (0..n)
    (<eval> | <link> | <set> | <unset>) (1..n)
```

**Tokens available**

No tokens for this element.

**Example**

```
<cancelled>
  <unset token="sourcetype_count" />
</cancelled>
```
error

<error>
Execute actions when there is a search error event, such as an invalid query.

Parent element
<search>

<error>
  <condition match="(dashboard eval expression)">(0..n)
    (<eval> | <link> | <set> | <unset>) (1..n)
</condition>
Tokens available
No tokens for this element.

Example
<search>
  <error>
    <set token="error_message">$message$</set>
  </error>
</search>

fail

<fail>
Execute an action when a search fails while running.

Parent element
<search>

<fail>
  <condition match="(dashboard eval expression)">(0..n)
    (<eval> | <link> | <set> | <unset>) (1..n)
</condition>
Tokens available
No tokens for this element. Only the failure message is available.

Example
<search>
  <fail>
    <set token="fail_message">$message$</set>
  </fail>
</search>
```
<search/>

**progress**

<table>
<thead>
<tr>
<th><strong>&lt;progress&gt;</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute an action on search progress events. Access job properties and the first results row.</td>
</tr>
</tbody>
</table>

**Parent element**

```
<search>
```

```
<progress>
    <condition match="(dashboard eval expression)">(0..n)
      (<eval> | <link> | <set> | <unset>) (1..n)
    </condition>
</progress>
```

**Tokens available**

```
job.property
result.field
```

**Example**

```
<progress>
    <condition match="'job.resultCount' == 0">
      <set token="show_html">true</set>
    </condition>
    <condition>
      <unset token="show_html"/>
    </condition>
</progress>
```

**done**

```
<done>
```

<table>
<thead>
<tr>
<th><strong>&lt;done&gt;</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute actions based on finished search events.</td>
</tr>
</tbody>
</table>

**Parent element**

```
<search>
```

```
<done>
    <condition match="(dashboard eval expression)">>(0..n)
      (<eval> | <link> | <set> | <unset>) (1..n)
    </condition>
</done>
```

**Tokens available**

```
job.property
```
result.field

Example

```xml
<done>
  <condition match="'job.resultCount' == 0">
    <set token="show_html">true</set>
  </condition>
  <condition>
    <unset token="show_html"/>
  </condition>
</done>
```

Visualization event handlers

```
<Visualization>
  Event handlers apply to the following visualization types:
  - chart
  - event
  - map
  - single
  - table

  <drilldown> (0..n)
    <condition [label=?foo? | value=?foo? | match=(dashboard eval expression)]>(0..n)
      (<eval> | <link> | <set> | <unset>) (1..n)
    </condition>
  </drilldown>

  <selection> (0..n, for charts of type area, line, and column only)
    (<eval> | <link> | <set> | <unset>) (1..n)
</Visualization>
```

<table>
<thead>
<tr>
<th>Child elements</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;drilldown&gt;</td>
<td>Event actions</td>
<td>—</td>
<td>Actions to take for drilldown behavior.</td>
</tr>
<tr>
<td>&lt;selection&gt;</td>
<td>&lt;set&gt;</td>
<td>—</td>
<td>Applies to charts of type area, column, or line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the &lt;set&gt; element to define tokens for the time window used in the pan and zoom feature of charts.</td>
</tr>
</tbody>
</table>

Example
Example line chart panel using an inline search. It limits results to a specified time window and provides labels for the X and Y axes:

```xml
<dashboard>
  <label>Top source types in the last week</label>
  <row>
    <panel>
      <title>Chart example</title>
      <chart>
        <title>Top sourcetypes in the last week</title>
        <search>
          <query>
            index=_internal source="*metrics.log" group=per_sourcetype_thruput
            timechart sum(kb) by series
          </query>
          <earliest>-1w</earliest>
          <latest>now</latest>
        </search>
        <option name="height">200px</option>
        <option name="charting.chart">line</option>
        <option name="charting.axisY.scale">log</option>
        <option name="charting.chart.nullValueMode">connect</option>
      </chart>
    </panel>
    ... 
  </row>
</dashboard>
```

**Drilldown event tokens**

For dynamic drilldown, there are predefined tokens available for each type of visualization. The value of a predefined token can vary, depending on the visualization.

- Chart event tokens
- Event event tokens
- Map event tokens
- Single event tokens
- Table event tokens
chart (event tokens)

The clicked field name is the name of the field or series for the y-Axis if present (similar to click.name2). If the name of the field or series is not available the field or category for the x-axis is used (click.name).

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click.name</td>
<td>Name of the field or category for the x-axis. Not available when the legend has been clicked.</td>
</tr>
<tr>
<td>click.value</td>
<td>Value of the field or category for the x-axis. Not available when the legend has been clicked.</td>
</tr>
<tr>
<td>click.name2</td>
<td>Name of the field or series for the y-axis.</td>
</tr>
<tr>
<td>click.value2</td>
<td>Value of the field or series for the y-axis. Not available when the legend has been clicked.</td>
</tr>
<tr>
<td>row.&lt;fieldname&gt;</td>
<td>Any field values along the y-axis at the same point as the click on the x-axis. Not available when the legend has been clicked.</td>
</tr>
<tr>
<td>row.&lt;x-axis-name&gt;</td>
<td>Value of the x-axis. Not available when the legend has been clicked.</td>
</tr>
<tr>
<td>earliest/latest</td>
<td>Time range of the clicked chart segment, or if not applicable, the time range of the search.</td>
</tr>
</tbody>
</table>

event (event tokens)

The value for click.name depends on the context of the click, as described below:

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click.name</td>
<td>The field name associated with the click.</td>
</tr>
</tbody>
</table>

For cases in the event viewer where the field name is ambiguous:

- Click a term in the raw event: Sets _raw as the field name.
- Click the event timestamp: Sets _time as the field name.
- Click a tag: Sets a field name according to the tag name, as follows:
<table>
<thead>
<tr>
<th>Data Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click.name</td>
<td>Name of the first, or only field, that displays the marker.</td>
</tr>
<tr>
<td>click.value</td>
<td>Value of the first, or only field, that displays the marker.</td>
</tr>
<tr>
<td>click.name2</td>
<td>Same as click.name.</td>
</tr>
<tr>
<td>click.value2</td>
<td>Same as click.value.</td>
</tr>
<tr>
<td>click.lat.name</td>
<td>Name of the latitude field that determines the location of the marker.</td>
</tr>
<tr>
<td>click.lat.value</td>
<td>Latitude value of the geo location of the marker.</td>
</tr>
<tr>
<td>click.lon.name</td>
<td>Name of the longitude field that determines the location of the marker.</td>
</tr>
<tr>
<td>click.lon.value</td>
<td>Longitude value of the geo location of the marker.</td>
</tr>
<tr>
<td>click.bounds.&lt;orientation&gt;</td>
<td>Outer boundaries of all clustered locations that the marker represents.</td>
</tr>
<tr>
<td>row.&lt;fieldname&gt;</td>
<td>Exposes each field value as row.&lt;fieldname&gt;.</td>
</tr>
</tbody>
</table>

map (event tokens)

The field for the <condition> tag in dynamic drilldown always corresponds to click.name.
Each field value of the clicked marker is exposed in this form.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>earliest/latest</td>
<td>Time range of the search driving the map visualization.</td>
</tr>
</tbody>
</table>

**single (event tokens)**

The field for the `<condition>` tag in dynamic drilldown always corresponds to `click.name`.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click.name</td>
<td>Name of the field that is displayed by the single value visualization.</td>
</tr>
<tr>
<td>click.value</td>
<td>Value that is displayed by the single value visualization.</td>
</tr>
<tr>
<td>click.name2</td>
<td>Same as click.name.</td>
</tr>
<tr>
<td>click.value2</td>
<td>Same as click.value.</td>
</tr>
<tr>
<td>row.&lt;fieldname&gt;</td>
<td>Exposes each field in the same result row from which the single value is taken.</td>
</tr>
<tr>
<td>earliest/latest</td>
<td>Time range of the search driving the single value visualization.</td>
</tr>
</tbody>
</table>

**table (event tokens)**

The field for the `<condition>` tag in dynamic drilldown always corresponds to `click.name2`.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>click.name</td>
<td>Name of the leftmost field that is displayed in the table. This is always <code>_time</code>, if present.</td>
</tr>
<tr>
<td>click.value</td>
<td>Value of the left-most column in the clicked row.</td>
</tr>
<tr>
<td>click.name2</td>
<td>Name of the clicked column.</td>
</tr>
<tr>
<td>click.value2</td>
<td>Value of the clicked column.</td>
</tr>
<tr>
<td>row.&lt;fieldname&gt;</td>
<td>All field values for the clicked table row, including those fields that are not displayed.</td>
</tr>
<tr>
<td>earliest/latest</td>
<td>Time range of the clicked table row, or if not applicable, the time range of the search.</td>
</tr>
</tbody>
</table>
**drilldown**

**<drilldown>**

Define custom destinations to link to when a user clicks on fields in a dashboard or form.

- Specify a path to the destination using the `<link>` tag.
- Set or unset tokens using the `<set>` or `<unset>` tags.
- Specify a condition to specify fields for setting or unsetting tokens.

**Note:** You can specify one or more actions (<eval>, <link>, <set>, <unset>) or conditions (<condition>) directly within `<drilldown>`, but you cannot specify both actions and conditions.

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>text</td>
<td>—</td>
<td>Corresponds to the <code>target</code> attribute of the <code>&lt;a&gt;</code> HTTP tag. Specify &quot;_blank&quot; to open the drilldown in a new window. Specify &quot;_self&quot; to open the drilldown in the same window. Specify an arbitrary string to open the drilldown in a new window. Subsequent references to this target open in this window.</td>
</tr>
</tbody>
</table>

### Parent elements

- `<chart>`
- `<event>`
- `<map>`
- `<single>`
- `<table>`

**Example 1: Pass a value to a form**

```xml
<table>
  <search>
    <query>index=_internal</query>
  </search>

  <!-- Pass the clicked row's 'count'-column value -->
  <!-- to populate a destination form's 'foo' token. -->
  <drilldown>
    <link>
      /app/search/simple_xml_form?form.foo=$row.count$
    </link>
  </drilldown>
</table>
```
Example 2: Pass parameters to a form

```xml
<table>
  <search>
    <query>index=_internal</query>
  </search>

  <!-- Pass the clicked cell's value, earliest time, -->
  <!-- and latest time to a destination form's       -->
  <!-- token ('foo') and search parameters           -->
  <drilldown>
    <link>
      <![CDATA[
        /app/search/simple_xml_form?form.foo=$click.value2$&earliest=$earliest$&latest=$latest$
      ]]>)
    </link>
  </drilldown>
</table>

Example 3: Pass a value from a chart to a website

```xml
<chart>
  <search>
    <query>index=_internal | chart count by sourcetype</query>
  </search>
  <option name="charting.chart">column</option>

  <!-- $click.value$ captures the value clicked by the user -->
  <!-- From the x-axis of a column chart and passes       -->
  <!-- it to the website as a query parameter          -->
  <drilldown>
    <link>
      http://splunk-base.splunk.com/integrated_search/?q=$click.value$
    </link>
  </drilldown>
</chart>
```

**selection**

Sets the time window for the pan and zoom feature of charts. You can also use tokens to set other values, such as the numerical values of the x-axis in a chart.

Only applies to charts of type area, column, or line.

See Chart controls for details on the pan and zoom feature of charts.

**Parent elements**

```xml
<selection>
  <chart>
    <option name="charting.chart">area</option>
  </chart>
</selection>
```
Use pre-defined tokens to capture the earliest and latest time of the time window and the earliest and latest values within that time window for a field.

For example:

```xml
<selection>
  <set token="selection.earliest">$start$</set>
  <set token="selection.latest">$end$</set>
  <set token="start.[fieldname]">$start.[fieldname]$</set>
  <set token="end.[fieldname]">$end.[fieldname]$</set>
</selection>
```

Can also be used to set a drilldown link.

```xml
<selection>
  <link>

Attributes

No attributes for this element.

Example

A selection on the left chart zooms into the right chart with details for the selected area.

```xml
<dashboard>
  <label>Pan and Zoom</label>
  <row>
    <panel>
      <chart>
        <title>Pan and Zoom (All source types)</title>
        <search>
          <query>
            index=_internal | timechart count by sourcetype
          </query>
          <earliest>-7d@h</earliest>
          <latest>now</latest>
        </search>
        <option name="charting.axisX.scale">linear</option>
        <option name="charting.axisY.scale">log</option>
        <option name="charting.chart">line</option>
      </chart>
    </panel>
  </row>
</dashboard>
```
<set token="selection.earliest">$start$</set> <set token="selection.latest">$end$</set>
<set token="start.splunk_web_access">$start.splunk_web_access$</set> <set token="end.splunk_web_access">$end.splunk_web_access$</set>

<title>Pan and Zoom (Web access source type)</title>
<search>
  <query>
    index=_internal sourcetype=splunk_web_access
    | timechart count by sourcetype
  </query>
  <earliest>$selection.earliest$</earliest>
  <latest>$selection.latest$</latest>
</search>
<option name="charting.chart">column</option>
<option name="charting.legend.placement">none</option>
<option name="charting.legend.masterLegend">null</option>
<option name="charting.axisX.scale">linear</option>
<option name="charting.axisY.scale">log</option>
<option name="charting.axisTitleX.text">Selected Time Range</option>
</chart>
</panel>
</row>

<html>
<h3>Token values for the splunk_web_access selection</h3>
<table border="0" cellpadding="12" cellspacing="0">
  <tr>
    <td>
      <p><b>Time range (epoch time)</b></p>
      <p>
        $selection.earliest$:
        $selection.earliest$<br/>
        $selection.latest$:
        $selection.latest$<br/>
      </p>
    </td>
    <td>
      <p><b>Count at the begining and end of time range.</b></p>
      <p>
        $start.splunk_web_access$:
        $start.splunk_web_access$<br/>
      </p>
    </td>
  </tr>
</table>
</html>
Condition element

The `<condition>` element specifies the scope of actions based on one more conditions. The available conditions on which to base actions differ, depending on the parent element. The attributes available to the condition element vary, depending on the parent element.

- Condition (input)
- Condition (search)
- Condition (drilldown)

Condition (input)

<table>
<thead>
<tr>
<th><code>&lt;condition&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the scope of actions based on input choices. If the parent element <code>&lt;change&gt;</code> is not present, then the actions apply to all choices. The <code>&lt;condition&gt;</code> element is not available for multiselect inputs.</td>
</tr>
</tbody>
</table>

**Parent element**

- `<input>`
  - `<change>`

<table>
<thead>
<tr>
<th><code>&lt;condition&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>( <code>&lt;eval</code></td>
</tr>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>label</td>
</tr>
<tr>
<td>match</td>
</tr>
<tr>
<td>value</td>
</tr>
</tbody>
</table>

**Example**

Use conditional inputs to select preset time ranges for a search.

The token for the selected choice appears in the title for the chart. The conditional token for the selected value drives the data for the chart.

```
<form>
  <label>Use tokens with conditional input choices</label>
  <fieldset submitButton="false">
    <input type="radio" token="period_tok">
      <label>Select a time range</label>
      <choice value="-24h@h">Last 24 Hours</choice>
      <choice value="-7d@h">Last 7 Days</choice>
      <choice value="-30d@h">Last 30 Days</choice>
      <default>Last 24 Hours</default>

      <!-- set condition based on the label defined by <choice> -->
      <!-- Within each condition, specify a custom label for display -->

      <!-- Capture the selected value in the token, earliest_tok -->
      <change>
        <condition label="Last 24 Hours">
          <set token="date_label">Yesterday</set>
          <set token="earliest_tok">$value$</set>
        </condition>
        <condition label="Last 7 Days">
          <set token="date_label">Last week</set>
          <set token="earliest_tok">$value$</set>
        </condition>
  </fieldset>
</form>
```
Condition (search)

**<condition>**

Specifies a condition and behavior for when the condition is met.

**Parent elements**
### Condition (match)

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>match</td>
<td>eval</td>
<td></td>
<td>An eval expression that defines the conditions needed for actions to be executed.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<condition match="'job.resultCount' == 0">
  <set token="show_table_query">true</set>
</condition>
```

### Condition (drilldown)

**Attributes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>text</td>
<td>*</td>
<td>Specifies the search field on which to implement the drilldown, or to set or unset a token.</td>
</tr>
</tbody>
</table>

**Example**

See the example for `<set>` for using the `<condition>` tag to set a token for in-page drilldown.

See the example for `<unset>` for using multiple `<condition>` tags.
Event actions

<eval>

Executes an eval statement. An eval statement evaluates an expression and puts the results into a field. <eval> for dashboards works similarly, with some exceptions, to the SPL eval command. For more details, see eval in the Search Reference.

Parent elements

<drilldown><condition>
<search><condition>
<change><condition>

<drilldown>
<eval token="[token_name]">

<drilldown>
<condition>
<eval token="[token_name]">

<change>
<eval token="[token_name]">

<change>
<condition>
<eval token="[token_name]">

<search>
<condition>
<eval token="[token_name]">

<search>
<eval token="[token_name]">

Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>text</td>
<td></td>
<td>Token whose value is the result of the eval expression. In an &lt;eval&gt; expression, you can use either $...$ delimiters or single quote delimiters for tokens. For example, both of the following options are valid.</td>
</tr>
</tbody>
</table>
Example

<eval token="new_token">[eval expression]</eval>

**link**

Specifies a link to a destination for drilldown or for a selected input choice.

<link> can be a child tag of <change>, <drilldown>, <search>, or <condition>.

Use <link> as a child tag of <condition> when you want to configure distinct drilldown actions for specific fields or inputs. Otherwise, use <link> as a child tag of <change> or <drilldown>.

There are various ways to specify a destination for the drilldown using relative paths or a URL, as described below

**Parent elements**

- <drilldown><condition>
- <search><condition>
- <change><condition>

```xml
<drilldown>
  <link>
</drilldown>

<search>
  <condition>
    <link>
  </condition>
</search>

<change>
  <link>
</change>
```
### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| field | Field name |         | Deprecated. Use `<condition field="[field]"...>`

(<drilldown> only) Specifies which values to capture in a table from the specified column or row. Cannot be specified together with the series attribute.

Although the field attribute is supported, Splunk recommends that you specify fields with the `<condition>` tag.

| series | Series name |         | Deprecated. Use `<condition field="[field]"...>`

(<drilldown> only) Specifies which values to capture in a chart from the specified series. Cannot be specified together with the field attribute.

Although the series attribute is supported, Splunk recommends that you specify series with the `<condition>` tag.

| target | text | — | Corresponds to the target attribute of the `<a>` HTTP tag.

Specifying target for the `<link>` element overrides the value of target specified in the `<drilldown>` element.

Specify "_blank" to open the drilldown in a new window.

Specify "_self" to open the drilldown in the same window.

Specify an arbitrary string to open the drilldown in a new window. Subsequent references to this target open in this window.

### Parent element

`<drilldown>` `<condition>`

1. Use the specified view, which must be in the same path as the current dashboard.
2. Relative path to connect to a dashboard.

365
3. Relative path to connect to a form, passing in a token to populate the form.
4. Pass in the earliest and latest time range from the original search.
   (Requires use of CDATA to escape special characters.)
5. URL and query argument to pass a value to the destination page.

<table>
<thead>
<tr>
<th>Path values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>path</strong></td>
<td>A path to the destination view from the current view. Typically, you specify path as: <code>/app/app_name/</code></td>
</tr>
<tr>
<td></td>
<td>However, you can also specify a relative path, based on the app context of the source and destination views.</td>
</tr>
<tr>
<td><strong>viewname</strong></td>
<td>The name of the Splunk view you are using for a destination.</td>
</tr>
<tr>
<td><code>$dest_value$</code></td>
<td>Specifies how to capture a value from a visualization. See Drilldown event tokens for details on each visualization.</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td>Specify a URL to a web page. Use the full address, including the protocol. For example: <code>http://</code>.</td>
</tr>
<tr>
<td><strong>q</strong></td>
<td>When specifying a URL, use q to specify the value of <code>dest_value</code> in a query string to a web resource.</td>
</tr>
</tbody>
</table>

**Example**

Use `<link>` with conditional inputs to open a new page.

```html
<form>
  ...
  <fieldset>
    <input type="dropdown" token="openNewPageToken">
      <label></label>
      <default>Select a page to open</default>
      <choice value="">Select a page to open</choice>
      <choice value="manager_page">View prebuilt panels</choice>
      <choice value="splk_page">Open Splunk home page</choice>
      <change>
        <condition value="manager_page">
          <link target="_blank">
            <![CDATA[/manager/search/data/ui/panels?ns=-&pwnr=&search=&count=25]]>
          </link>
        </condition>
        <condition value="splk_page">
          <link target="_blank">
            http://splunk.com
          </link>
        </condition>
    </input>
  </fieldset>
</form>
```
<set>

Allows you to publish new global tokens that can be consumed by any other element or search within the dashboard. You typically publish tokens when using form inputs or when using drilldown.

For form inputs, specify tokens for actions to take for specific inputs.

For drilldown, specify the value to capture when clicked. The value can be set dynamically using a token.

For form inputs, `<set>` can be a child tag of `<change>` or `<condition>`. For drilldown, `<set>` can be a child tag of `<drilldown>` or `<condition>`.

Use `<set>` as a child tag of `<condition>` when you want to configure distinct actions for specific inputs or for fields for drilldown. Otherwise, use `<set>` as a child tag of `<change>` or `<drilldown>` to specify an action for all inputs or for all fields.

Parent elements

<table>
<thead>
<tr>
<th>Parent elements</th>
</tr>
</thead>
</table>
There are two ways to set a value of a token.

1. Use a template to combine input tokens and static portions to form the new token value. Templates let you reference multiple tokens when setting the value, and also specify quotes for the value using the `|s` token filter.

   `<set token="Token Name">sourcetype=$click.value|s$</set>`

2. Use the prefix and suffix attributes to specify static portions for the input token. The following is equivalent to the template example above.

   `<set token="Token Name" prefix="sourcetype=" suffix="">$click.value$</set>`

### Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>Token name</td>
<td></td>
<td><strong>Required</strong> The name of the token to be consumed by the target visualization on the same page.</td>
</tr>
<tr>
<td>prefix</td>
<td>text</td>
<td></td>
<td>String to place before the value of the token.</td>
</tr>
<tr>
<td>suffix</td>
<td>text</td>
<td></td>
<td>String to append to the value of the token.</td>
</tr>
</tbody>
</table>

### Example

A click on the table sets a token which is consumed by the search of the chart visualization.

```xml
<dashboard>
  <label>In-page Drilldown</label>
  <row>
    <panel>
      <table>
        <title>Set sourcetype token on click</title>
        <search>
```
<query>
  index=_internal | stats count by sourcetype
</query>
<earliest>-1h</earliest>
<latest>now</latest>
</search>
<drilldown>
  <condition field="sourcetype">
    <set token="sourcetype">$click.value2$</set>
  </condition>
</drilldown>
</table>
<chart>
  <title>Chart for $sourcetype$</title>
  <search>
    <query>
      index=_internal sourcetype=$sourcetype$ | timechart count by sourcetype
    </query>
    <earliest>-1h</earliest>
    <latest>now</latest>
  </search>
</chart>
</panel>
</row>
</dashboard>

<unset>
Use <unset> to remove a token that was previously set.

Parent element

<change>
  <condition>
  <drilldown>
    <condition>
    <change>
    <drilldown>

<unset token="Token Name">

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>

369
<table>
<thead>
<tr>
<th><strong>token</strong></th>
<th><strong>Required</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>token name</strong></td>
<td>The name of a token that was previously set, but to be ignored.</td>
</tr>
</tbody>
</table>

**Example**

Use `<set>` and `<unset>` to define the visualization to use.

Use token definitions to hide a panel.

```xml
<dashboard>
  <label>Example for `<set>` and `<unset>`</label>
  <row>
    <panel>
      <table>
        <title>Set sourcetype token</title>
        <search>
          <query>
            index=_internal | stats count by sourcetype
          </query>
          <earliest>-1h</earliest>
          <latest>now</latest>
        </search>
        <drilldown>
          <!-- For the sourcetype field clicked: -->
          <!-- Set token to display a chart -->
          <!-- Unset token to display a table -->
          <condition field="sourcetype">
            <set token="sourcetype">$row.sourcetype$</set>
            <set token="showChart">foo</set>
            <unset token="showTable"></unset>
          </condition>
          <!-- For any other field clicked: -->
          <!-- Set token to display a table -->
          <!-- Unset token to display a chart -->
          <condition field="*">
            <set token="sourcetype">$row.sourcetype$</set>
            <set token="showTable">foo</set>
            <unset token="showChart"></unset>
          </condition>
        </drilldown>
        </table>
    </panel>
    <!-- Hide the html panel when either token is present -->
    <!-- Click in the original table to set either token -->
    <panel>
      <html rejects="$showTable$, $showChart$">
        <h2>Details</h2>
        <div style="padding: 50px; margin: 0 auto; width: 350px;">
          370
        </div>
      </html>
    </panel>
  </row>
</dashboard>
```
Token reference

Tokens are a type of variable that can be used to pass values in a simple XML dashboard. This reference lists the types of tokens available for various scenarios.

See Token usage in dashboards for further details on token usage.

<table>
<thead>
<tr>
<th>Token Type</th>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
</table>

371
<table>
<thead>
<tr>
<th><strong>Form input</strong></th>
<th><code>&lt;input&gt;</code></th>
<th>User defined input for referencing the value selected from an input. See Define tokens for form inputs. Form input example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time picker input</strong></td>
<td><code>&lt;input type=&quot;time&quot;&gt;</code></td>
<td>Optional user defined input token to associate two or more time pickers with multiple panels in a dashboard. Contains the <em>earliest</em> and <em>latest</em> modifiers to capture the time range. See Define tokens for time inputs. Time input example</td>
</tr>
<tr>
<td><strong>Drilldown event</strong></td>
<td><code>&lt;drilldown&gt;</code></td>
<td>Predefined tokens to capture the value from a click in a chart. Dynamic drilldown operations use the captured value from the source chart when accessing the drilldown target. See Drilldown event tokens for a list of the predefined tokens. See Define tokens for drilldown.</td>
</tr>
<tr>
<td><strong>Pan and zoom event</strong></td>
<td><code>&lt;selection&gt;</code></td>
<td>Predefined tokens to capture a range of values for a pan and zoom operation. The token values apply to a user selection on the chart. The context of the tokens is only for the chart. Copy the token values into user defined tokens to access the values in the dashboard. <em>start</em> and <em>end</em> capture the values of the X-axis of a chart for the beginning and end of the selected area. For example, a selection in a time chart captures the starting and ending time of the selection.</td>
</tr>
</tbody>
</table>

372
start.<field> and end.<field> capture the values of the Y-axis of a chart at the beginning and end of the selected area. For example, a selection in a time chart captures the number of events for the series specified by <field>.

See Define tokens for pan and zoom chart controls.

Define tokens for pan and zoom chart controls contains an example using a time chart.

| Conditional drilldown action | <drilldown> <condition> <link> <set>|<unset> |
|-----------------------------|--------------------------------------------------|
| User defined token within a condition element to configure conditional operations. Conditional operations include: |
| • Set token values based on the condition. |
| • Select a value for a multivalue fields in a visualization. |
| • Select a view to open based on a token value. |
| • Hide or show panels based on conditions. |
| See Define tokens for conditional operations with the <drilldown> element. |

| Conditional form input action | <input> <change> <condition> <link> <set>|<unset> |
|-------------------------------|--------------------------------------------------|
| User defined token within a condition element to modify searches or select which visualization to display based on the conditional value of a token. |
| See Define tokens for conditional operations with form inputs. |
| Conditional operations with form inputs example |

| Set destination | <input>|<drilldown> <condition> |
|-----------------|----------------------------------|
| Set and unset tokens to specify a target page to open. |
| action | <link><set>|<unset> | Can be used with the <input> element or <drilldown> element. The <condition> element defines the condition for the action. The <link> element consumes the token to open the target destination. |

**Customize Simple XML**

Splunk Enterprise users can extend Simple XML to incorporate custom CSS and JavaScript into a dashboard.

For more information, see Modify dashboards using Simple XML on the Splunk developer portal.