Creating Dashboards

Document Usage Guidelines

- Should be used only by enrolled students
- Not meant to be a self-paced document, an instructor is needed
- Do not distribute

Course Prerequisites

Classes

- Splunk Fundamentals 1 & 2
- Splunk Fundamentals 3 (suggested)

Skills

- Working knowledge of XML
- Experience with HTML, CSS, and JavaScript

Important

In order to receive credit for this course, you must complete all lab exercises.

Course Goals

- Apply best practices when creating views
- Capture and access dynamic values
- Optimize dashboard performance
- Create well formed, global searches
- Customize dashboard appearance
- Create dynamic drilldowns and behaviors
- Troubleshoot views

Course Outline

Module 1: Creating a Prototype

Module 2: Using Forms

Module 3: Improving Performance

Module 4: Customizing Dashboards

Module 5: Using Drilldowns

Module 6: Adding Advanced Behaviors & Visualizations

Course Scenario

- As in the other Splunk courses, the use cases in this course are based on a gaming company called Buttercup Games
- The views are based on business analytics from web server logs and lookups

Data	Host	Sourcetype	Interesting Fields
Online transactions & web server	www1	access_combined	action, bytes, categoryId, clientip, itemId, JSESSIONID, price, productId, product_name, referer, referer_domain, sale_price, status, user, useragent
	www2		
	www3		
Retail sales data	vendorUS1	vendor_sales	AcctID, categoryId, product_name, productId, sale_price, Vendor, VendorCity, VendorCountry, VendorID, VendorStateProvince
Server access data	www1	linux_secure	action, app, dest, process, src_ip, src_port, user, vendor_action
	www2		
	www3		
	mailsv1		

Callouts

Scenarios

- Many of the examples in this course relate to a specific scenario
- For each example, a question is posed from a colleague or manager at Buttercup Games

Scenario

How can we link multiple views to investigate data from different perspectives?

Notes & Tips

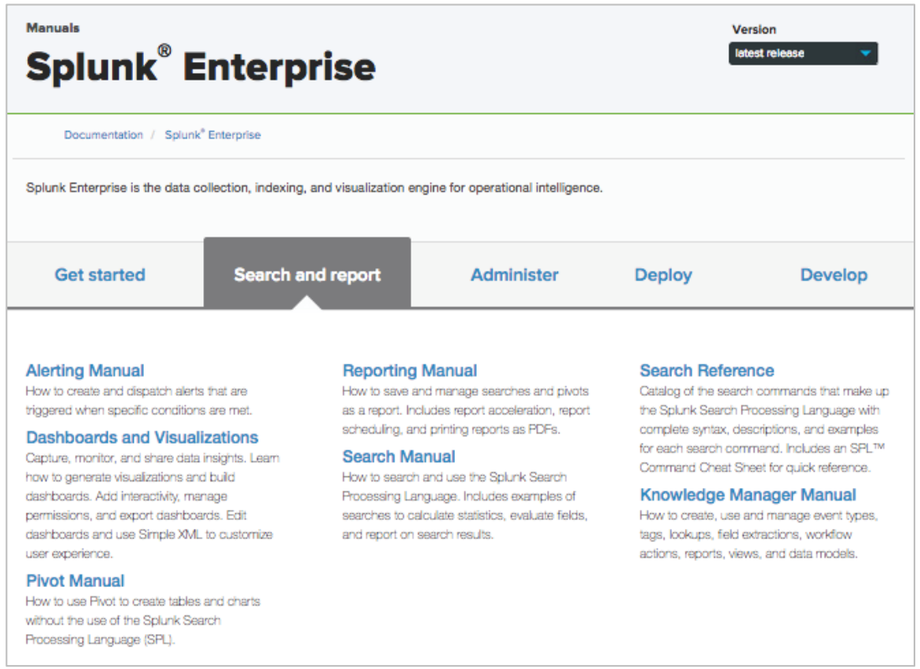
- References for more information on a topic and tips for best practices

Note

Functions and arguments used with stats and chart can also be used with timechart.

Useful References

- Dashboards & Visualizations Manual: docs.splunk.com/Documentation/Splunk/latest/Viz
- Dashboards Quick Reference: splunk.com/pdfs/solution-guides/splunk-dashboards-quick-reference-guide.pdf
- Search Quick Reference: splunk.com/content/dam/splunk2/pdfs/solution-guides/splunk-quick-reference-guide.pdf
- Reporting Manual: docs.splunk.com/Documentation/Splunk/latest/Report/Createandeditreports
- Knowledge Manager Manual: docs.splunk.com/Documentation/Splunk/latest/Knowledge/WhatisSplunkknowledge



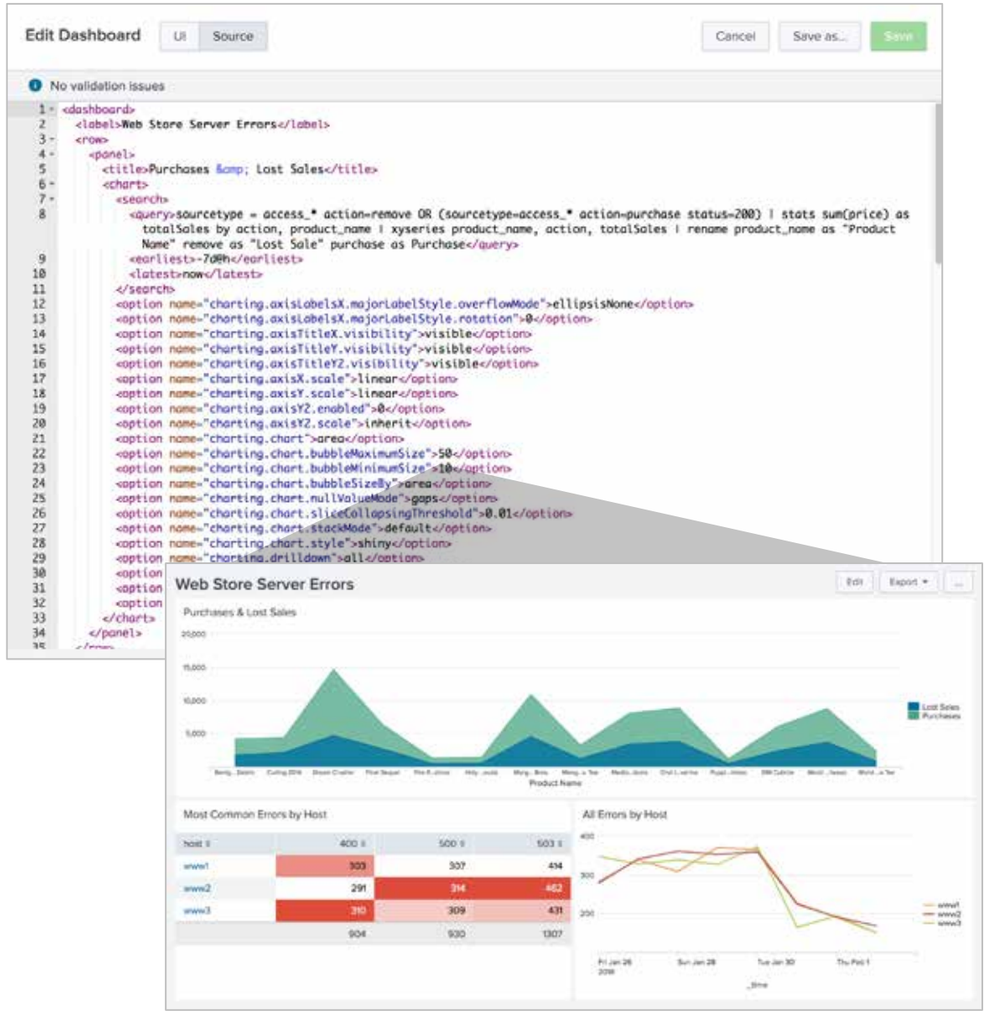
Module 1:
Creating a Prototype

Module Objectives

- Name the components of a view
- Compare dashboard and form simple XML syntax
- Troubleshoot views
- Use best practices for creating views
- Identify the primary transforming commands

What is a view?

- Every page in Splunk Web is a view
  - Dashboards
  - Forms
- Each view is a web page built from:
  - Simple XML file that defines the content
  - HTML file that defines the layout
  - CSS and JavaScript files that define the appearance and interactions

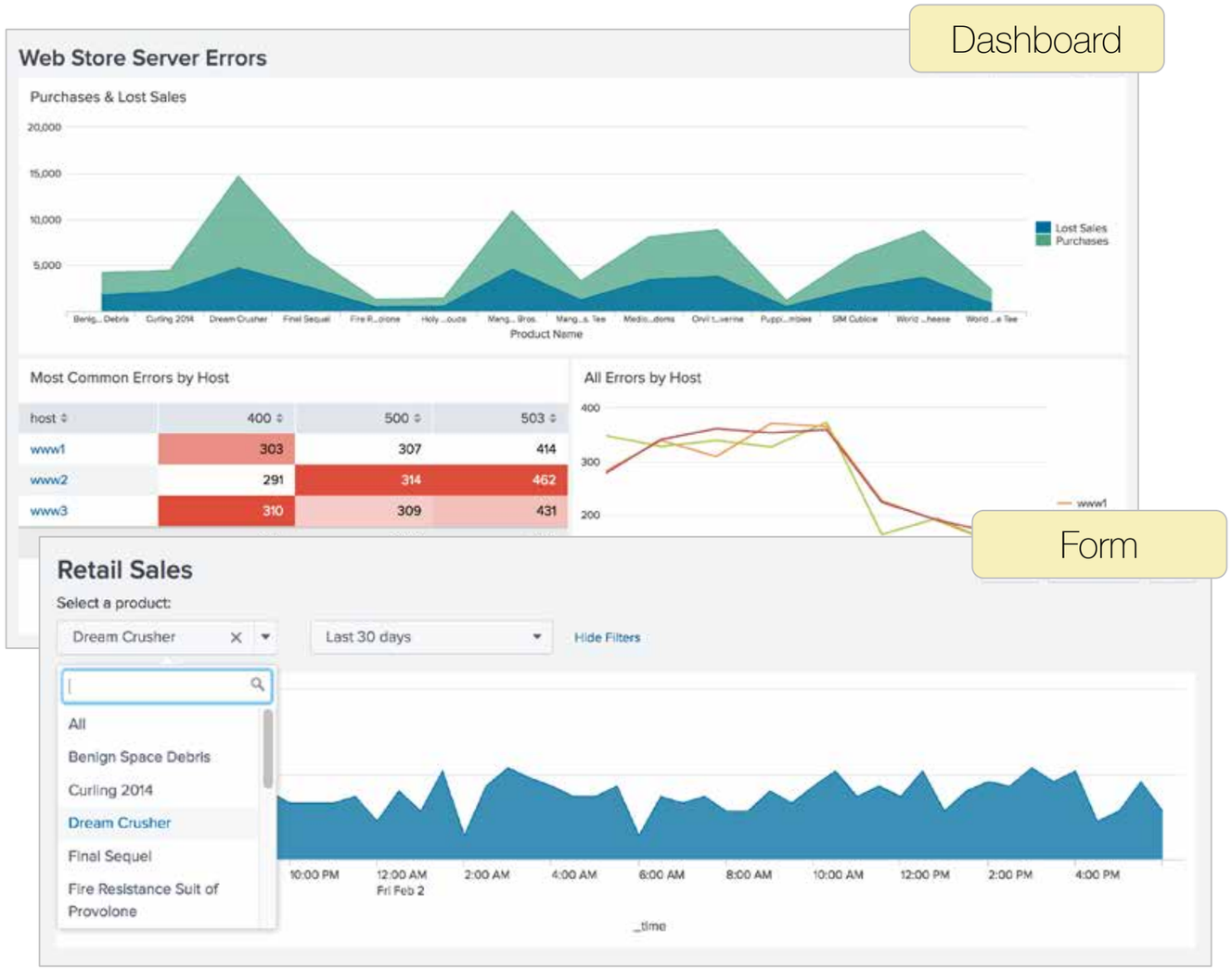


Note

A form is a dashboard with at least one user input. All views, those with or without inputs are commonly referred to as dashboards.

Dashboards & Forms

- Dashboard
  - Most common type of view
  - Limited user input
  - Default interactive features
- Form
  - Enter values from a variety of inputs
  - User input stored as tokens



Dashboards & Forms – (cont.)

```
<dashboard>
  <label>Sample Dashboard</label>
  <description>This is a basic dashboard.</description>
  <row>
    <panel>
      <title>Top Sourcetypes (Last 24 hours)</title>
      <table>
        <search>
          <query>index=_internal | top limit=100 sourcetype
          | eval percent = round(percent,2)</query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="count">20</option>
        <option name="dataOverlayMode">none</option>
        <option name="drilldown">none</option>
        <option name="percentagesRow">false</option>
        <option name="rowNumbers">false</option>
        <option name="totalsRow">false</option>
        <option name="wrap">true</option>
      </table>
    </panel>
  </row>
</dashboard>
```

Root Element

Layout of rows, panels, and panel visualizations

```
<form>
  <label>Sample Form</label>
  <description>This is a basic form.</description>
  <fieldset submitButton="false">
    <input type="time" token="time_tok" searchWhenChanged="true">
      <label></label>
      <default>
        <earliest>-24h@h</earliest>
        <latest>now</latest>
      </default>
    </input>
  </fieldset>
  <row>
    <panel>
      <title>Top Sourcetypes</title>
      <table>
        <search>
          <query>index=_internal | top limit=100 sourcetype
          | eval percent = round(percent,2)</query>
          <earliest>$time_tok.earliest$</earliest>
          <latest>$time_tok.latest$</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="count">20</option>
        <option name="dataOverlayMode">none</option>
        <option name="drilldown">none</option>
        <option name="percentagesRow">false</option>
        <option name="rowNumbers">false</option>
        <option name="totalsRow">false</option>
        <option name="wrap">true</option>
      </table>
    </panel>
  </row>
</form>
```

Form Input

Simple XML Syntax – Dashboard

```
<dashboard>
 <label>Your Dashboard Title</label>
 ...
</dashboard>
```

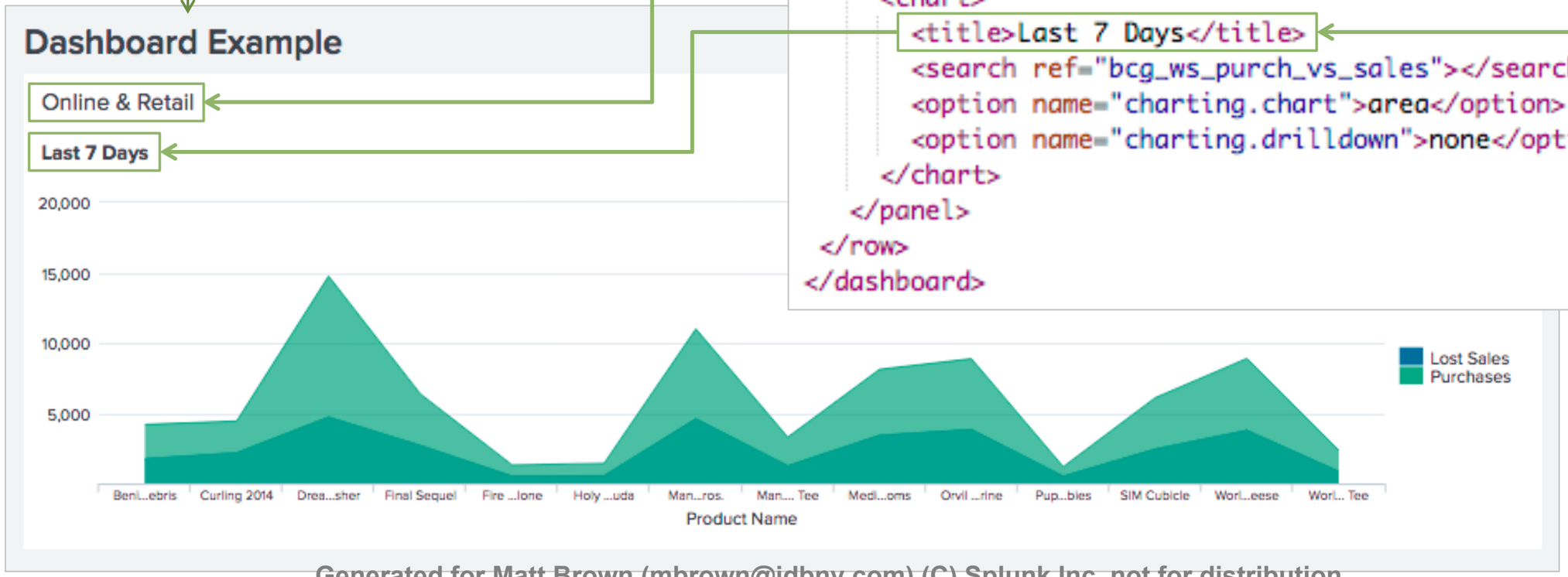
Root Element

Label

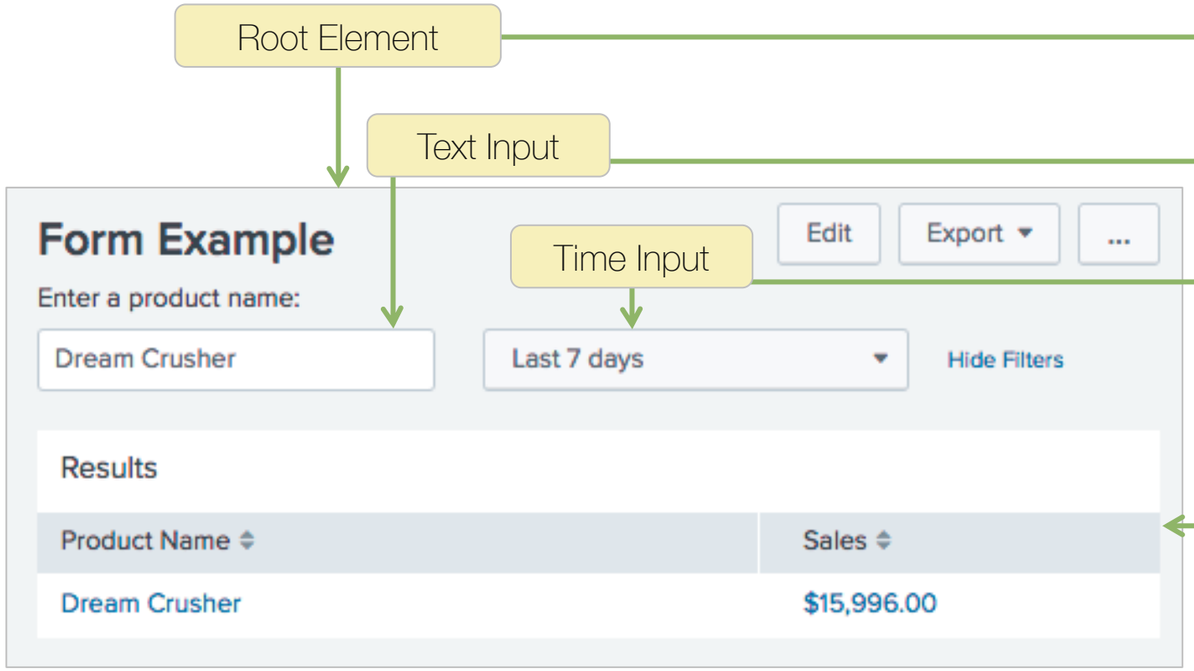
Panel Title

Chart Title

```
<dashboard>
 <label>Dashboard Example</label>
 <row>
   <panel>
     <title>Online &amp;amp; Retail</title>
     <chart>
       <title>Last 7 Days</title>
       <search ref="bcg_ws_purch_vs_sales"></search>
       <option name="charting.chart">area</option>
       <option name="charting.drilldown">none</option>
     </chart>
   </panel>
 </row>
</dashboard>
```



Simple XML Syntax – Form



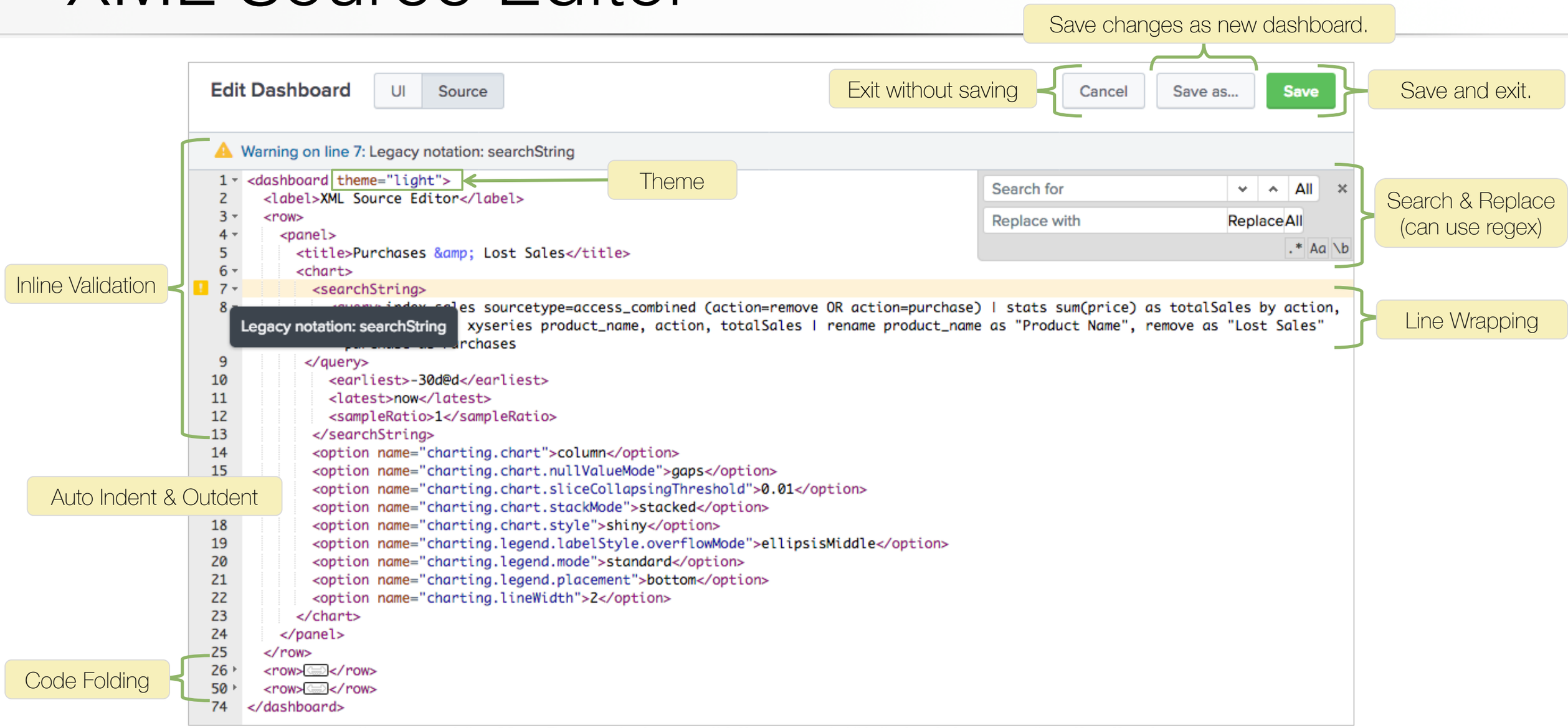

```
<form>
 <label>Your Form Title</label>
 <fieldset>
  <input type="text" token="myToken">
  <label>Enter text:</label>
 </fieldset>
 ...
</form>
```

```
<form>
 <label>Form Example</label>
 <fieldset autoRun="true" submitButton="false">
  <input type="text" token="p_name_tok" searchWhenChanged="true">
   <label>Enter a product name:</label>
   <default>*</default>
  </input>
  <input type="time" token="time_tok">
   <label></label>
   <default>
    <earliest>-7d@d</earliest>
    <latest>now</latest>
   </default>
  </input>
 </fieldset>
 <row>
  <panel>
   <table>
    <title>Results</title>
    <search>
     <query>sourcetype=vendor_sales product_name="$p_name_tok$"
     | stats sum(price) as Sales by product_name
     | fieldformat Sales = "$" + tostring(Sales, "commas")
     | rename product_name as "Product Name"</query>
     <earliest>$time_tok.earliest$</earliest>
     <latest>$time_tok.latest$</latest>
    </search>
    <option name="count">50</option>
    <option name="dataOverlayMode">none</option>
    <option name="drilldown">cell</option>
    <option name="rowNumbers">false</option>
    <option name="wrap">true</option>
   </table>
  </panel>
 </row>
</form>
```

Search to perform with token for product name: `$p_name_tok$`

Tokens for time range:
`$time_tok.earliest$`
`$time_tok.latest$`

XML Source Editor



Escaping Special Characters

- Some characters have special meaning in an XML file and cannot be used literally
- Wrap any text with special characters within CDATA tags

```
<![CDATA[ "Text within tags" ]]>
```

Character	Replacement Entity
"	&quot;
'	&apos;
<	&lt;
>	&gt;
&	&amp;

```
...
<condition match="$job.resultCount$ &gt;= 5">
   <set token="result_count_condition">Matched!</set>
</condition>
...
```

This conditional match statement uses the greater than (>) entity.

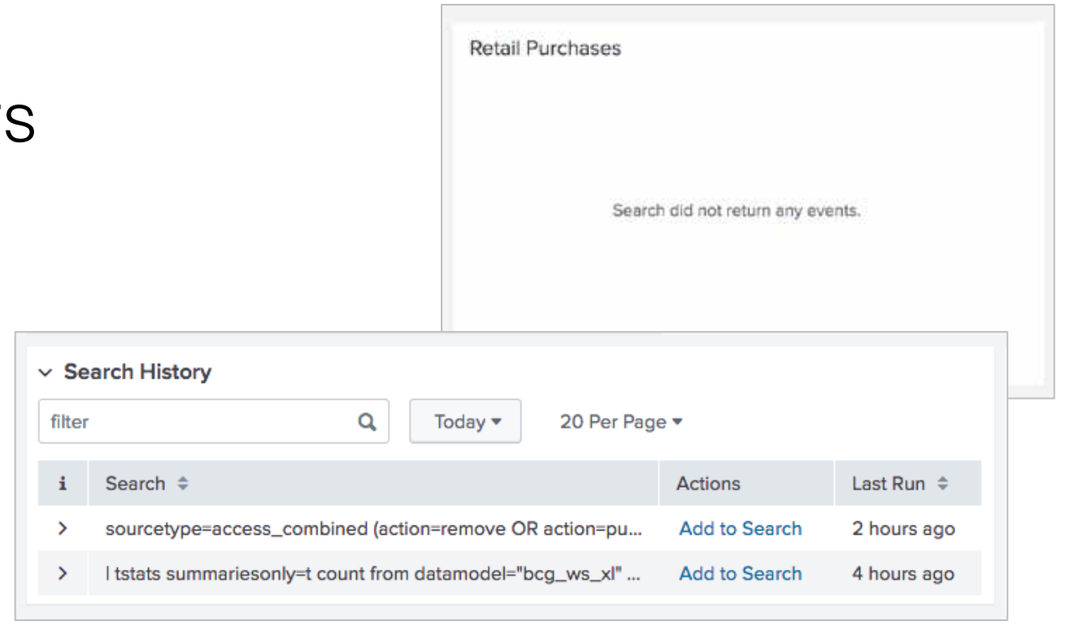
CDATA Tags

```
...
<link>
<![CDATA[
 /app/search/form_for_drilldown?form.sourcetype=$row.sourcetype$ & earliest=$earliest$ & latest=$latest$
 ]]>
</link>
...
```

The search in this dynamic drilldown has ampersands

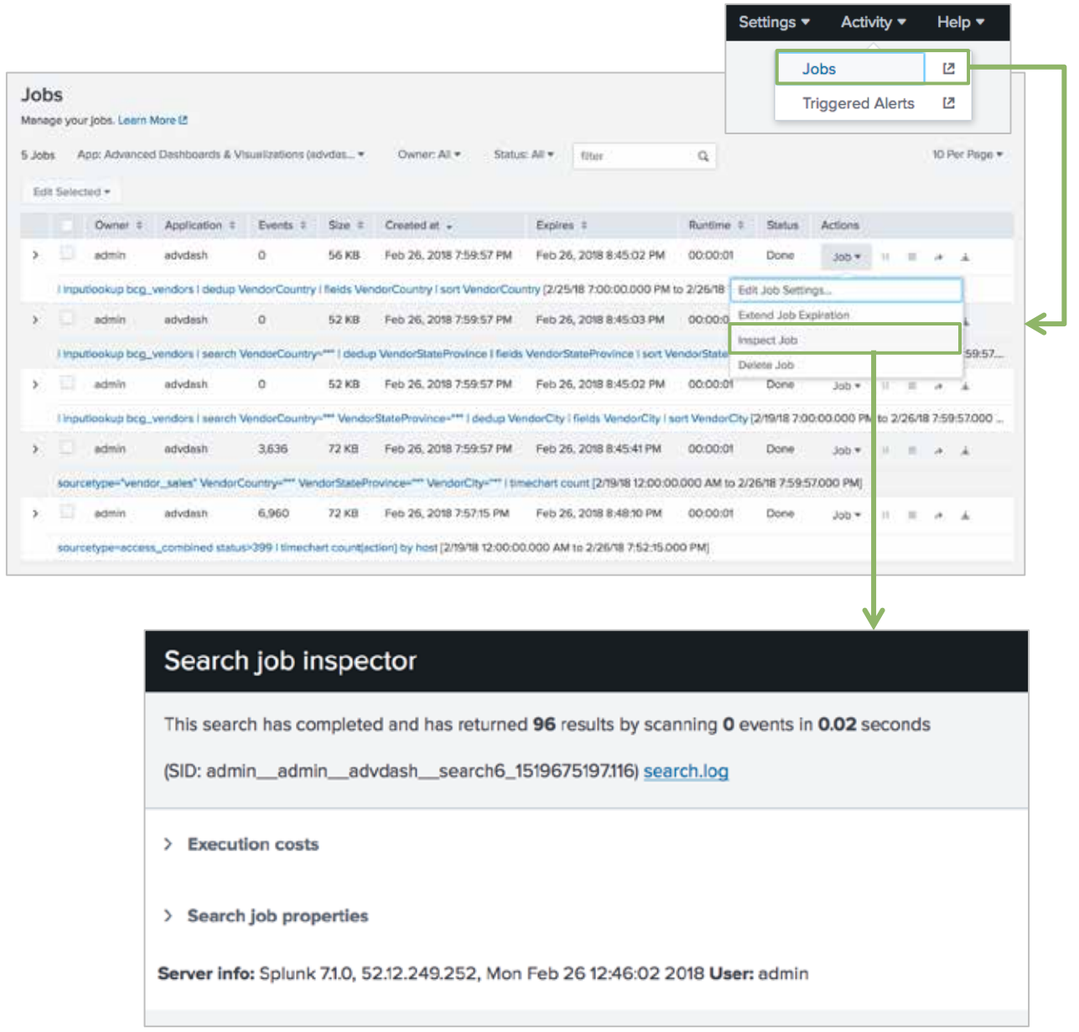
Troubleshooting Views

- Examine the view's source
  - Check for search and XML syntax errors
  - Run search manually
- View all previous searches
  - Examine your search history
  - Run `| history`
- Expand macros and event types
- Verify the tokens are being set and have the expected values
  - Create an html panel that shows all your tokens
  - Use the **View what tokens are available** dashboard in the Splunk Dashboard Examples app



Troubleshooting Views (cont.)

- Job Inspector
  - Examine impact of knowledge object processing, such as event types, tags, lookups etc.
- Search Job Inspector
  - Debug messages

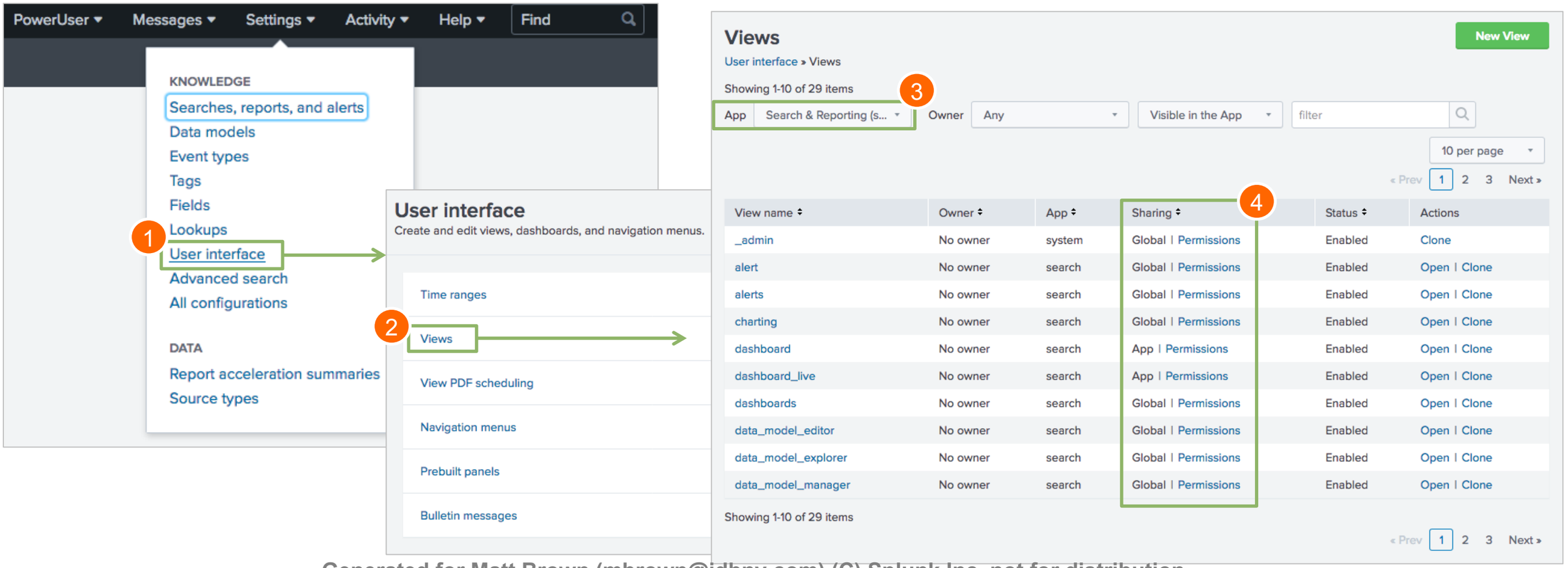


Note

You won't see debug messages until the search has completed.

Managing Views

Views are scoped to your app context and permissions can be applied to them



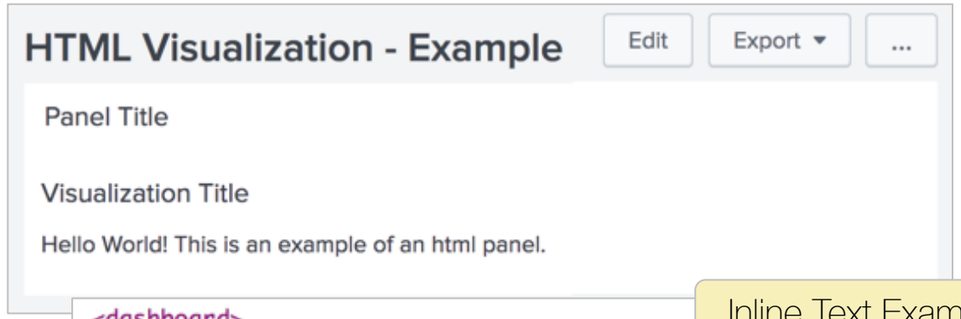
Panels


```
<dashboard>
  <label>Panel Example</label>
  <row>
    <panel>
      <title>Panel - Title</title>
        <chart>
        <title>Visualization Title</title>
          <search>
           <query>sourcetype=access_combined
            | timechart count by categoryId usenull=f</query>
           <earliest>-7d@d</earliest>
           <latest>now</latest>
           <sampleRatio>1</sampleRatio>
          </search>
        </chart>
    </panel>
  </row>
</dashboard>
```

```
<panel>
 <title>
  <chart> | <event> | <html> | <map> | <single> | <table> | <viz>
   <title>
    <search>
     . . .
```

Panels (cont.)

- HTML Panels
  - Add inline text
  - Reference text or image files
  - Reference a file from another app
  - Use HTML elements: <h1>, <h2>, <h3>, , <img>, <p>, etc.



Inline Text Example

```
<dashboard>
 <label>HTML Visualization - Example</label>
 <row>
   <panel>
     <title>Panel Title</title>
     <html>
       <h2>Visualization Title</h2>
         <p>Hello World! This is an example of an html panel.</p>
     </html>
   </panel>
 </row>
</dashboard>
```

HTML File

```
<html src="myFile.html"> </html>
```

HTML file from another app

```
<html src="myApp:myFile.html"> </html>
```

Image File

```
<html> <img src="/static/app/<app_name>/images/myFile.png"> </img> </html>
```

Note

The ability to upload files is restricted to Splunk Administrators or user roles given that right by the administrator.

Panel Visualization Elements

Each panel has seven possible visualization elements

	Element	Syntax	Description
1.	Chart	`<chart>`	Search results as a chart, filler, marker and radial gauges
2.	Event	`<event>`	Search results as individual events
3.	Map	`<map>`	Search results as map
4.	Single value	`<single>`	Single value visualizations
5.	Table	`<table>`	Displays search results as a table
6.	Custom Visualization	`<viz>`	Displays a Splunk Custom Visualization
7.	HTML	`<html>`	Inline HTML. Create or edit in XML editor only

Panel Types

- Inline
- Report
- Prebuilt

Inline Panel

```
<dashboard>
  <label>Total Online Purchases</label>
  <row>
    <panel>
      <title>Total Online Purchases</title>
      <single>
        <search>
          <query>sourcetype=access* action=purchase status=200
          | stats count as purchases</query>
          <earliest>-30d@d</earliest>
          <latest>now</latest>
          <sampleRatio>1</sampleRatio>
        </search>
        <option name="drilldown">none</option>
      </single>
    </panel>
  </row>
</dashboard>
```

Report Panel

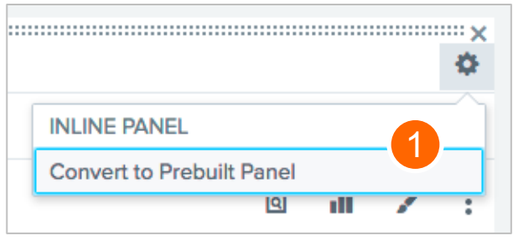
```
<dashboard>
  <label>Total Online Purchases</label>
  <row>
    <panel>
      <title>Total Online Purchases</title>
      <single>
        <search ref="Total Online Purchases"></search>
        <option name="drilldown">none</option>
      </single>
    </panel>
  </row>
</dashboard>
```

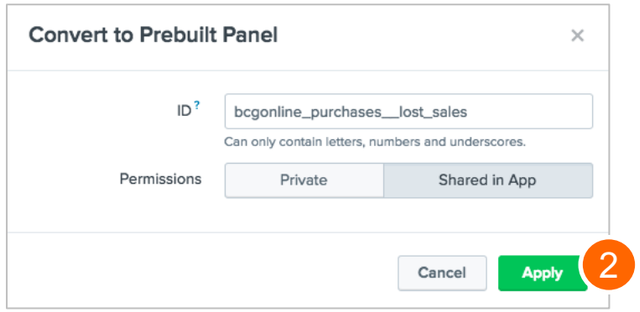
Prebuilt Panel

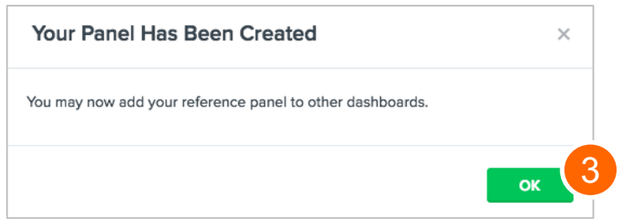
```
<dashboard>
  <label>Total Online Purchases</label>
  <row>
    <panel ref="total_online_purchases"></panel>
  </row>
</dashboard>
```

Prebuilt Panels

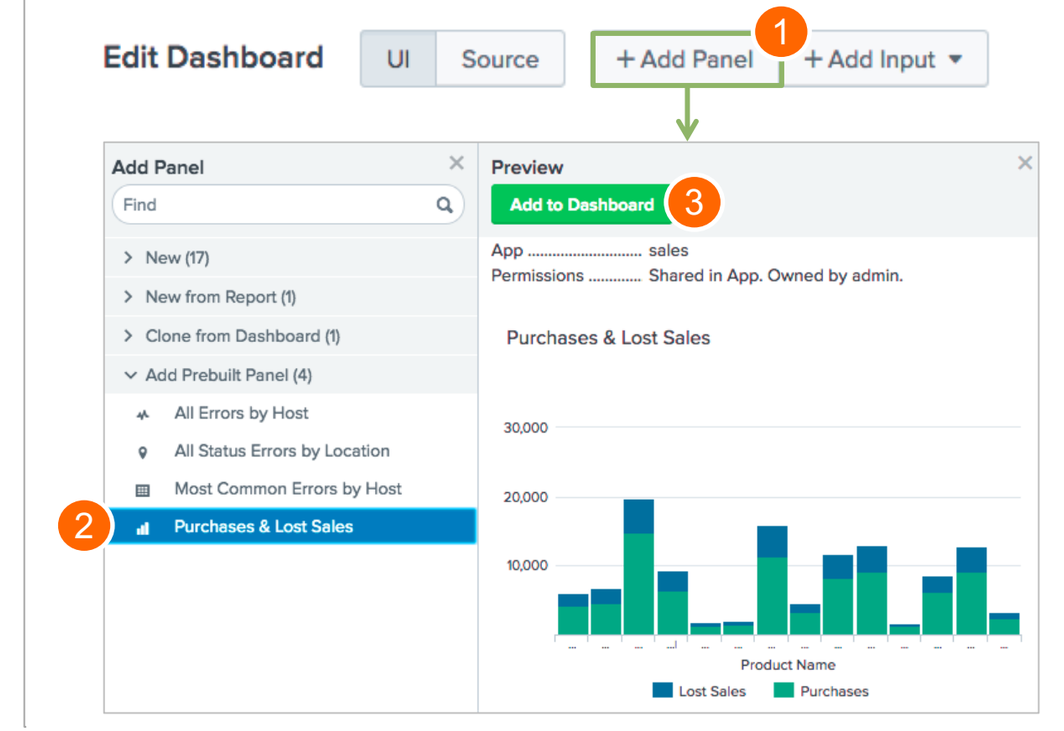
Create a Prebuilt Panel





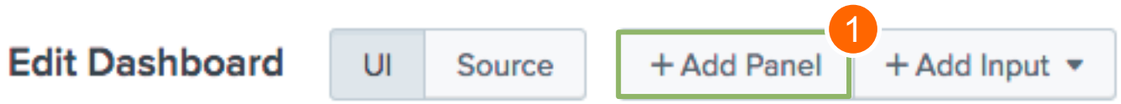


Add a Prebuilt Panel

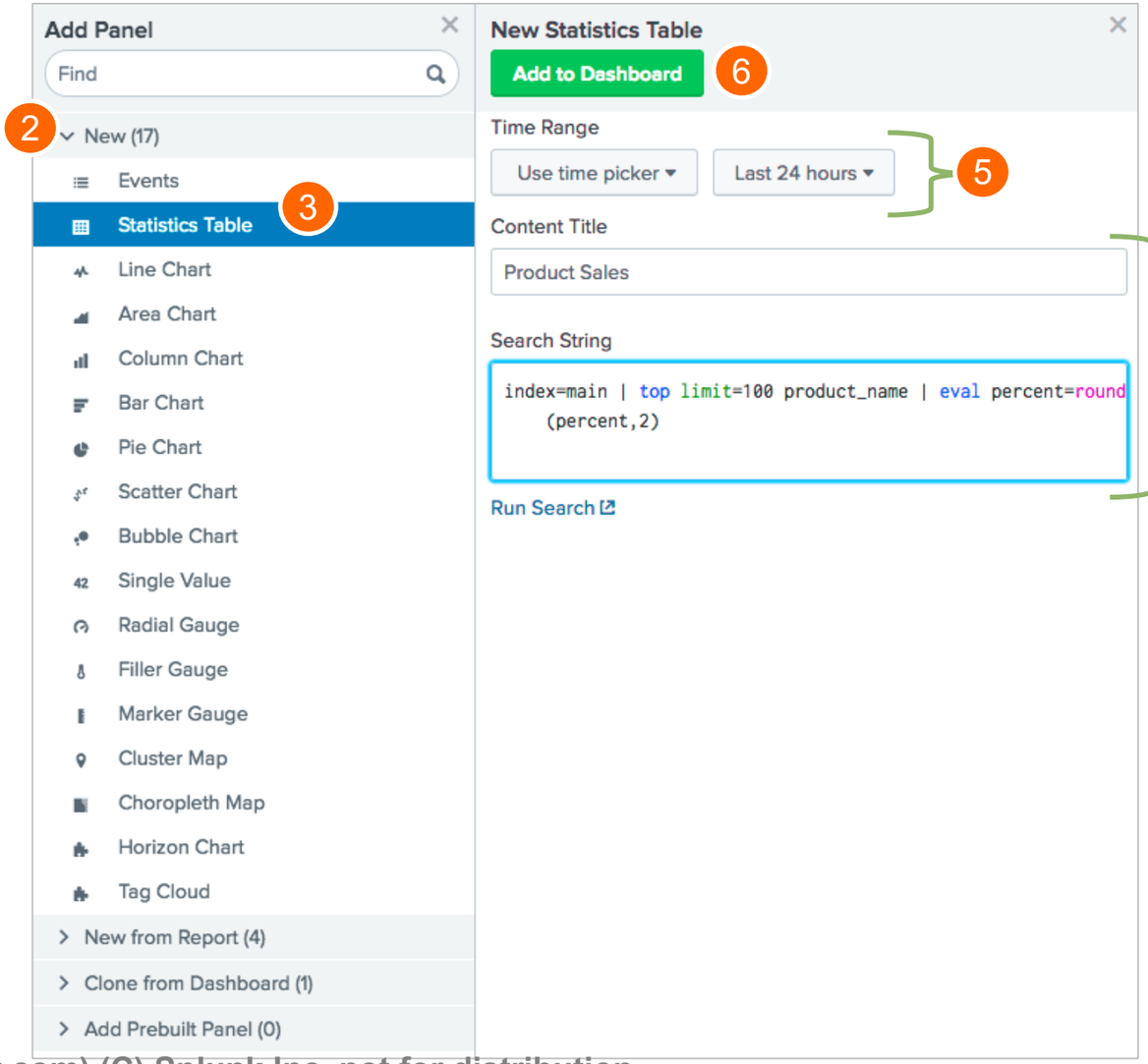


- One panel, multiple dashboards
- Complex panels available to non-technical users
- All instances referencing a panel receive updates

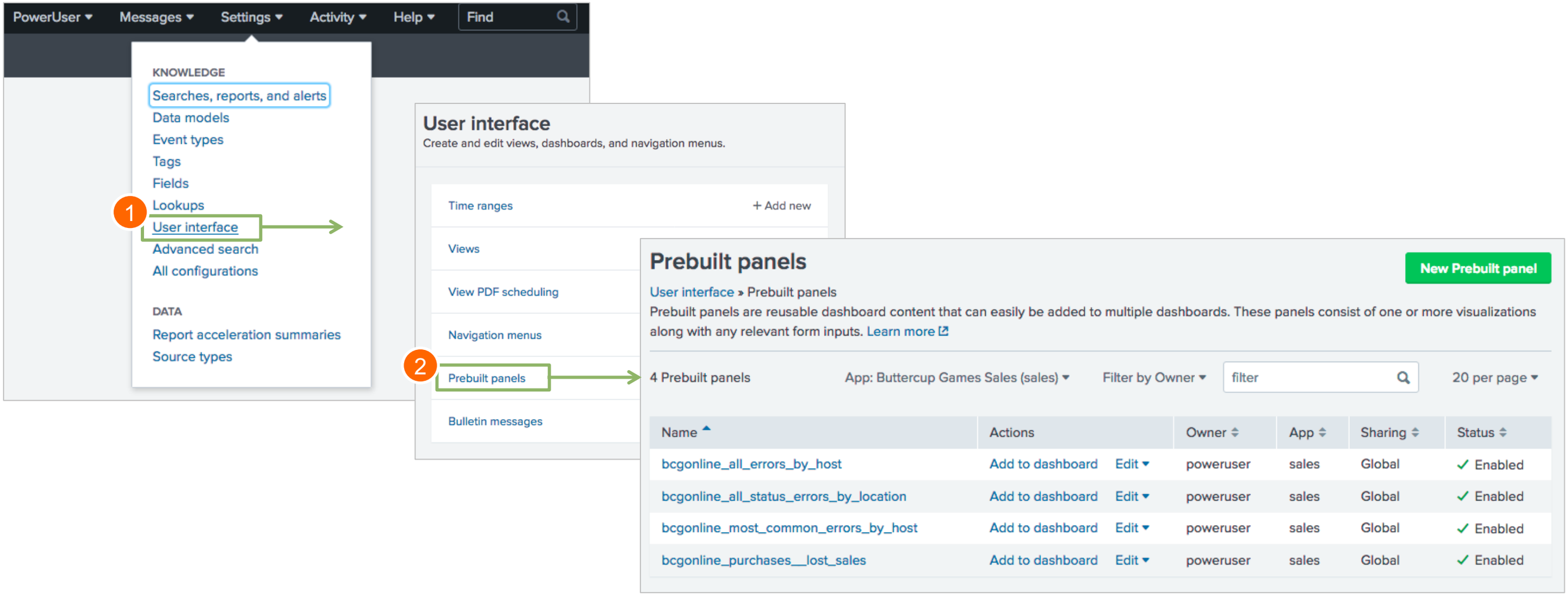
Adding Panels



- Add Panel sidebar
  - New
  - New from Report
  - Clone from Dashboard
  - Add Prebuilt Panel
- Time Range
  - Use Time Picker
  - Tokens
  - Global



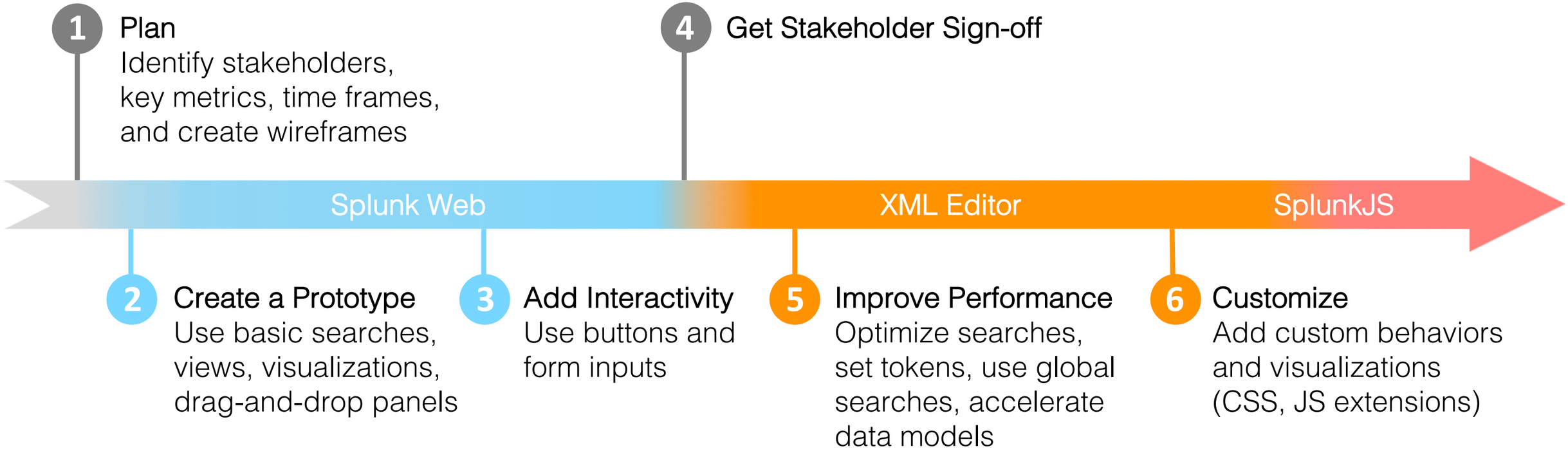
Managing Prebuilt Panels



What do you want to do?

	**Create or modify basic dashboards**	**Add advanced features, optimizations, and logic**	**Add custom stylesheets and logic**
Use this:	Dashboard Editor	Simple XML	Simple XML Extensions
Skills	Mouse click	XML	• XML • CSS • HTML • JavaScript
Benefits	• Drag-and-drop UI • PDF generation	• Complex interactivity • More efficient dashboards • Visualization and panel customizations	• Third-party libraries • Custom behaviors • Custom graphics • Custom layouts
Drawbacks	• Limited layout • Limited features	• Limited layout • Limited features	None
See this:	Modules 1, 2	Modules 3, 4, 5	Module 6

Best Practice

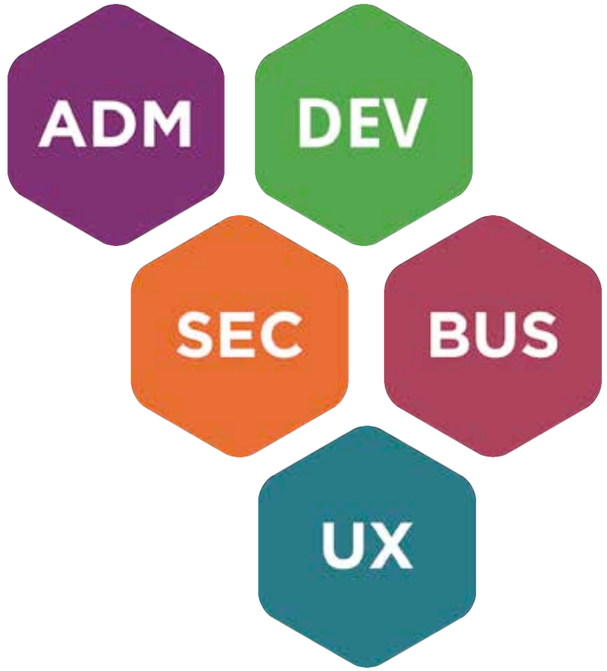


Note

Creating views with JavaScript is discussed in the Advanced Dashboards & Visualizations course.

Stakeholders

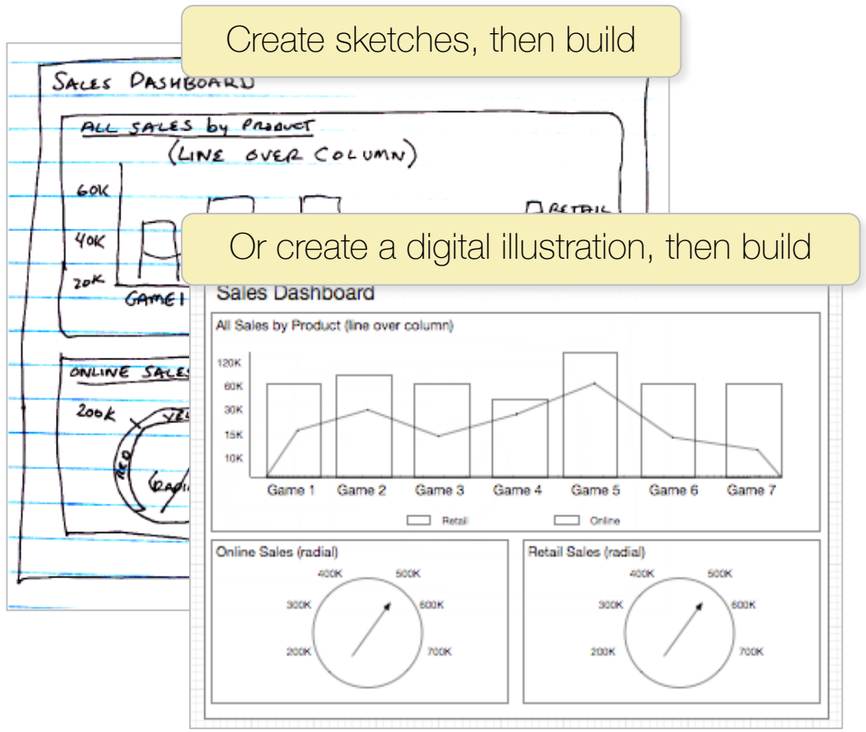
- Depending on the complexities of the view, your stakeholders may include:
  - Splunk Administrator
  - JavaScript developer
  - Security expert
  - Business user
  - UX designer



- Questions to ask:
  - How will users access your view?
  - Will the view use JavaScript or a custom stylesheet?
  - Should the view be deployed with its own app?

Plan

- An iterative process between you and the stakeholders
  - What **critical metrics** do users want?
  - What is the **time span** for the data?
  - What is the **timeframe for refreshing** data?
  - What **visualizations** will be required?
  - What should the **layout** look like?
- Wireframing is the process of designing a view including its information, navigation, and interface design



Create Basic Searches

- Start with basic inline searches
- Add reports, tokens, macros, and data models later
  - Use naming conventions
    - At least the same prefix
    - Group, search type, view type, platform, category, time interval, description, and project

Command	Description
bucket	Puts continuous numerical values into discrete sets.
chart	Returns results in a tabular output for charting.
dedup	Removes subsequent results that match a specified criteria.
eval	Calculates an expression and puts the value into a field.
fields	Adds or removes fields from search results.
lookup	Explicitly invokes field value lookups.
multikv	Extracts field-values from table-formatted events.
rangemap	Sets range field to the name of the ranges that match.
rex	Specify a Perl regular expression named groups to extract fields while you search.
spath	Extracts key-value pairs from XML or JSON formats.
stats	Provides statistics, grouped optionally by fields.
timechart	Create a time series chart and corresponding table of statistics.
top	Calculate a count and percentage for the most common values of the fields in the field list.
transaction	Groups search results into transactions.
where	Performs arbitrary filtering on your data.

Transforming Commands

- Orders the results into a data table
- Transforms specified cell values into numerical values that can be used for statistical purposes
- Primary transforming commands include:
  - chart
  - timechart
  - top
  - rare
  - stats

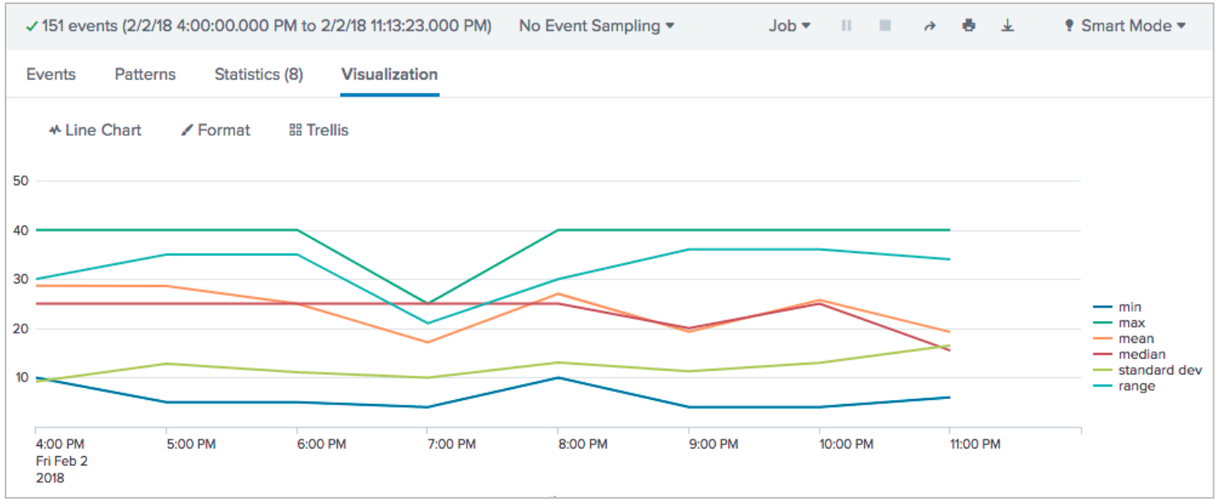
Note

In previous versions of Splunk, transforming commands were referred to as *reporting* commands.

Statistical Functions

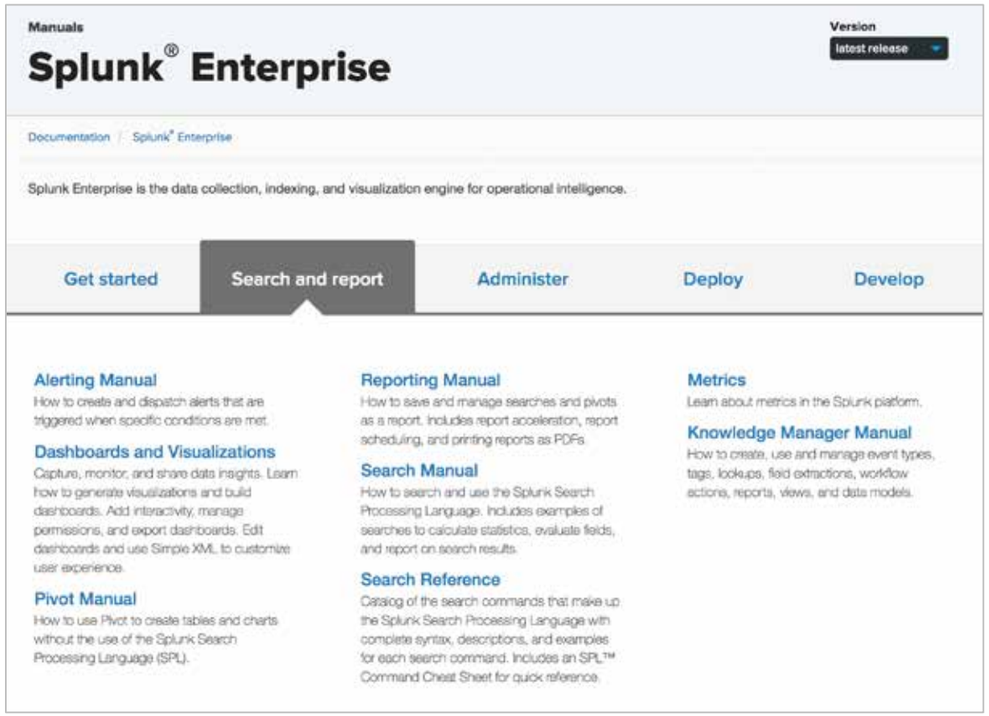
- These transforming commands work with statistical functions:
  - chart, timechart, stats, geostats
- Available statistical functions:
  - count, distinct count
  - first occurrence, last occurrence
  - mean, median, mode
  - min, max, range, percentiles
  - standard deviation, variance
  - sum

```
sourcetype=access_combined action=purchase
| timechart span=1h
  min(price) as min,
  max(price) as max,
  mean(price) as mean,
  median(price) as median,
  stdev(price) as "standard dev",
  range(price) as range
```



Useful References

- Data Structure Requirements for Visualizations
  docs.splunk.com/Documentation/Splunk/latest/Viz/Datastructurerequirementsforvisualizations
- Visualization Reference
  docs.splunk.com/Documentation/Splunk/latest/Viz/Visualizationreference
- Transforming Commands
  docs.splunk.com/Documentation/Splunk/latest/SearchReference/Commandsbytype#Transformingcommands
- Statistical and Charting Functions
  docs.splunk.com/Documentation/Splunk/latest/SearchReference/CommonStatsFunctions



Lab 1 – Create a Prototype

Time: 30 minutes

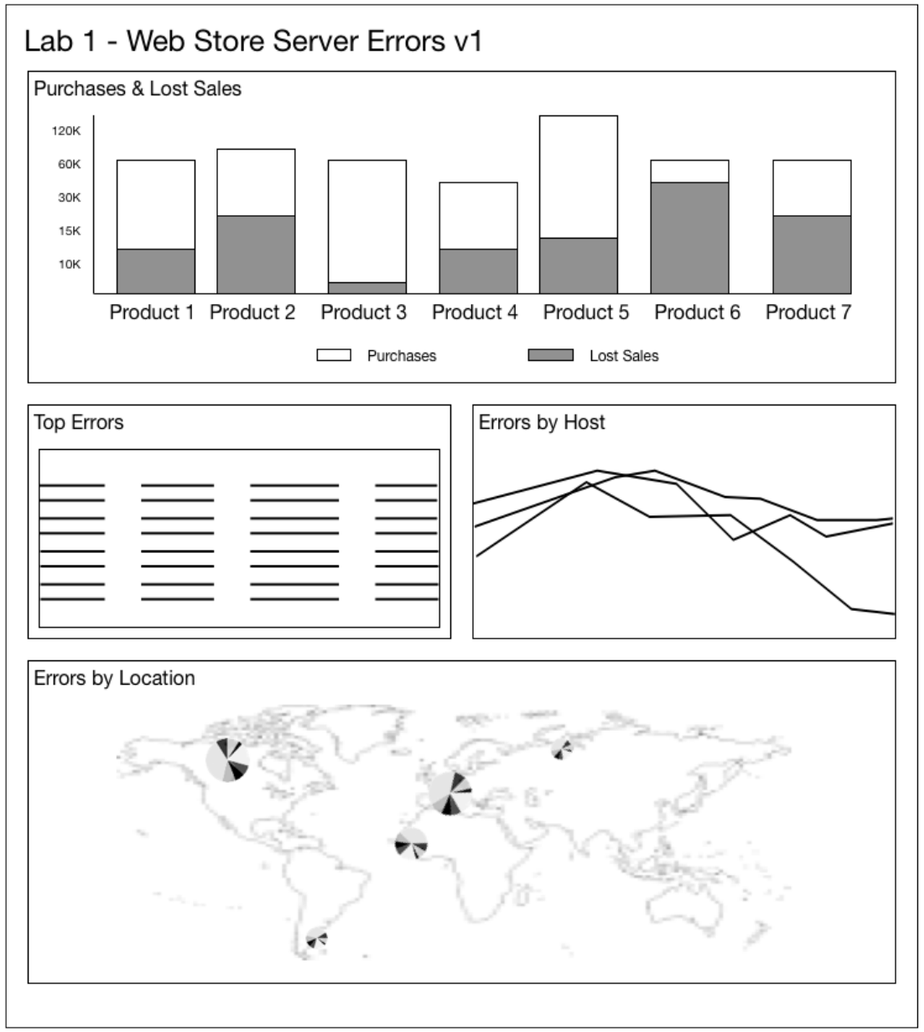
Scenario: The sales team wants a dashboard that displays information about web store server health

Tasks:

- Change the account name and time zone
- Create a dashboard
- Add table and chart panels
- Add a map
- Create prebuilt panels

Challenge Task (optional):

- Create a data model



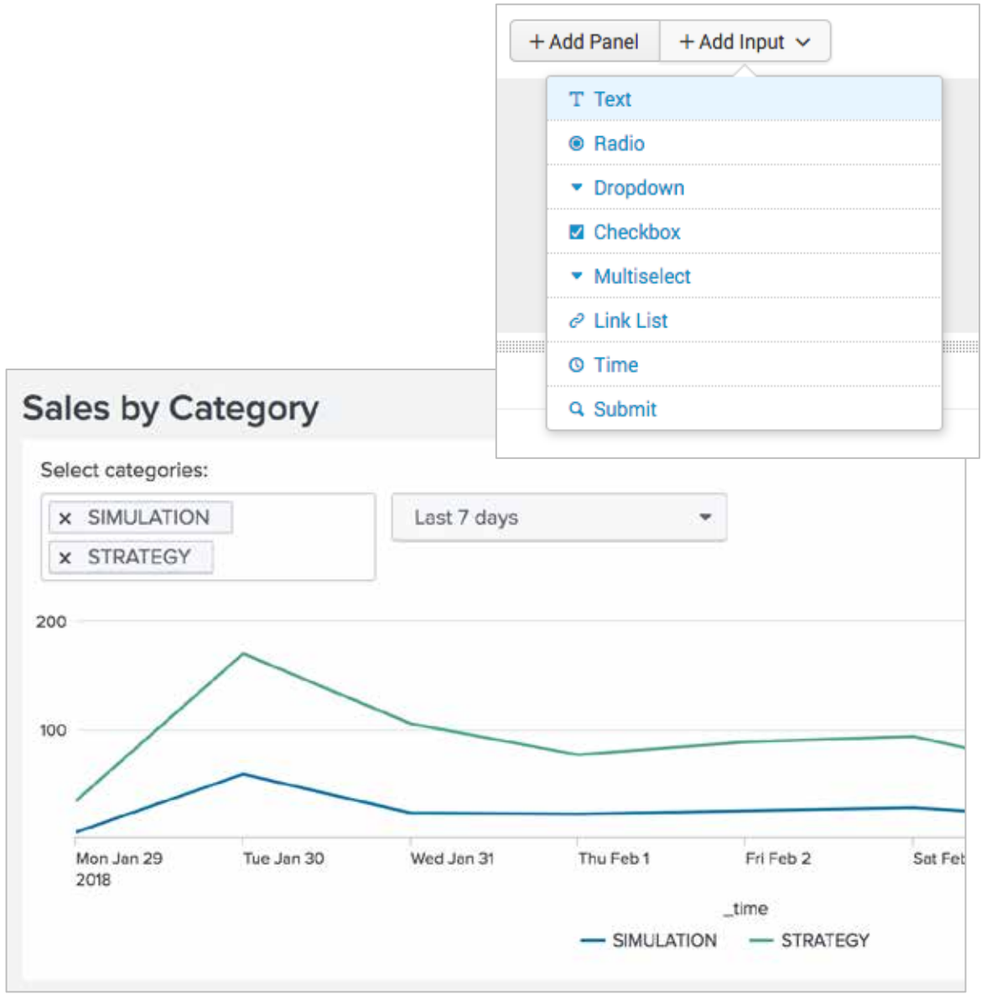
Module 2:
Using Forms

Module Objectives

- Identify types of form inputs
- Describe how tokens are created and used
- Use tokens with form inputs
- Create cascading inputs
- Define types of token filters

Form Inputs

- Adding an input changes the root element from **<dashboard>** to **<form>**
- Add to a page or panel
- User defined token for selected value
- Configure when input values populate a form:
  - On page load
  - Input change
  - Clicking Submit
- Specify default and initial values



Tokens

- Dynamically capture and pass values
  - Form input values
  - Search terms
  - Field values
- Dashboard behavior
  - Show & Hide Panels
  - Drilldowns
  - Event handlers
  - Conditional behaviors
- User defined tokens
- Predefined tokens

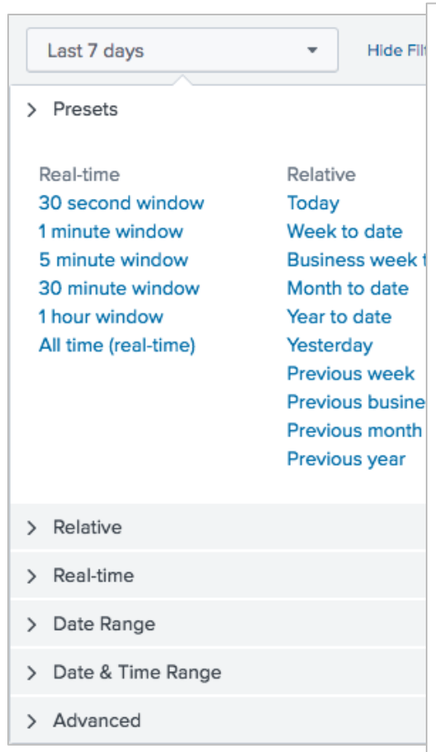
```
...
  <fieldset submitButton="false" autoRun="false">
    <input type="dropdown" token="prod_name_tok" searchWhenChanged="true">
      <label>Select a product:</label>
      <choice value="*">All</choice>
      <search>
        <query>sourcetype=vendor_sales | stats count by product_name</query>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
      </search>
      <default>*</default>
      <fieldForLabel>product_name</fieldForLabel>
      <fieldForValue>product_name</fieldForValue>
      <initialValue>*</initialValue>
    </input>
  </fieldset>
  <row>
    <panel>
      <chart>
        <title>$prod_name_tok$ Sales by City</title>
       <search>
         <query>sourcetype=vendor_sales product_name=$prod_name_tok|s$
          | stats count by VendorCity, Vendor | sort - count</query>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
       </search>
...
```

Defined here…

Used here…

…and here

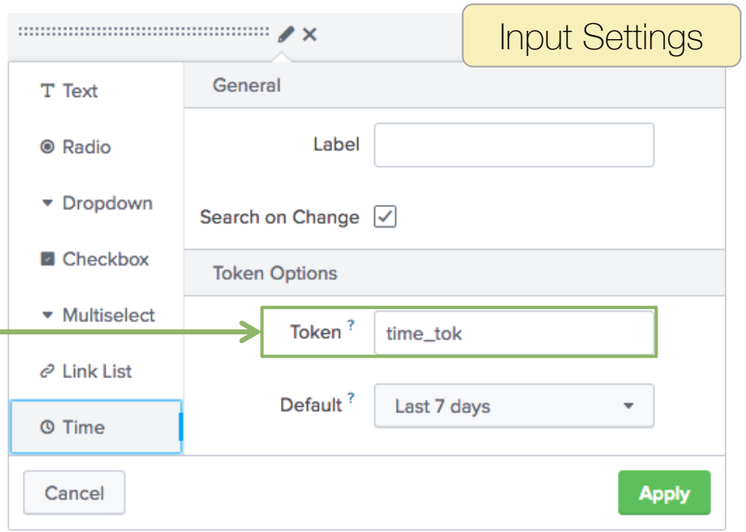
Time Input – Example 1

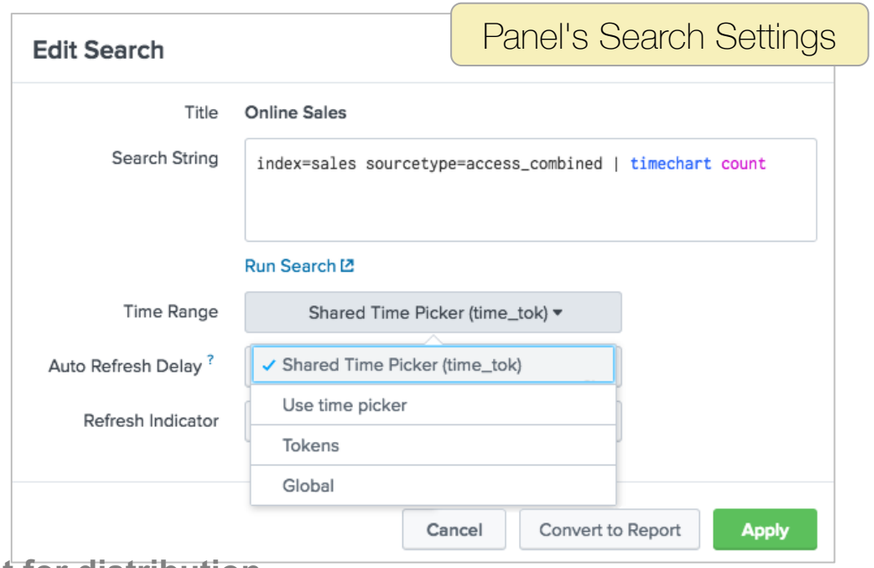


User defined token

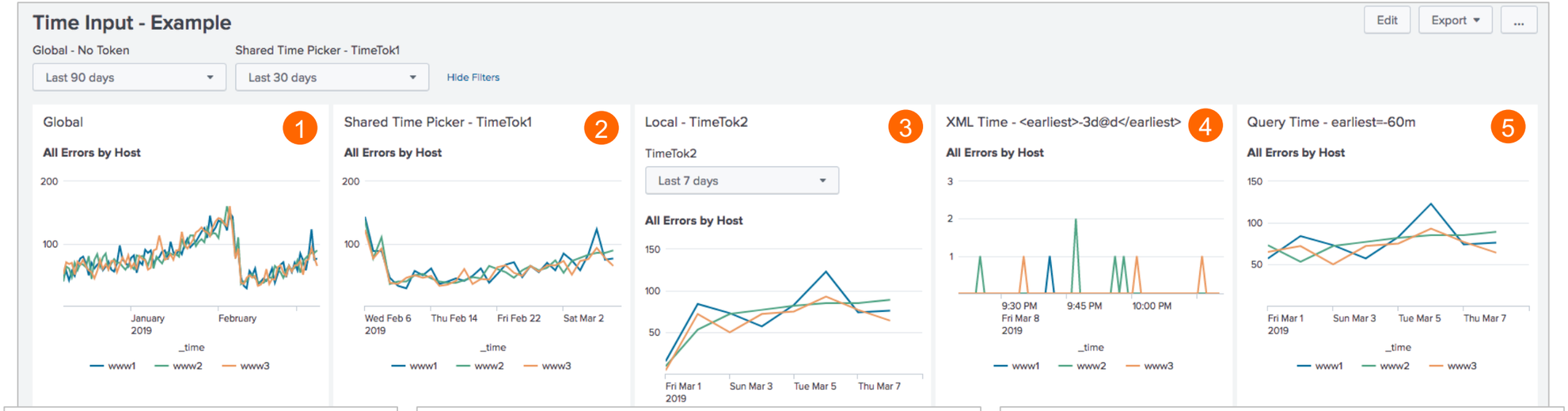
```
...
  <fieldset submitButton="false">
    <input type="time" token="time_tok" searchWhenChanged="true">
      <label></label>
      <default>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
      </default>
    </input>
  </fieldset>
  <row>
    <panel>
      <chart>
        <title>Online Sales</title>
        <search>
          <query>index=sales sourcetype=access_combined
          | timechart count</query>
          <earliest>$time_tok.earliest$</earliest>
          <latest>$time_tok.latest$</latest>
        </search>
...
```

Predefined modifiers to capture the time range





Time Input – Example 2




```
...
<input type="time"
searchWhenChanged="true">
 <label>Global - No Token</label>
  <default>
   <earliest>-90d@d</earliest>
   <latest>now</latest>
  </default>
</input>
...
<panel>
 <title>Global</title>
 <chart>
 <title>All Errors by Host</title>
 <search>
 <query>sourcetype=access_combined
  status&gt;399 | timechart count(action)
  by host
 </query>
 <sampleRatio>1</sampleRatio>
 </search>
...
```

1

```
...
<input type="time" token="TimeTok1" searchWhenChanged="true">
 <label>Shared Time Picker - TimeTok1</label>
  <default>
   <earliest>-30d@h</earliest>
   <latest>now</latest>
  </default>
</input>
...
<panel>
 <title>Shared Time Picker - TimeTok1</title>
  <chart>
   <title>All Errors by Host</title>
    <search>
      <query>sourcetype=access_combined status&gt;399
         | timechart count(action) by host</query>
      <earliest>$TimeTok1.earliest$</earliest>
      <latest>$TimeTok1.latest$</latest>
      <sampleRatio>1</sampleRatio>
    </search>
...
```

2

```
...
<title>Local - TimeTok2</title>
<input type="time" token="TimeTok2" searchWhenChanged="true">
  <label>TimeTok2</label>
   <default>
   <earliest>-7d@d</earliest>
   <latest>now</latest>
   </default>
 </input>
...
<chart>
 <title>All Errors by Host</title>
  <search>
   <query>sourcetype=access_combined status&gt;399
      | timechart count(action) by host</query>
   <earliest>$TimeTok2.earliest$</earliest>
   <latest>$TimeTok2.latest$</latest>
   <sampleRatio>1</sampleRatio>
  </search>
...
```

3

```
...
<panel>
 <title>XML Time - <earliest>-3d@d</earliest>
</title>
  <chart>
  <title>All Errors by Host</title>
  <search>
   <query>sourcetype=access_combined
    status&gt;399 | timechart
    count(action) by host
   </query>
   <earliest>-3d@d</earliest>
   <latest>now</latest>
   <sampleRatio>1</sampleRatio>
  </search>
...
```

4

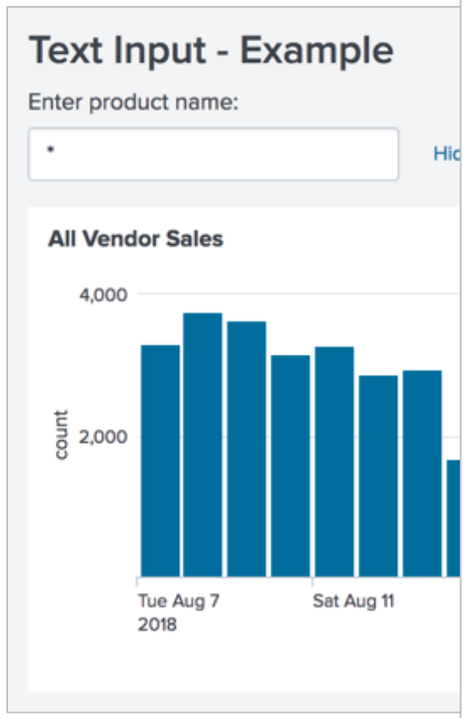
```
...
<panel>
 <title>Query Time - earliest=-60m</title>
  <chart>
  <title>All Errors by Host</title>
  <search>
   <query>sourcetype=access_combined
    status&gt;399 earliest=-60m | timechart
     count(action) by host
   </query>
   <earliest>-7d</earliest>
   <latest>now</latest>
   <sampleRatio>1</sampleRatio>
  </search>
...
```

5

Ignored

Text Input Example

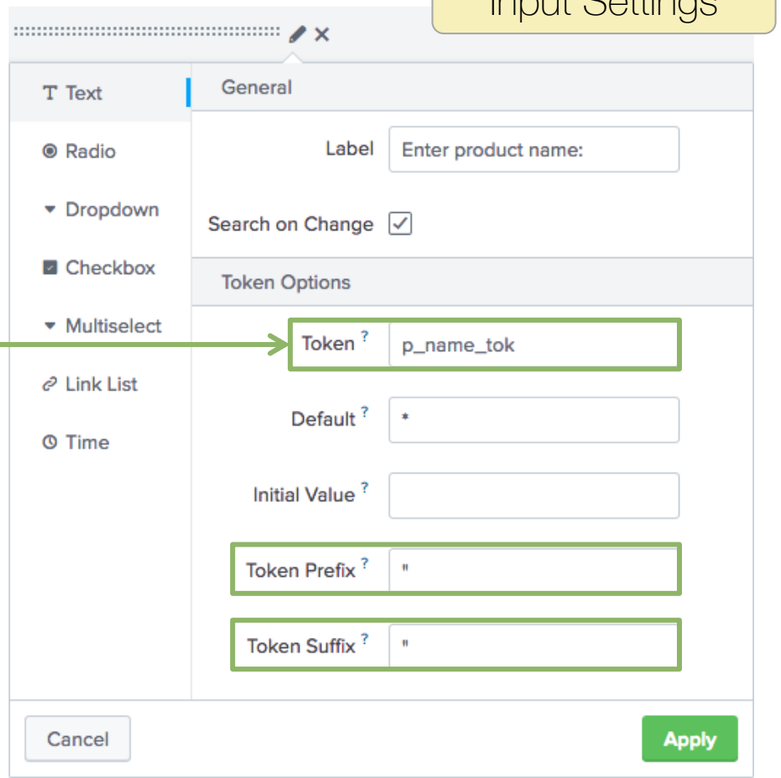
Use a name that identifies your token as a token



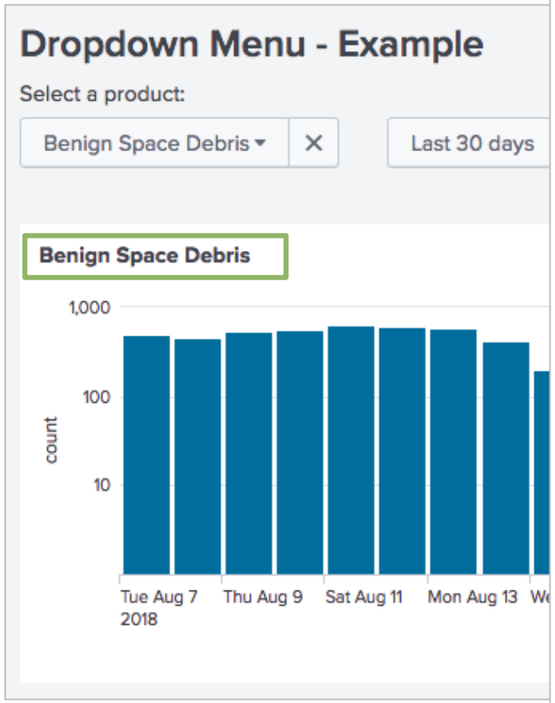

```
...
  <fieldset submitButton="false">
    <input type="text" token="p_name_tok" searchWhenChanged="true">
      <label>Enter product name:</label>
      <default>*</default>
      <prefix>"</prefix>
      <suffix>"</suffix>
    </input>
  </fieldset>
  <row>
    <panel>
      <chart>
        <title>All Vendor Sales</title>
        <search>
          <query>sourcetype="vendor_sales" product_name=$p_name_tok$
           | timechart count</query>
          <earliest>-30d@d</earliest>
          <latest>now</latest>
        </search>
...
```

Use **$...$** delimiters to access a token value

Input Settings



Dropdown Menu Example



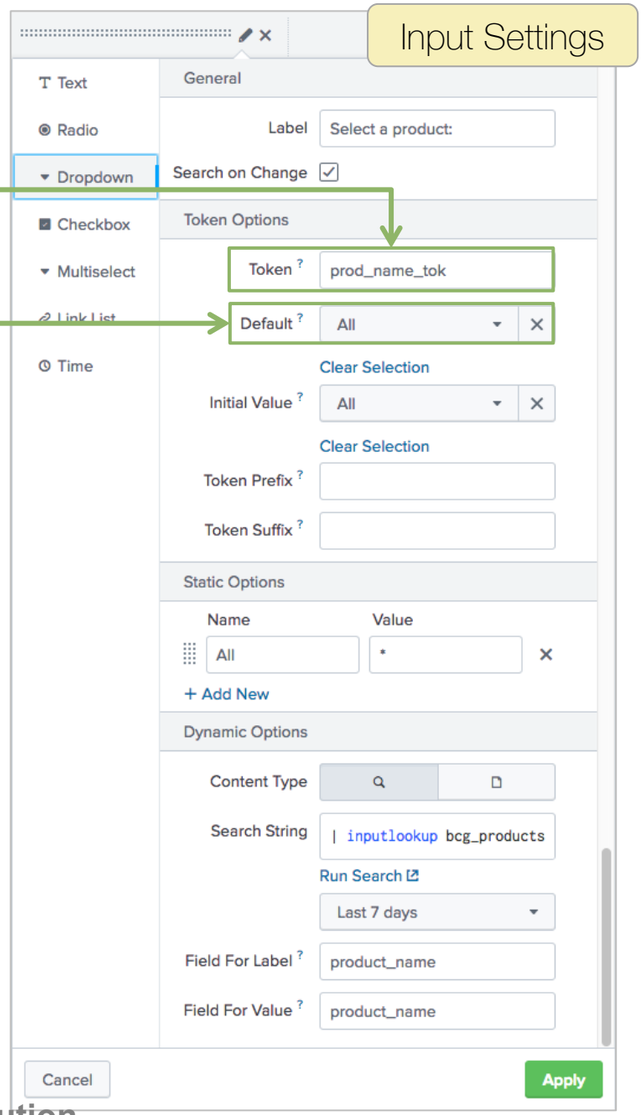

```
...
  <fieldset submitButton="false" autoRun="false">
    <input type="dropdown" token="prod_name_tok" searchWhenChanged="true">
      <label>Select a product:</label>
      <choice value="*">All</choice>
      <search>
        <query>| inputlookup bcg_products</query>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
      </search>
      <default>*</default>
      <fieldForLabel>product_name</fieldForLabel>
      <fieldForValue>product_name</fieldForValue>
      <initialValue>*</initialValue>
    </input>
    <input type="time" token="time_tok">
      <label></label>
      <default>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
      </default>
    </input>
  </fieldset>
  <row>
    <panel>
      <chart>
        <title>$prod_name_tok$</title>
        <search>
          <query>sourcetype=access_combined product_name="$prod_name_tok$"
          | timechart count</query>
          <earliest>$time_tok.earliest$</earliest>
          <latest>$time_tok.latest$</latest>
        </search>
...
```

User defined token

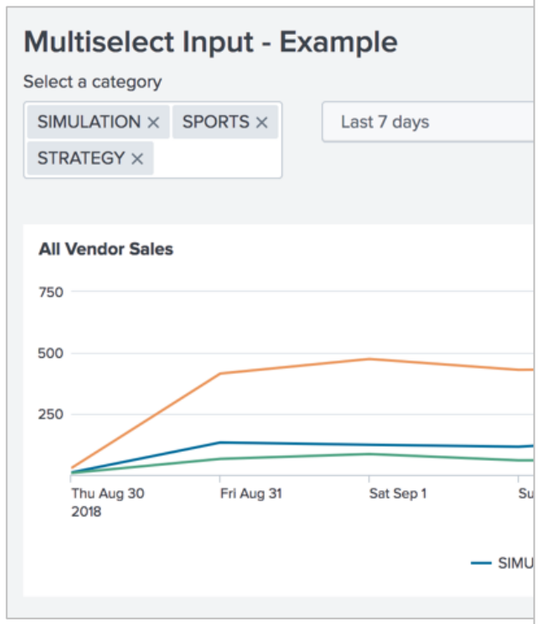
Query uses an input lookup

Define a default menu selection

Access the value using **$...$** delimiters



Multiselect Input Example

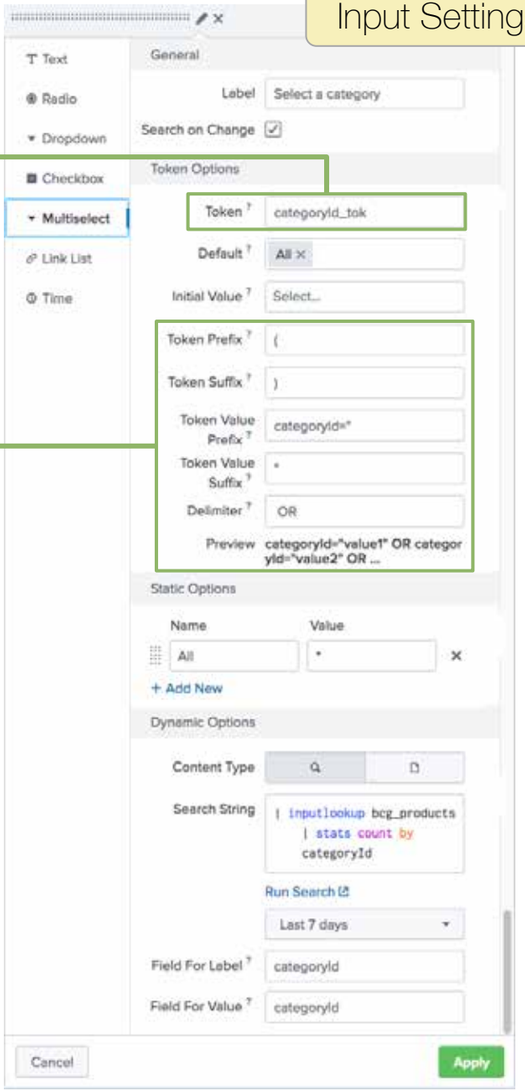



```
...
<fieldset submitButton="false" autoRun="false">
 <input type="multiselect" token="categoryId_tok" searchWhenChanged="true">
  <label>Select a category</label>
  <choice value="*">All</choice>
  <search>
   <query>| inputlookup bcg_products | stats count by categoryId</query>
   <earliest>-7d@h</earliest>
   <latest>now</latest>
  </search>
  <fieldForLabel>categoryId</fieldForLabel>
  <fieldForValue>categoryId</fieldForValue>
  <default>*</default>
  <prefix>(</prefix>
  <valuePrefix>categoryId="</valuePrefix>
  <valueSuffix>"</valueSuffix>
  <delimiter> OR </delimiter>
  <suffix>)</suffix >
 </input>
</fieldset>
 <row>
  <panel>
  <chart>
   <title>All Vendor Sales</title>
   <search>
    <query>sourcetype="access_combined" $categoryId_tok$
      | timechart count by categoryId usenull=f</query>
    <earliest>$time_tok.earliest$</earliest>
    <latest>$time_tok.latest$</latest>
   </search>
...
```

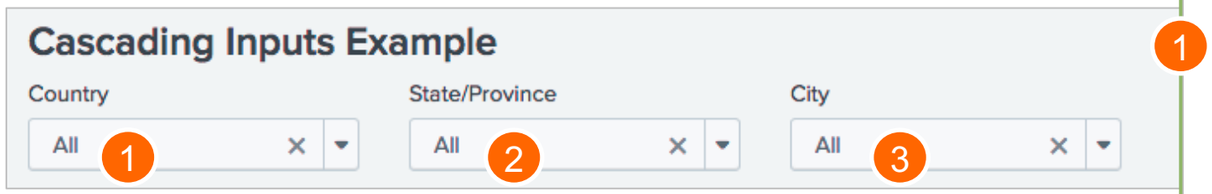
User defined token

Query uses an input lookup

Token prefix, suffix, and delimiter



Cascading Inputs



Use the selection of one input to reduce or set the values of another

```
...
<fieldset submitButton="false">
  <input type="dropdown" token="v_country_tok" searchWhenChanged="true">
    <label>Country</label>
    <choice value="*">All</choice>
    <default>*</default>
    <fieldForLabel>VendorCountry</fieldForLabel>
    <fieldForValue>VendorCountry</fieldForValue>
    <search>
      <query>| inputlookup bcg_vendors
             | search VendorStateProvince=$v_state_tok|s$ VendorCity=$v_city_tok|s$
             | dedup VendorCountry
             | fields VendorCountry
             | sort VendorCountry</query>
      <earliest>-24h@h</earliest>
      <latest>now</latest>
    </search>
  </input>
  <input type="dropdown" token="v_state_tok" searchWhenChanged="true">
    <label>State/Province</label>
    <choice value="*">All</choice>
    <default>*</default>
    <fieldForLabel>VendorStateProvince</fieldForLabel>
    <fieldForValue>VendorStateProvince</fieldForValue>
    <search>
      <query>| inputlookup bcg_vendors | search VendorCountry=$v_country_tok|s$ VendorCity=$v_city_tok|s$
             | dedup VendorStateProvince | fields VendorStateProvince | sort VendorStateProvince</query>
      <earliest>-24h@h</earliest>
      <latest>now</latest>
    </search>
  </input>
  <input type="dropdown" token="v_city_tok" searchWhenChanged="true">
    <label>City</label>
    <choice value="*">All</choice>
    <default>*</default>
    <fieldForLabel>VendorCity</fieldForLabel>
    <fieldForValue>VendorCity</fieldForValue>
    <search>
      <query>| inputlookup bcg_vendors | search VendorCountry=$v_country_tok|s$ VendorStateProvince=$v_state_tok|s$
             | dedup VendorCity | fields VendorCity | sort VendorCity</query>
      <earliest>-24h@h</earliest>
      <latest>now</latest>
    </search>
    <allowCustomValues>true</allowCustomValues>
  </input>
</fieldset>
...
```

Note

Use an event handler to reset a menu when another menu is changed. Event handlers are discussed in module 6.

Token Filters

Token filters ensure that you correctly capture a token's value

Built-in	
`$token_name\|s$`	Wrap value in quotes
`$token_name\|h$`	Escape any HTML in value
`$token_name\|u$`	Encode URL values
`$token_name\|n$`	No encode
`$$token_name$$`	Escape the $ token delimiter character
Custom	
`$token_name\|myFilter$`	Build your own token filters in JavaScript

Token Filters – Example 1

Use the |s filter to add quotation marks around the returned value

```
...
<search>
<query>index=bcg_index sourcetype=$srctyp_tok|s$ | timechart count by sourcetype</query>
<earliest>-30d@d</earliest>
<latest>now</latest>
<sampleRatio>1</sampleRatio>
</search>
...
```

```
index=bcg_index sourcetype="access_combined" | timechart count by sourcetype
```

Token Filters – Example 2

Use the |n filter to not encode

```
...
  <search>
    <query>index=news sourcetype=$media_tok$
    | table sourcetype title link
    </query>
    <earliest>$time_tok.earliest$</earliest>
    <latest>$time_tok.latest$</latest>
    <sampleRatio>1</sampleRatio>
  </search>
  <option name="count">20</option>
  <option name="wrap">true</option>
  <drilldown>
   <link target="_blank"> $row.link|n$ </link>
  </drilldown>
...
```

```
https://www.nytimes.com/2017/10/16/technology/personalt
ech/macos-high-sierra-upgrade.html?partner=rss&emc=rss
```

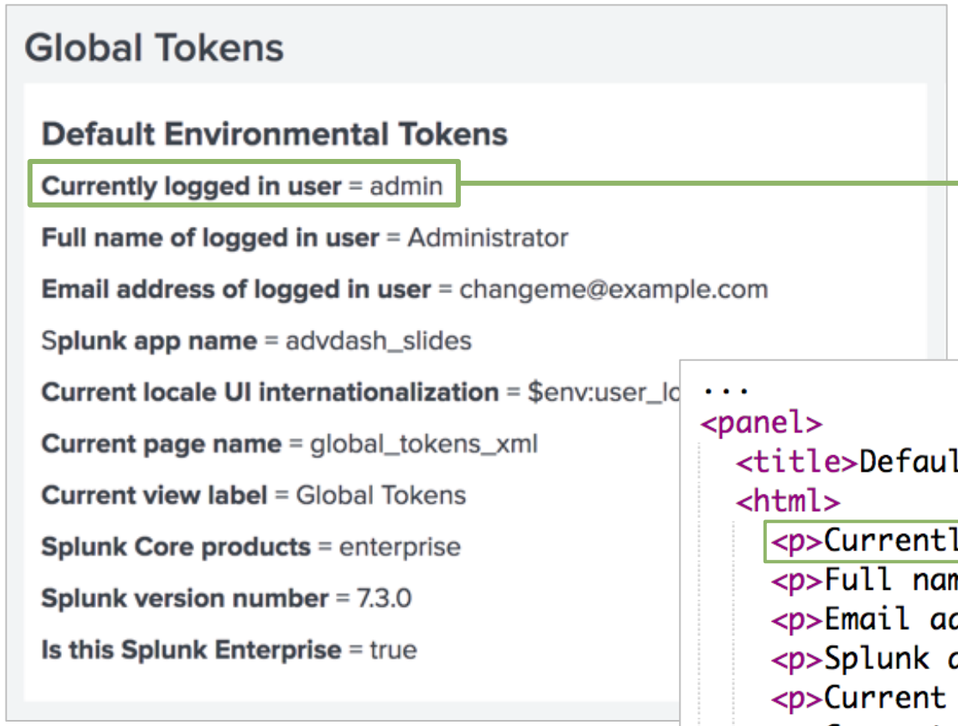
```
...
  <search>
    <query>index=news sourcetype=$media_tok$
    | table sourcetype title link
    </query>
    <earliest>$time_tok.earliest$</earliest>
    <latest>$time_tok.latest$</latest>
    <sampleRatio>1</sampleRatio>
  </search>
  <option name="count">20</option>
  <option name="wrap">true</option>
  <drilldown>
   <link target="_blank"> $row.link$ </link>
  </drilldown>
...
```

```
http://52.12.61.149/en-
US/app/search/https%3A%2F%2Fwww.nytimes.com%2F2017%2F10
%2F16%2Ftechnology%2Fpersonaltech%2Fmacos-high-sierra-
upgrade.html%3Fpartner%3Drss%26emc%3Drss
```

Global Environment Tokens

UX

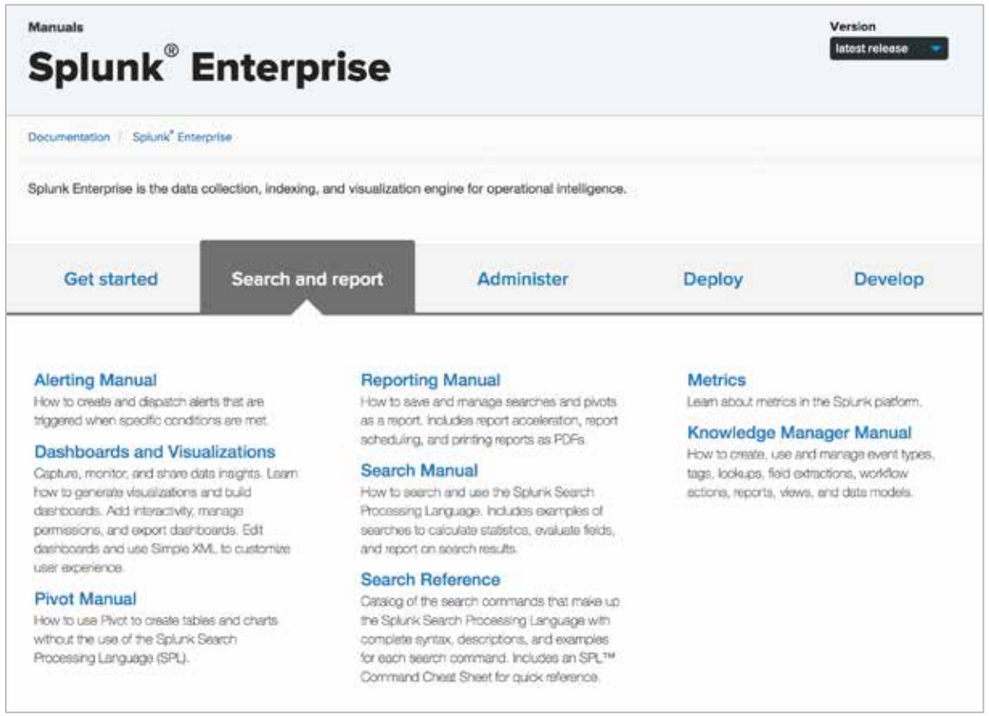
- $env:user$
- $env:user_realname$
- $env:user_email$
- $env:app$
- $env:locale$
- $env:page$
- $env:product$
- $env:version$
- $env:view_label$
- $env:is_enterprise$




```
...
<panel>
  <title>Default Environmental Tokens</title>
  <html>
    <p>Currently logged in user = $env:user$</p>
    <p>Full name of logged in user = $env:user_realname$</p>
    <p>Email address of logged in user = $env:user_email$</p>
    <p>Splunk app name = $env:app$</p>
    <p>Current locale UI internationalization = $env:locale$</p>
    <p>Current page name = $env:page$</p>
    <p>Current view label = $env:view_label$</p>
    <p>Splunk Core products = $env:product$</p>
    <p>Splunk version number = $env:version$</p>
    <p>Is this Splunk Enterprise = $env:is_enterprise$</p>
  </html>
</panel>
...
```

Useful References

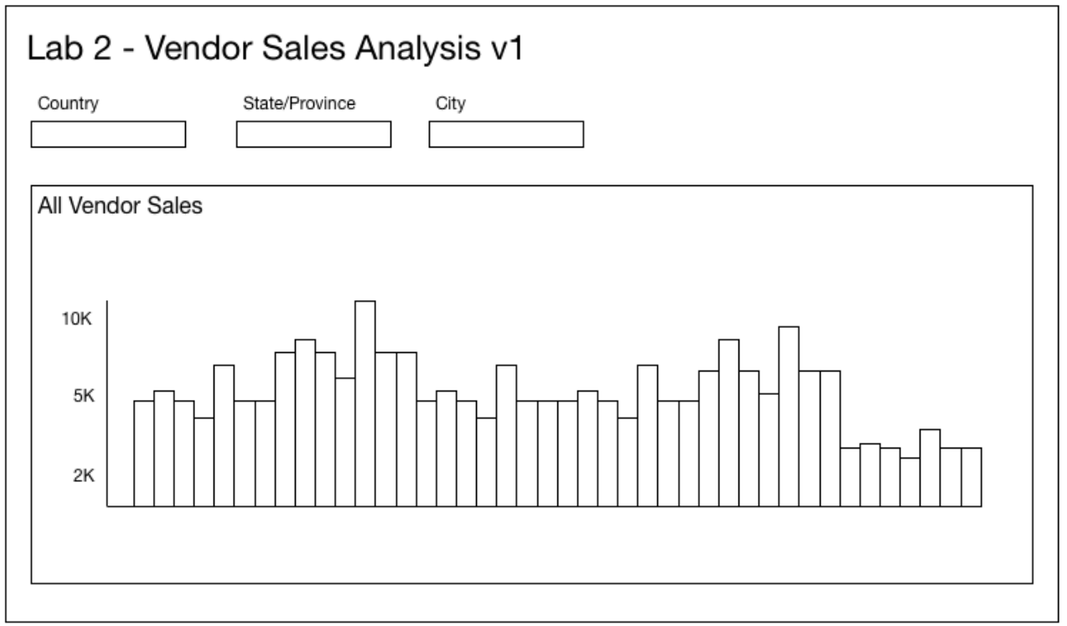
- Token Usage in dashboards
  docs.splunk.com/Documentation/Splunk/latest/Viz/tokens
- Token Filters
  docs.splunk.com/Documentation/Splunk/latest/Viz/tokens#Token_filters
- Create and Edit Forms
  docs.splunk.com/Documentation/Splunk/latest/Viz/FormEditor



Lab 2 – Add Interactivity

Time: 40 minutes

Scenario: The sales team is impressed with the new server errors dashboard. They would like to add a form to their app. The form should display a column chart of vendor data based on user input for country, state or province, and city.



Tasks:

- Create a form
- Add a token filter
- Add cascading inputs

Challenge Task (optional):

- Add an event handler to reset menus

Module 3: Improving Performance

Module Objectives

- Identify ways to improve dashboard performance
- Describe techniques to improve search efficiency
- Name two ways to accelerate a search
- Use the tstats command with a global search
- Use a post-process search

Improving Performance

- Refine searches
- Use scheduled reports
- Accelerate reports
- Accelerate data models
- Use the tstats command
- Use a global search
- Use tokens

Refining Searches

- Limit your search to a specific time window and quantity of data retrieved
- Use the most efficient command for the use case
  - transaction vs. stats
  - tstats
- Make the base search part of a global search, as specific as possible
- Avoid using NOT expressions

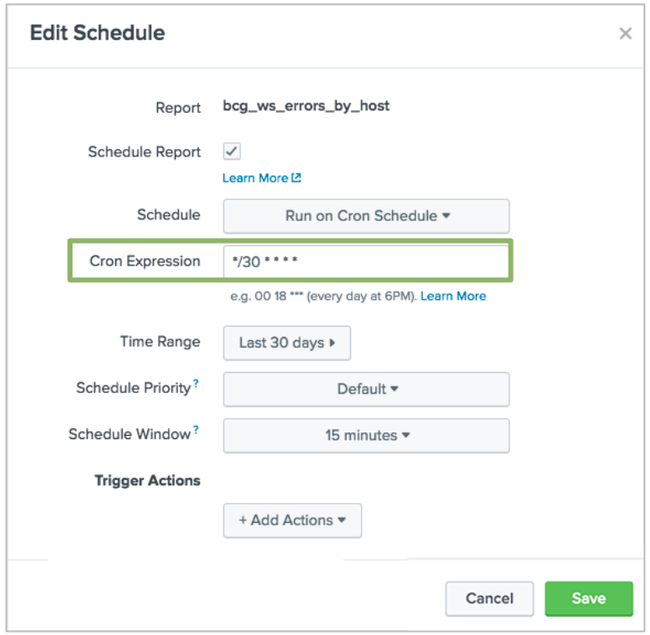
```
... | transaction trade_id | chart count by duration
```

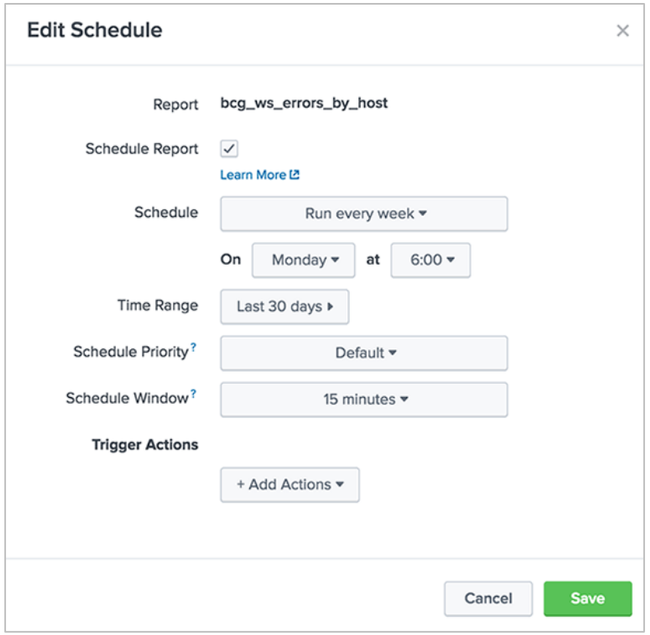
```
... | stats range(_time) as duration by trade_id
    | chart count by duration
```

```
...
 <search>
   <query>index=_internal $user$ | timechart $metric$($what$) by $group$</query>
   <earliest>-30d@d</earliest>
   <latest>now</latest>
...
 <search>
   <query>| tstats $metric$($what$) from datamodel=foo where $user$ by all.$group$
   prestats=t summariesonly=t | timechart $metric$($what$) by $group$</query>
   <earliest>-30d@d</earliest>
   <latest>now</latest>
 ...
```

Using Scheduled Reports

- Avoid inline searches
- Schedule to run every 5 or 10 minutes or less frequently
- Prevent a flood of search jobs when dashboards are loaded

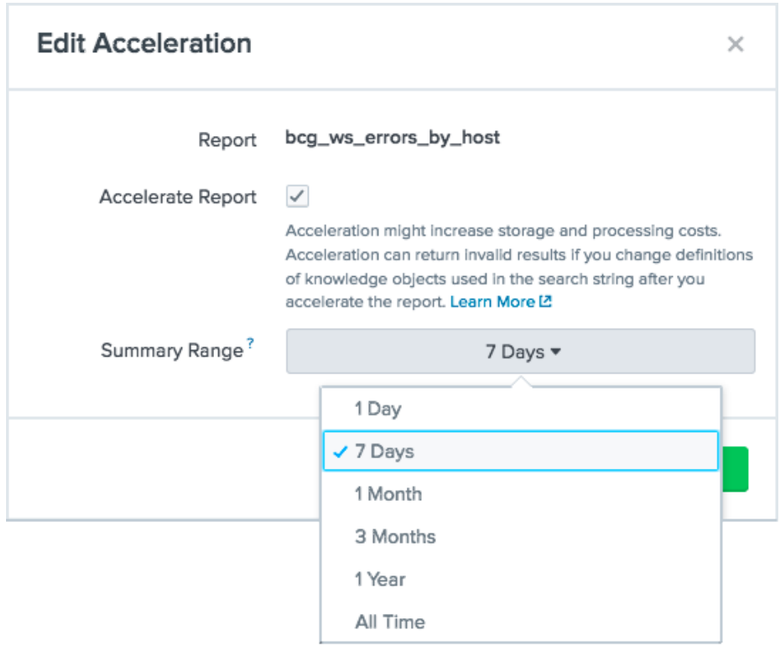




Cron Parameter	Schedule
*/5 * * * *	Every 5 minutes
*/30 * * * *	Every 30 minutes
0 */12 * * *	Every 12 hours, on the hour
*/20 * * * 1-5	Every 20 minutes, Monday through Friday
0 9 1-7 * 1	First Monday of each month, at 9am.

Accelerating Reports

- Automatically creates summaries to speed completion times
- Periodically ages out data
- Data is stored on the indexers
- Search must meet three criteria:
  - Uses a transforming command
  - Commands before the first transforming command, must be streamable
  - Cannot use event sampling

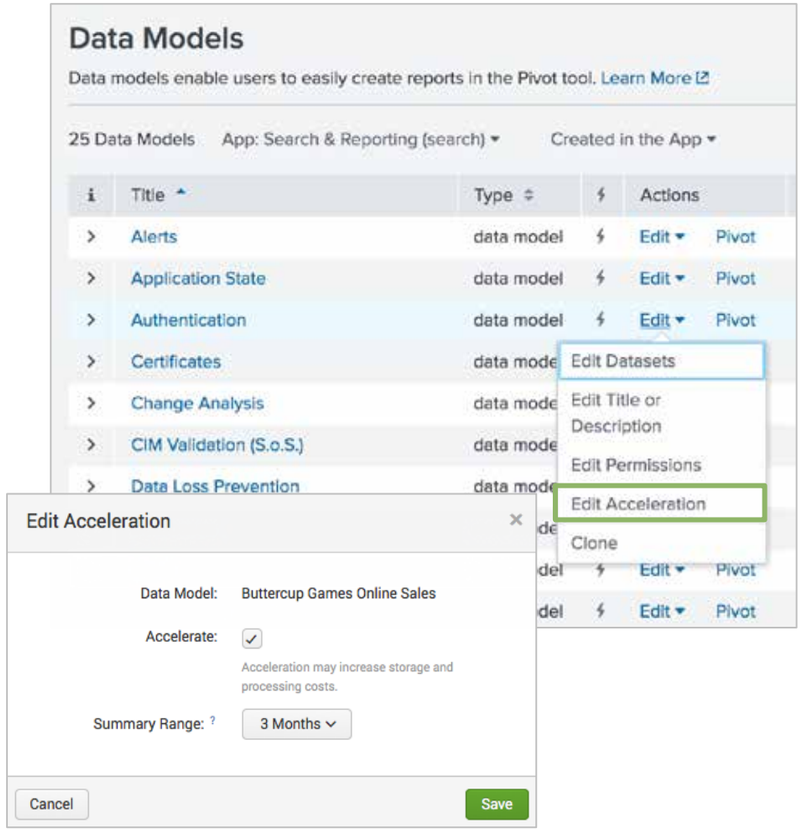


Note

Report acceleration only works for reports that have Search Mode set to Smart or Fast.

Accelerating Data Models

- Accelerates all fields defined in the data model
- Creates time-series index (TSIDX) files
- Updates every five minutes
- Only administrators can enable acceleration
- Anyone can search using an accelerated data model
- Most efficient when root event dataset includes the index in the initial constraint search



`tstats` Command

- Generating command
- Use to search data models or data model objects
- Perform statistical queries on indexed fields in tsidx files
  - Also against indexed fields like source, host, sourcetype, and index
- Wildcard characters are not supported in field values in aggregate functions or BY clauses

```
| tstats [prestats=boolean] [summariesonly=boolean] <stats-function> [FROM datamodel=<datamodel-name>] [BY field-list]
```

- Use to pipe results to chart, stats or timechart
- Use to generate results from TSIDX data
- Perform a basic count of a field or a function on a field
- Specify the filename (object ID) of an accelerated data model
- Specify one or more fields to group results

`tstats` Command – Arguments

- `prestats=<boolean>`
  - **`true`** allows you to pipe the data to **`chart`**, **`stats`**, or **`timechart`**
    - Prevents renaming the result using the AS keyword
  - **`false`** is the default
  - Enables **`append=t`** where the results append to existing results instead of generating them

```
| tstats [prestats=boolean] [summariesonly=boolean] <stats-function> [FROM datamodel=<datamodel-name>] [BY field-list]
```


`tstats` Command – Arguments (cont.)

- `summariesonly=boolean`
  - Applies only to an accelerated data model
  - **`true`** generates results from the accelerated data model's TSIDX data
  - **`false`** (default) generates results from both summarized (accelerated data model TSIDIX data) and non-summarized data
    - May cause a larger result count if: some of the data has not yet been added to the summary OR has been aged out of it

```
| tstats [prestats=boolean] [summariesonly=boolean] <stats-function> [FROM datamodel=<datamodel-name>] [BY field-list]
```

`tstats` Command – Functions

- `stats-function`
  - Perform a basic count or a function on a field
  - Perform any number of aggregates
  - Can rename the result using AS

Type	Supported functions and syntax			
Aggregate functions	avg() count() distinct_count() estdc()	exactperc<int>() max() median() min() mode()	perc<int>() range() stdev() stdevp()	sum() sumsq() upperperc<int>() var() varp()
Event order functions	earliest()	first()	last()	latest()
Multivalue stats and chart functions	values(x)			

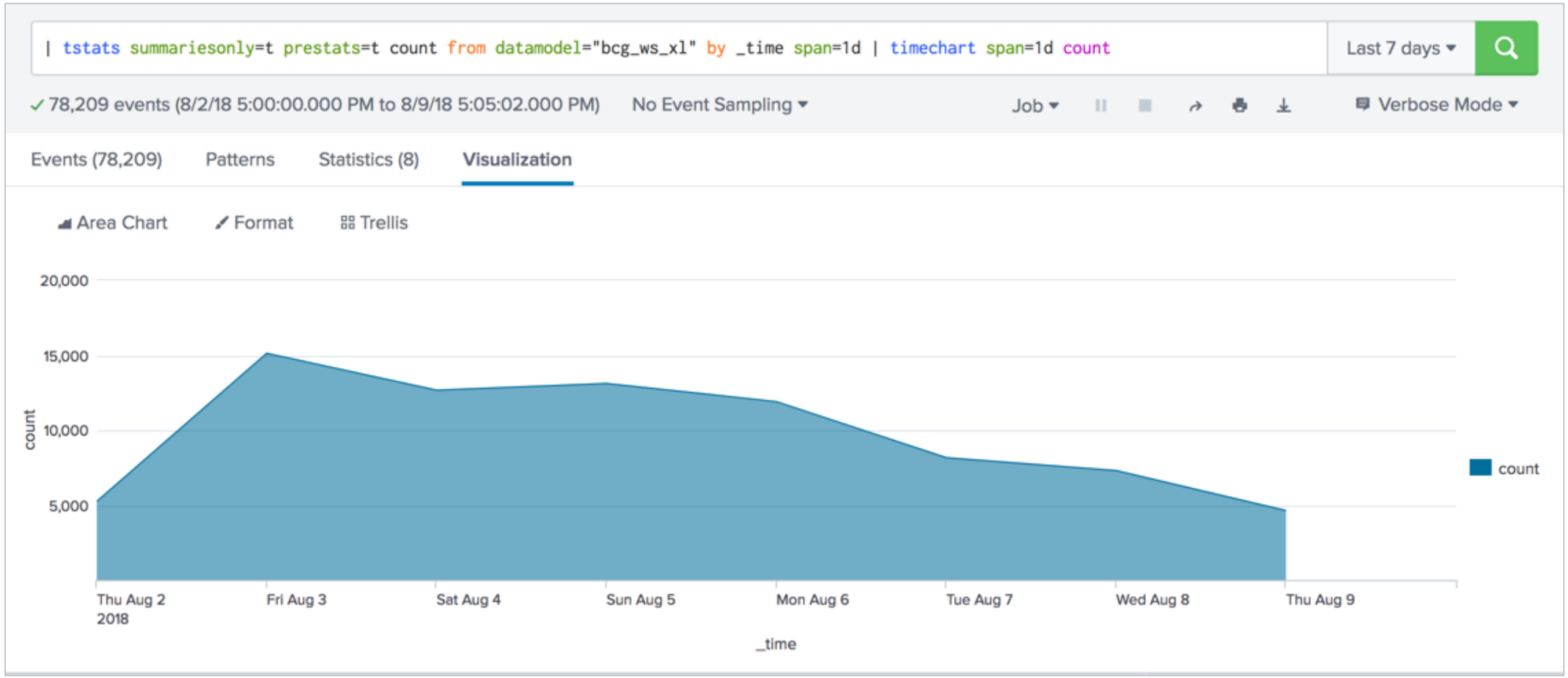
```
| tstats [prestats=boolean] [summariesonly=boolean] <stats-function> [FROM datamodel=<datamodel-name>] [BY field-list]
```

`tstats` Command – Clause Arguments

- `FROM datamodel=<datamodel-name>`
  - Accesses an accelerated data model's summaries
- `WHERE <search-query>`
  - Specify a search
  - Can specify a set of values with the IN operator
- `BY <field-list>`
  - You must specify a field-list
  - If you used to group by _time, use `span` to group the time buckets
  - Cannot use wildcards

```
| tstats [prestats=boolean] [summariesonly=boolean] <stats-function> [FROM datamodel=<datamodel-name>] [BY field-list]
```

`tstats` Command – Example



Use a Global Search

- Global search: a **single** base search with post-process searches that populate all panels on a dashboard or form
- Base search: `<search id="myBaseSearch">`
  - Provides the input for post-process searches
  - Typically a dedicated search not in a panel
  - Can be any inline search or report on a dashboard

    `<search id=myBaseSearch ref=myReport>`
  - Use a transforming command to avoid the 10,000 results limit that can be passed to a post-process search
- Post-process search: `<search base="myBaseSearch">`
  - Perform additional processing on base search results
  - Can populate form inputs

> **Warning**
>
> Passing a large number of search results from a base search can cause a server time out.

Use a Global Search (cont.)

- The base search gathers statistics for the downline processing

```
<dashboard>
 <label>Global Search Example</label>
 <search id="BaseSearch">
  <query><![CDATA[index=main sourcetype=vendor_sales OR (sourcetype=access_combined action=purchase OR action=remove status=200)
    | fillnull value="NONE" | search product_name!=NONE | stats count by _time, action, status, sourcetype, product_name, host, VendorID]]>
  </query>
  <earliest>-30d@h</earliest>
  <latest>now</latest>
 </search>
 ...
```

Base search ID

Base search

- The post process performs further processing of results

```
...
<title>Online vs. Retail</title>
<chart>
  <search base="BaseSearch">
    <query>search action!=NONE | chart sum(count) as count by product_name, action</query>
  </search>
  ...
```

Base search reference

Post-process

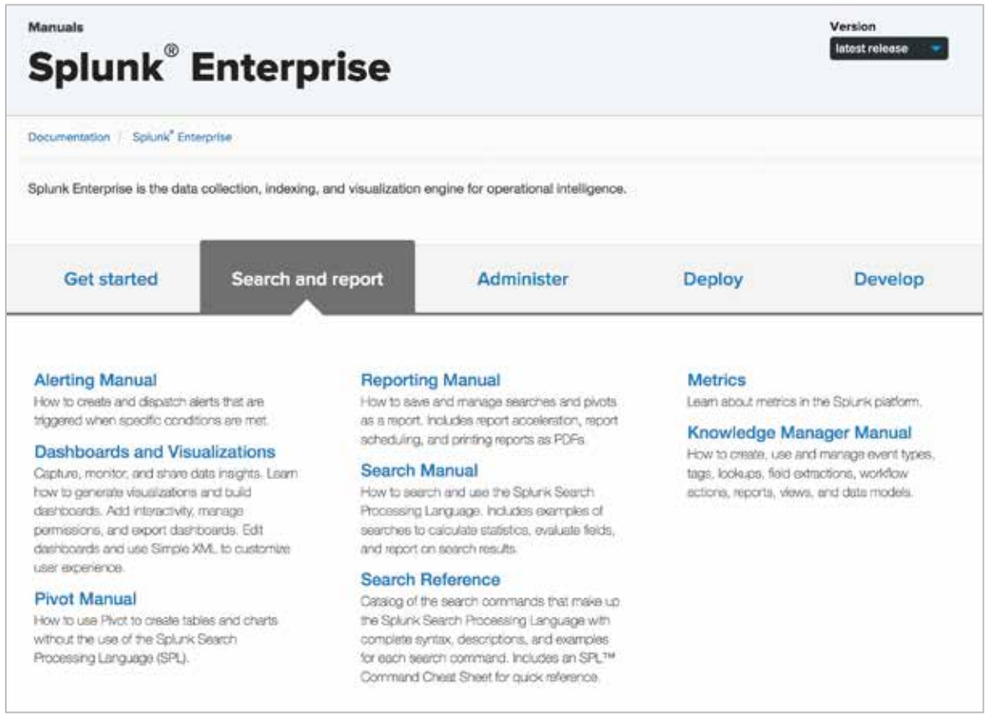
Global Search – Null Values

- Make sure all fields identified in a base search have a value
- Remove or include null values as required in the post-process

```
...
<search id="baseSearch">
 <query>index=bcg_index sourcetype=access_combined | fillnull value="NONE"
     | stats count by _time, sourcetype, action, status, product_name, host</query>
 <earliest>-30d@d</earliest>
 <latest>now</latest>
</search>
<row>
 <panel>
  <chart>
   <title>Purchases &amp;amp; Lost Sales</title>
   <search base="baseSearch">
    <query>search product_name!=NONE action!=NONE AND (action=purchase OR action=remove)
    | chart sum(count) as count by product_name, action
    | rename product_name as "Product Name", remove as "Lost Sales", purchase as Purchases</query>
   </search>
...
```

Useful References

- Search Reference – tstats
  docs.splunk.com/Documentation/Splunk/latest/SearchReference/Tstats
- Searches Power Dashboards and Forms
  docs.splunk.com/Documentation/Splunk/latest/Viz/Savedsearches
- Dashboards & Visualizations Manual
  docs.splunk.com/Documentation/Splunk/latest/Viz/Aboutthismanual



Lab 3 – Improve Performance

Time: 40 minutes

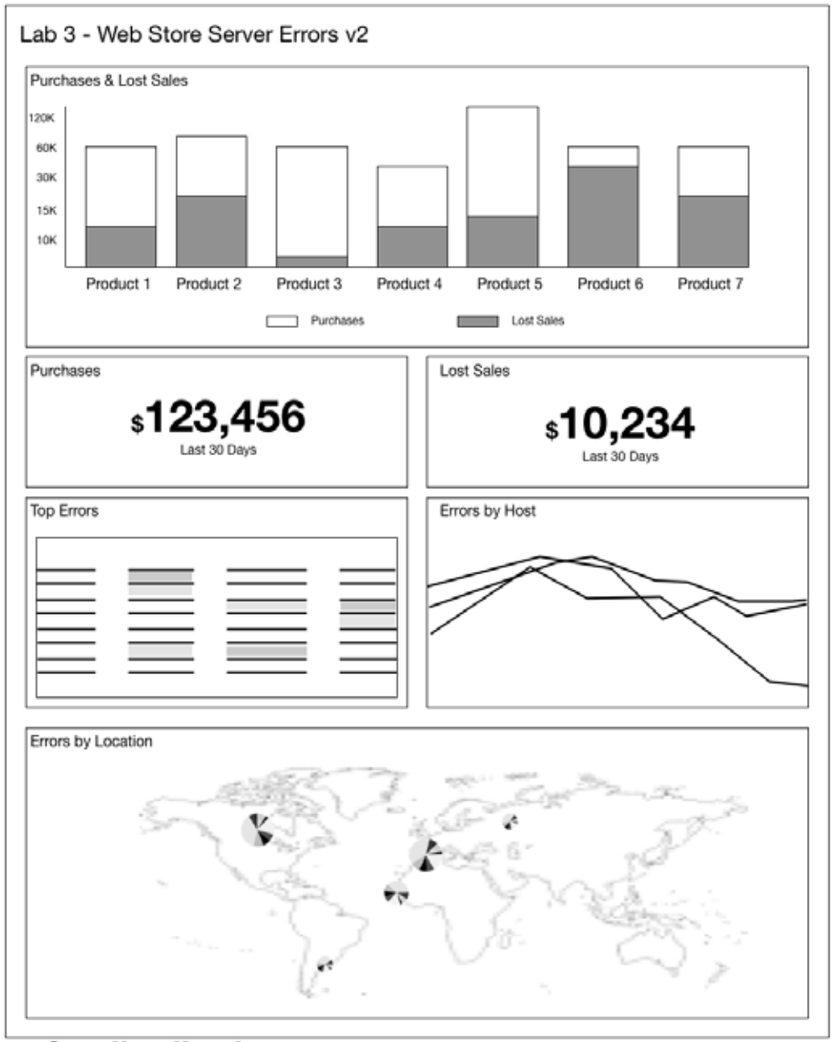
Scenario: Stakeholders have approved the prototype dashboard with some changes. They want it to load faster, have lower search overhead, and include two new panels that show the dollar amount of purchases and lost sales.

Tasks:

- Accelerate and schedule reports
- Create a dashboard
- Add a base search
- Add panels driven by post-process searches
- Add prebuilt panels and convert them to inline
- Accelerate the global search

Challenge Task (optional):

- Accelerate your data model
- Revise the base search to reference your data model



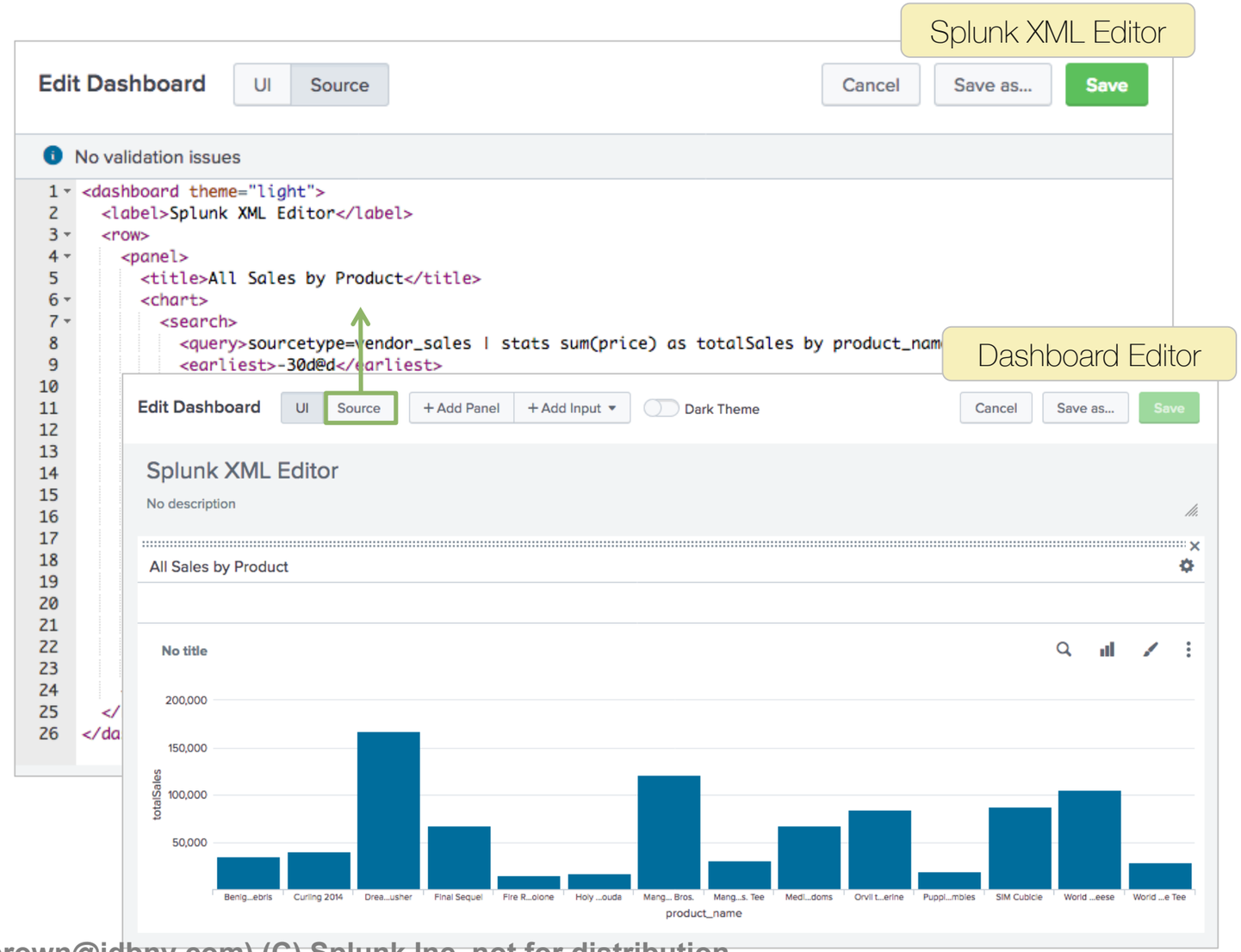
Module 4: Customizing Dashboards

Module Objectives

- Modify chart and panel colors
- Disable search access features
- Define Simple XML attributes
- Set panel refresh and delay times

Customizing Views

- Dashboard Editor
  - Create and update dashboards and panels
- Customizing Simple XML
  - Chart and panel colors
  - Panel links
  - Panel refresh
  - Grouping panels



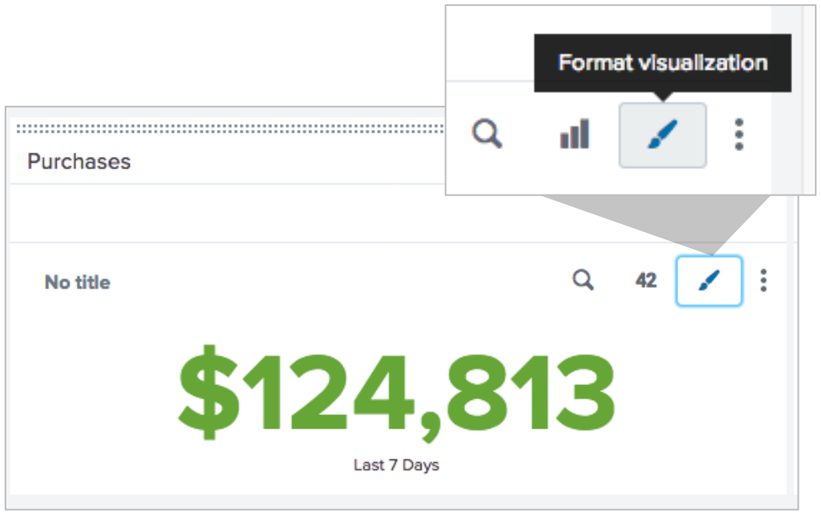
Element Attributes

- Configure appearance and behavior of dashboards and forms
- Available for most Simple XML elements
  - `<dashboard>`, `<form>`, `<fieldset>`, `<row>`, `<panel>`, etc.
  - For example: `<dashboard script="myScript">`

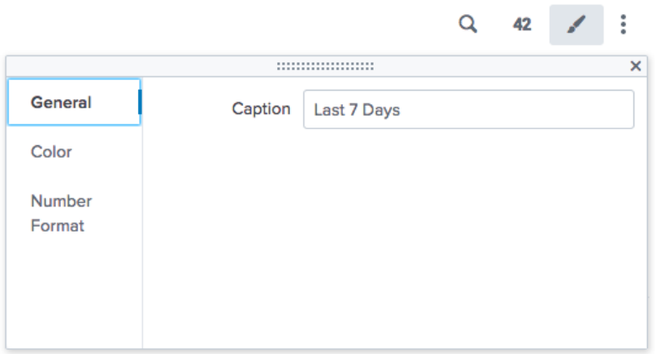
Attribute	Type	Description	Example
`onunloadCancelJobs`	boolean	Cancels search jobs when a user navigates away from the dashboard.	`<dashboard onunloadCancelJobs="true">`
`script`	string	Comma-separated list of custom .js files to load. The files must be in a folder or subfolder of the appserver/static directory.	`<form script="myScript.js">`
`rejects`	Comma-separated list	Prevent an element from rendering if one or more tokens in this list are defined.	`<panel id="panel1" rejects="$panel2$">`
`depends`	Comma-separated list	All tokens in the list must be defined for the element to render.	`<chart id="column" depends="$showCol$">`
`id`	text	Element identifier. Only alpha-numeric and underscore characters are valid.	`<row id="row1">`

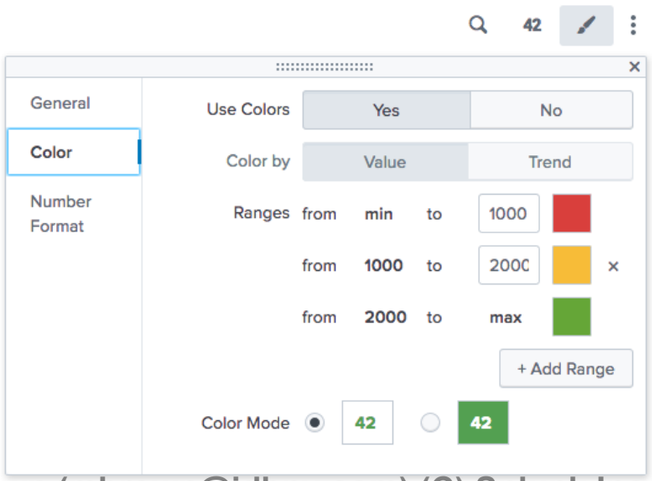
Chart Customization – Dashboard Editor

- Visualization Formatter
  - Change chart axis labels
  - Define color ranges
  - Options for each visualization



Single Value Visualization Options





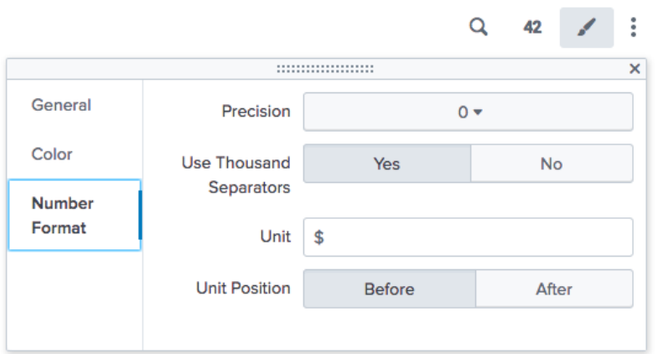


Chart Customization – XML Editor

- Edit the panel XML directly to customize the appearance and behavior of your charts
  - Axis label text styles
  - Chart colors
  - Chart background and foreground colors
  - Chart height
  - Spacing between bars, columns, or clustered series

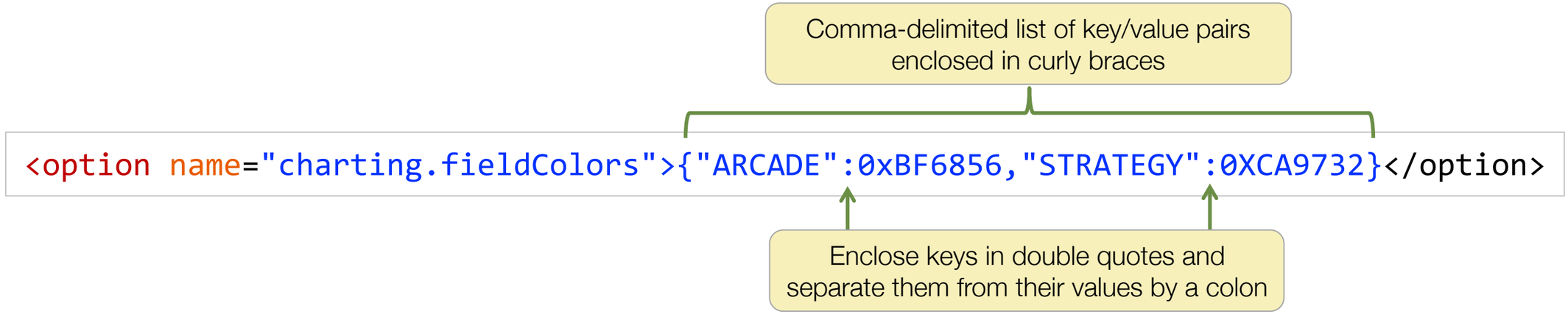
```
<option name="option name">value</option>
```

```
...
<option name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone</option>
<option name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>

<option name="charting.axisTitleX.visibility">visible</option>
<option name="charting.axisTitleY.visibility">visible</option>
<option name="charting.axisTitleY2.visibility">visible</option>

<option name="charting.axisX.abbreviation">none</option>
<option name="charting.axisX.scale">linear</option>
<option name="charting.axisY.abbreviation">none</option>
<option name="charting.axisY.scale">linear</option>
<option name="charting.axisY2.abbreviation">none</option>
<option name="charting.axisY2.enabled">0</option>
<option name="charting.axisY2.scale">inherit</option>

<option name="charting.chart">column</option>
<option name="charting.chart.bubbleMaximumSize">50</option>
<option name="charting.chart.bubbleMinimumSize">10</option>
<option name="charting.chart.bubbleSizeBy">area</option>

<option name="charting.chart.nullValueMode">gaps</option>
<option name="charting.chart.showDataLabels">none</option>
<option name="charting.chart.sliceCollapsingThreshold">0.01</option>
<option name="charting.chart.stackMode">stacked</option>
<option name="charting.chart.style">shiny</option>
<option name="charting.drilldown">none</option>

<option name="charting.layout.splitSeries">0</option>
<option name="charting.layout.splitSeries.allowIndependentYRanges">0</option>
<option name="charting.legend.labelStyle.overflowMode">ellipsisMiddle</option>
<option name="charting.legend.mode">standard</option>
<option name="charting.legend.placement">bottom</option>
<option name="charting.lineWidth">2</option>
</chart>
...
```

Chart Options Examples

Chart Customization – XML Editor Example

- `charting.fieldColors`
  - The map of hexadecimal color values to use for each field



- Escape existing double quotes or backslashes or colons with a preceding backslash
- Escape special characters `[]{}(),:"` in a key or string value with double quotes

Chart Customization – XML Editor Example (Cont.)

1. Add `charting.fieldColors` option
2. Add hexadecimal for each data series

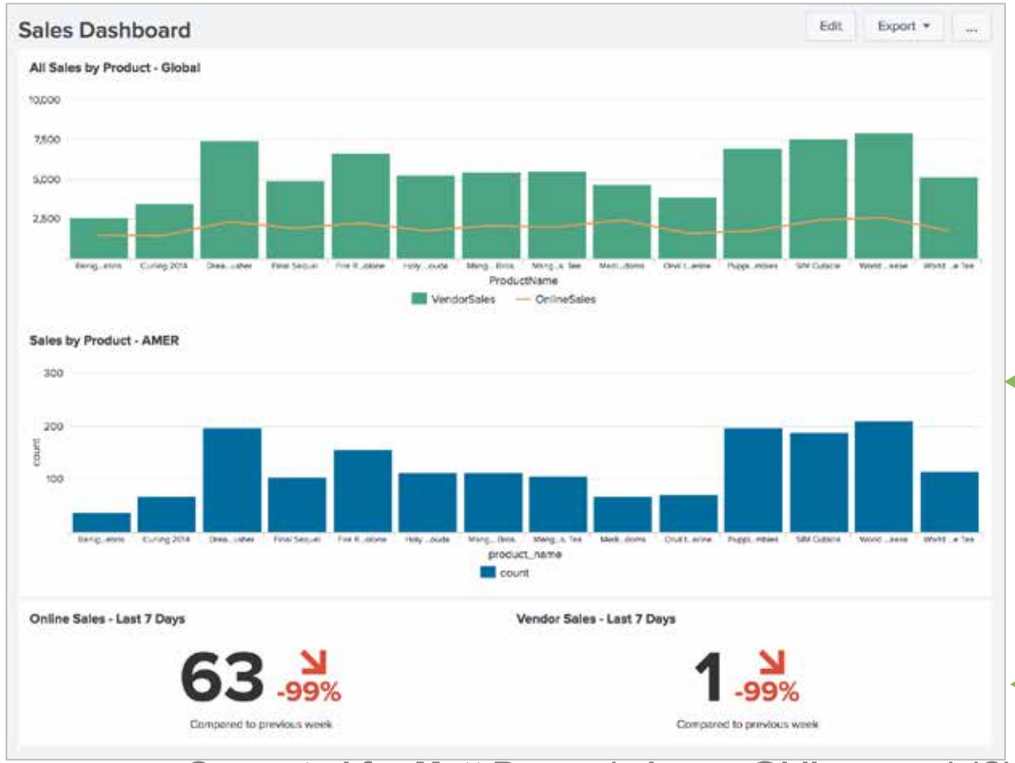
```
<option name="charting.fieldColors">
{"fieldname": hexcolor, "fieldname": hexcolor, ...}
</option>
```


```
<dashboard>
 <label>Chart Coloring Example</label>
 <row>
   <panel>
     <title>Default Palette</title>
     <chart>
       <search ref="bcg_arcade_vs_strategy"></search>
       ...
       <option name="charting.legend.placement">bottom</option>
     </chart>
   </panel>
 </row>
 <row>
   <panel>
     <title>Custom Palette</title>
     <chart>
       <search ref="bcg_arcade_vs_strategy"></search>
       ...
       <option name="charting.legend.placement">bottom</option>
       <option name="charting.fieldColors">
         {"ARCADE":0xBF6856,"STRATEGY":0xCA9732}
       </option>
     </chart>
   </panel>
 </row>
</dashboard>
```

1 2

Grouping Visualizations

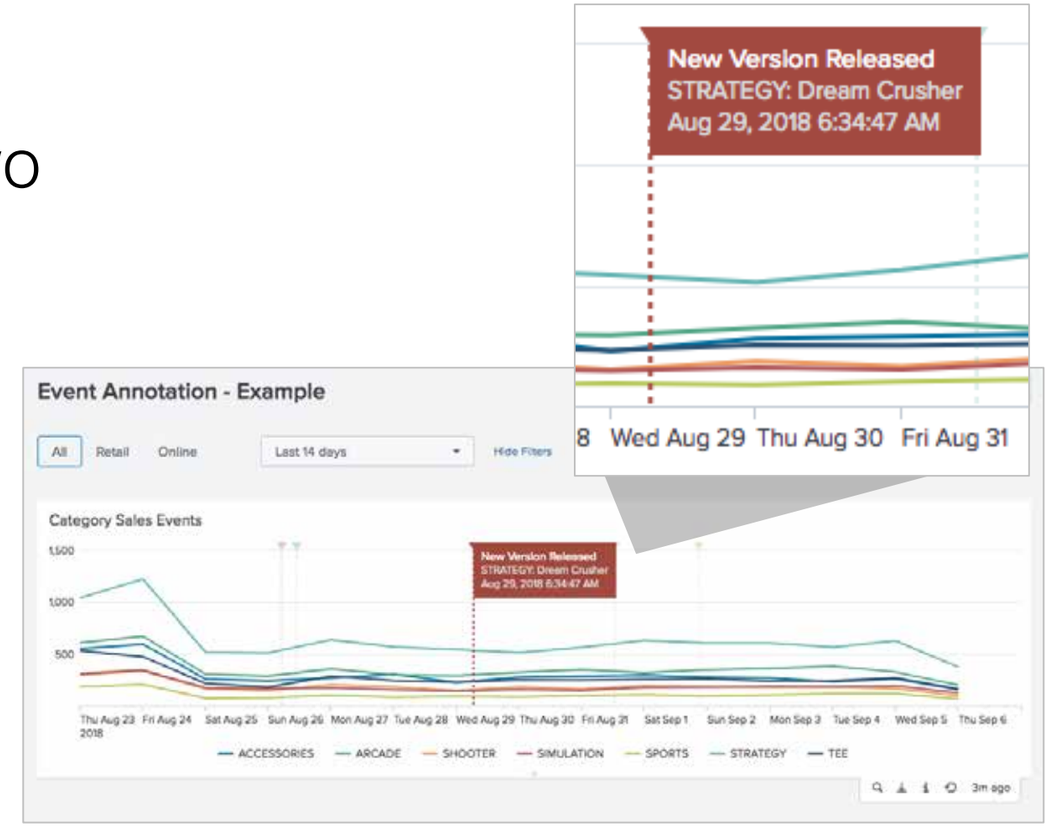
- `<row>`
  - Single value visualizations group horizontally
  - Default: visualizations group vertically




```
<dashboard>
 <label>Sales Dashboard</label>
 <row>
   <panel>
     <chart>
       <title>All Sales by Product - Global</title>
       <search ref="All Sales by Product - Global"></search>
       ...
     </chart>
     <chart>
       <title>Sales by Product - AMER</title>
       <search ref="Sales by Product - AMER"></search>
       ...
     </chart>
   </panel>
 </row>
 <row>
   <panel>
     <single>
       <title>Online Sales - Last 7 Days</title>
       <search ref="Online Sales - Last 7 Days"></search>
       ...
     </single>
     <single>
       <title>Vendor Sales - Last 7 Days</title>
       <search ref="Vendor Sales - Last 7 Days"></search>
       ...
     </single>
   </panel>
 </row>
</dashboard>
```

Event Annotation

- `<search type="annotation">`
- Annotate a time series visualization with two field values from another search or lookup
  - One value as a “label”
  - Second value as a “category”
- Displays as a callout
  - Annotation search _time value matches a _time value in the timeseries visualization
- Automatically filters for events matching the chart's time range



Note

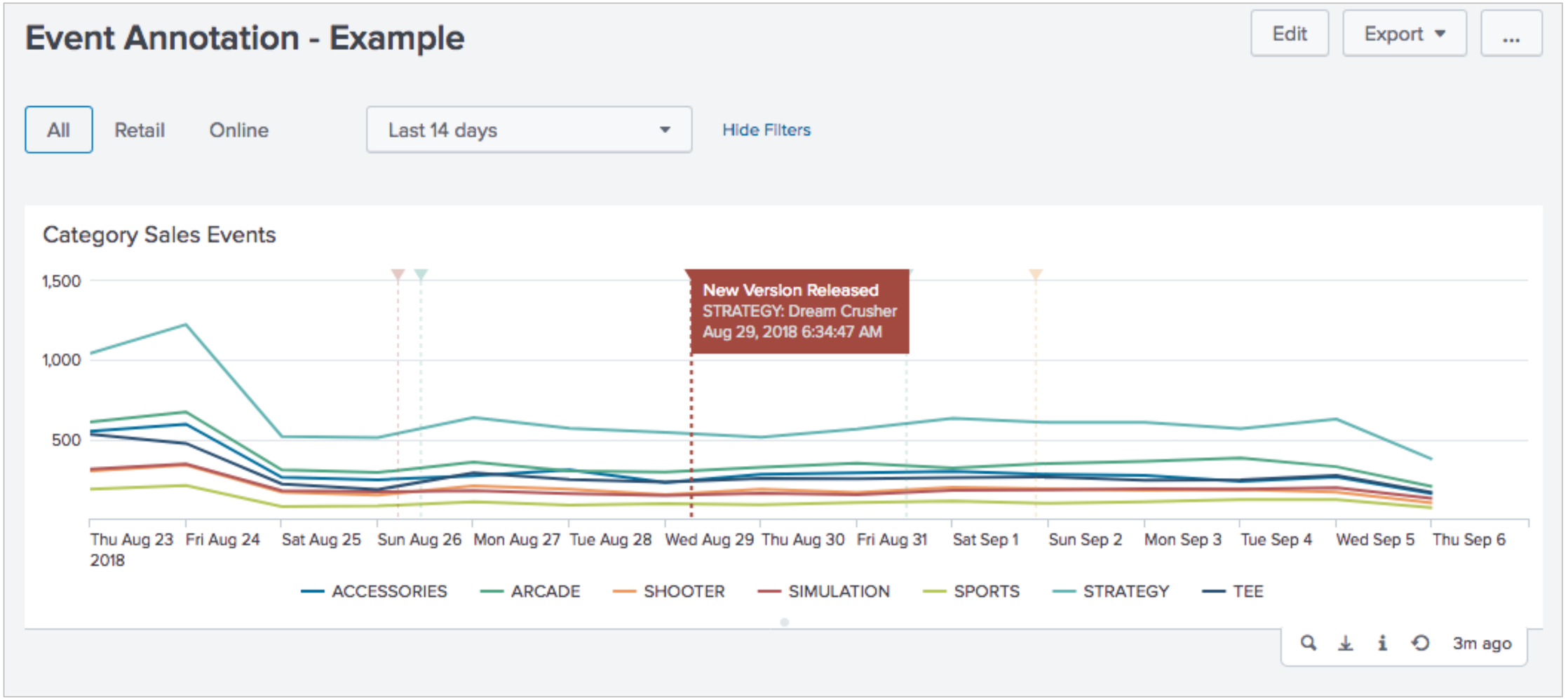
PDF export is not available for event annotations.

Event Annotation – Example

1. `_time`: required
2. `Search type`: required
3. `annotation_label`: Field in your data containing the text to display
4. `annotation_category`: Field in your data to group annotation events by
5. `annotation_color`: Assign a color to an annotation event using either HEX or RGB notation

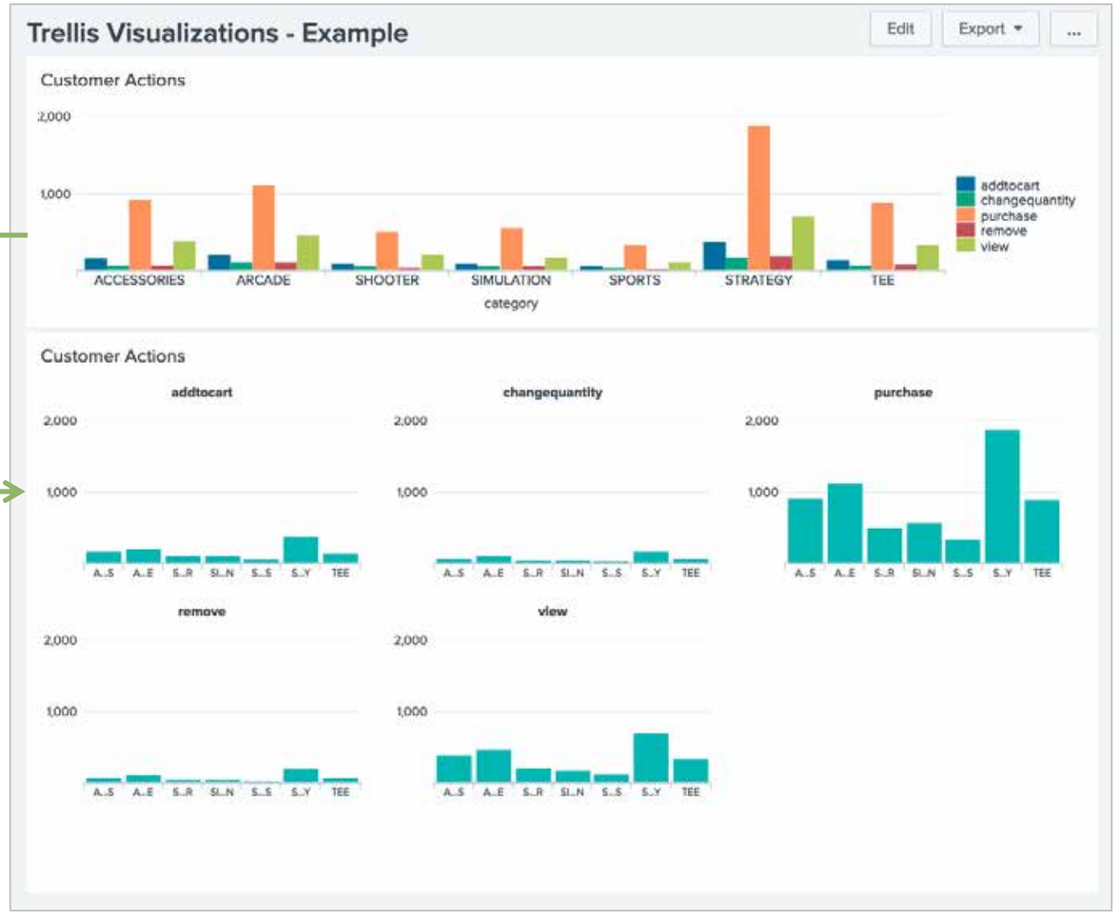
```
...
 <chart>
  <search>
1   <query>$sales_type_tok$ categoryId!=NULL
    | timechart count by categoryId</query>
   <earliest>$time_tok.earliest$</earliest>
   <latest>$time_tok.latest$</latest>
   <sampleRatio>1</sampleRatio>
  </search>
2 <search type="annotation">
3   <query>index="sales" sourcetype="bcg_sale_dates"
    | eval annotation_label = message
4   | eval annotation_category = Sale_Category
   </query>
   <earliest>$time_tok.earliest$</earliest>
   <latest>$time_tok.latest$</latest>
  </search>
5 <option name="charting.annotation.categoryColors">
  {"ARCADE":"0xff3300","SIMULATION":"0xff9900",
   "STRATEGY":"0xff3300", "TEE":"0xff9900",
   "SPORTS":"0xff3300"}
  </option>
...
```

Event Annotation – Example (cont.)



Trellis Visualizations

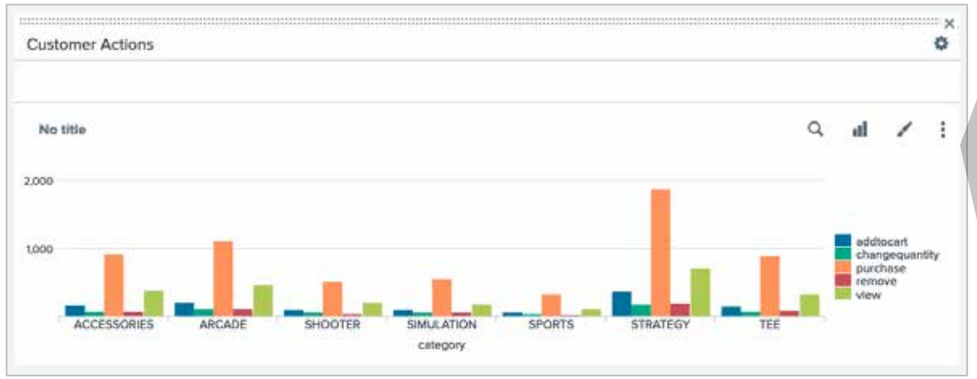
- Multiple instances of a visualization, where each instance shows values for just one category
- Driven by one query
- Split search results by fields or aggregations and visualize each field value separately

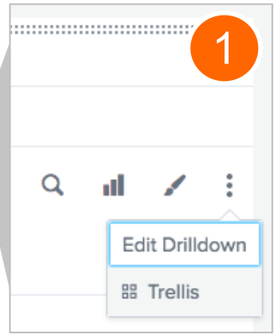


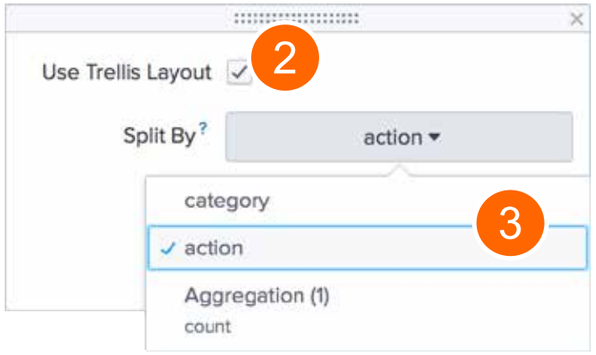
Note

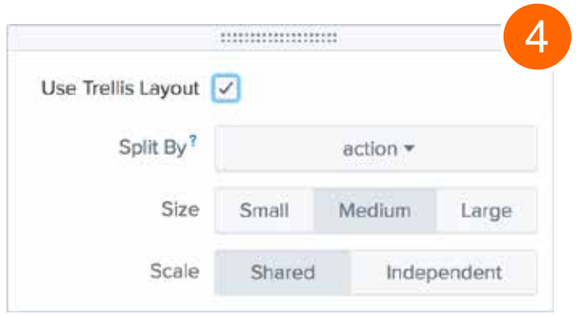
Trellis layout is not available for tables or cluster maps.

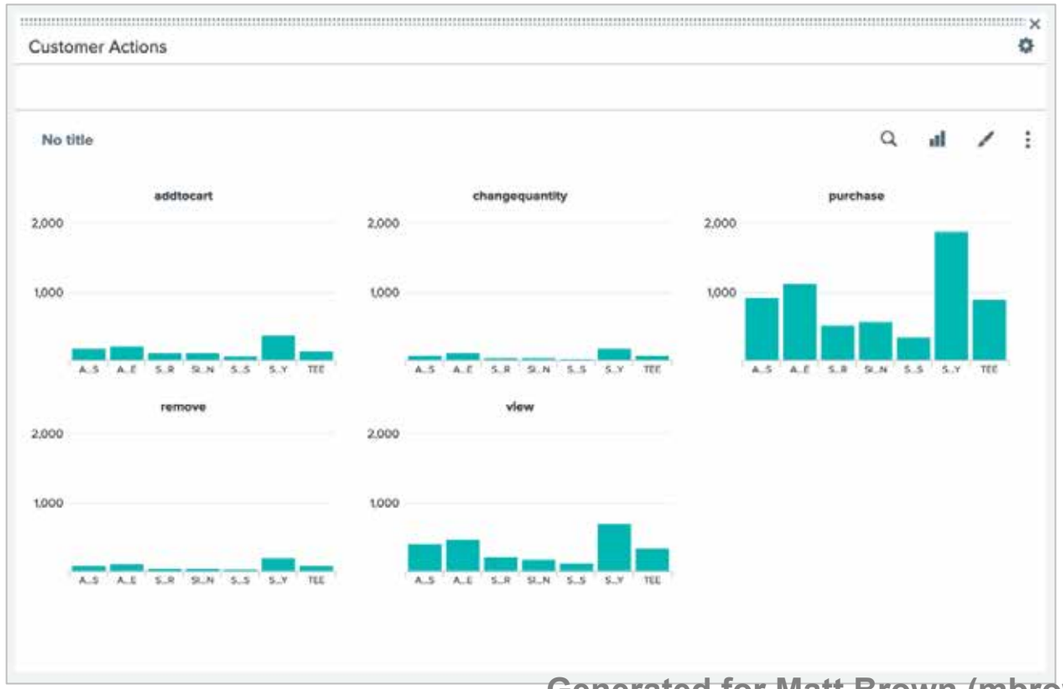
Trellis Visualizations (cont.)







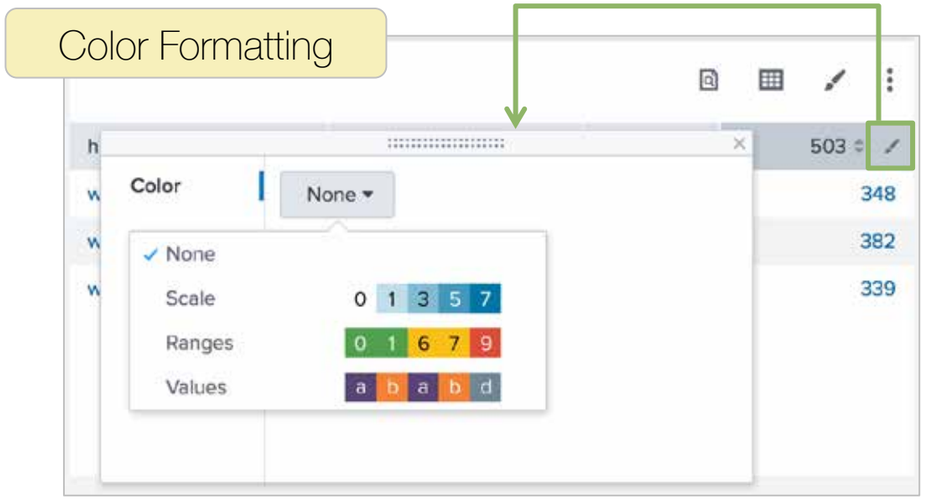


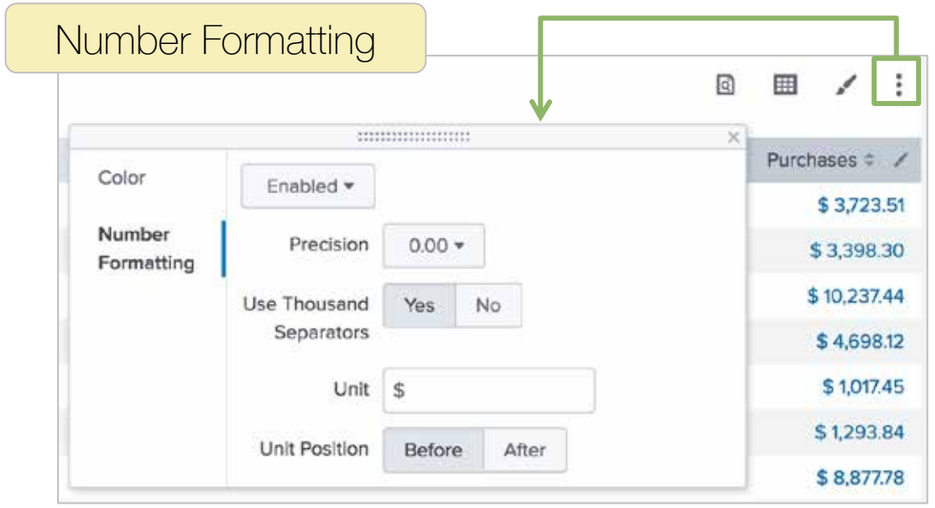



```
...
<panel>
  <title>Customer Actions</title>
  <chart>
    <search>
      <query>sourcetype=access_combined action!=NULL categoryId!=NULL
        | rename categoryId as "category"
        |  chart count by category, action</query>
      <earliest>-30d@d</earliest>
      <latest>now</latest>
      <sampleRatio>1</sampleRatio>
    </search>
    <option name="charting.axisTitleX.visibility">collapsed</option>
    <option name="charting.axisTitleY.visibility">collapsed</option>
    <option name="charting.axisTitleY2.visibility">collapsed</option>
    <option name="charting.chart">column</option>
    <option name="charting.drilldown">none</option>
    <option name="charting.legend.placement">none</option>
    <option name="height">627</option>
    <option name="trellis.enabled">1</option>
    <option name="trellis.scales.shared">1</option>
    <option name="trellis.splitBy">action</option>
  </chart>
</panel>
...
```

Table Formats

- Set color and number format by column
- Color data
  - Scale, range or value
- Format numbers
  - Currency symbols and thousands separators



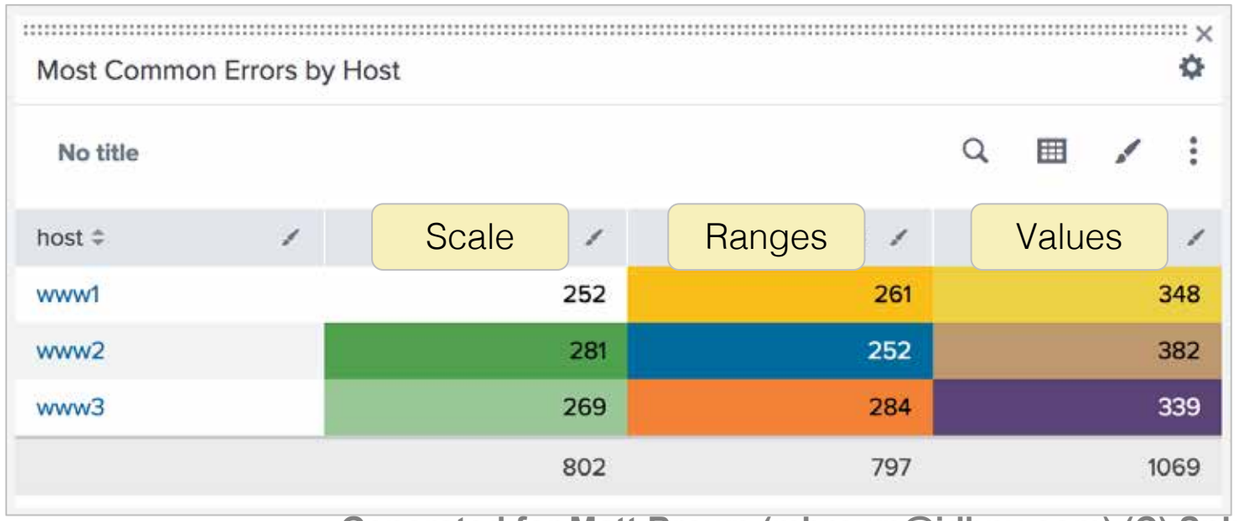


Note

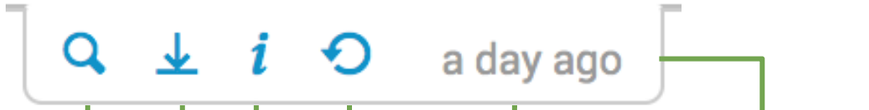
Table formats are discussed in detail in the Splunk Fundamentals 1 course.

Table Formats (cont.)

- Cell colors change based on the scale, range or values you define:
  - **None**: no color coding (default)
  - **Scale**: numeric data
  - **Ranges**: numeric data
  - **Values**: numeric and non-numeric

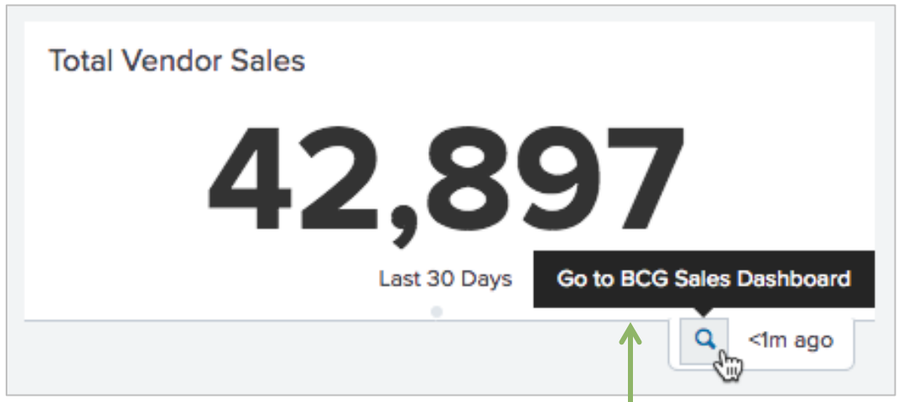


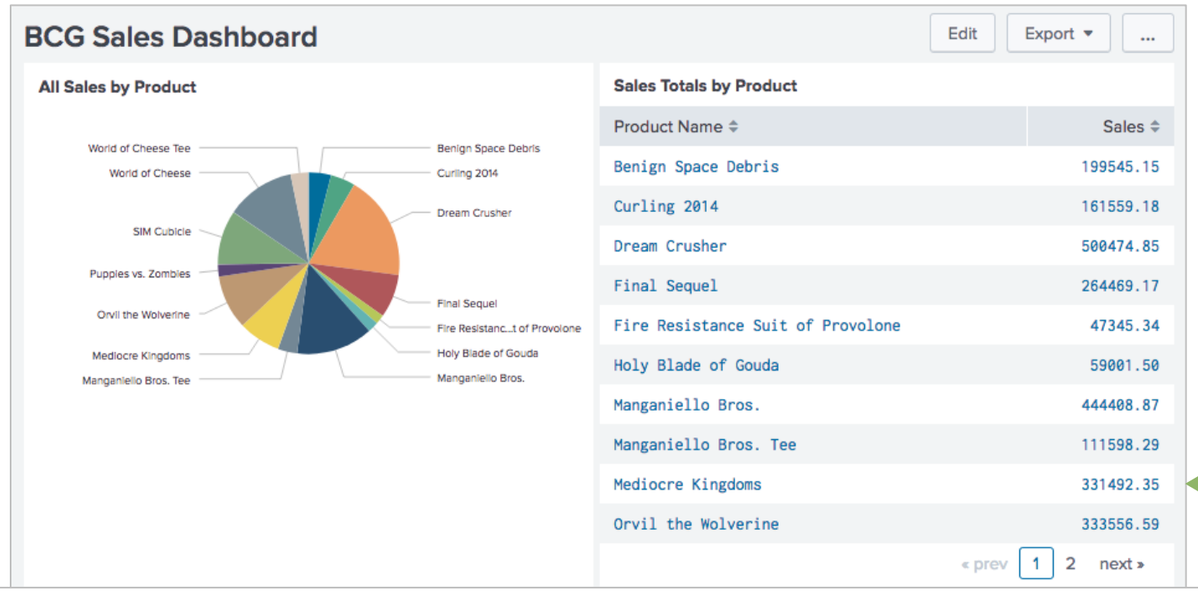
Panel Link Buttons



Property	Type	Description
link.visible	Boolean	Show link buttons at bottom of panel
refresh.time.visible	Boolean	Display the Refresh Time
refresh.link.visible	Boolean	Show the Refresh Link
link.inspectSearch.visible	Boolean	Show the Inspect button
link.exportResults.visible	Boolean	Show the Export Results button
link.openSearch.visible	Boolean	Show the Open Search button
- link.openSearch.search	Search String	Alternative search to use for the Open in Search button
- link.openSearch.text	Text	Label to use for the Open in Search button
- link.openSearch.viewTarget	View Name	Target view for the Open in Search button

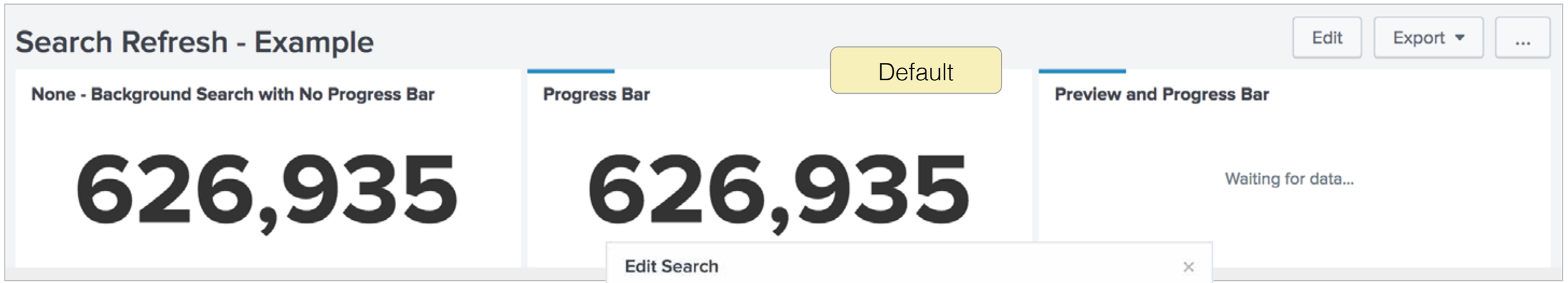
Panel Link Buttons – Example

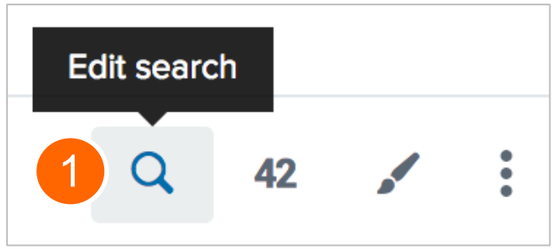


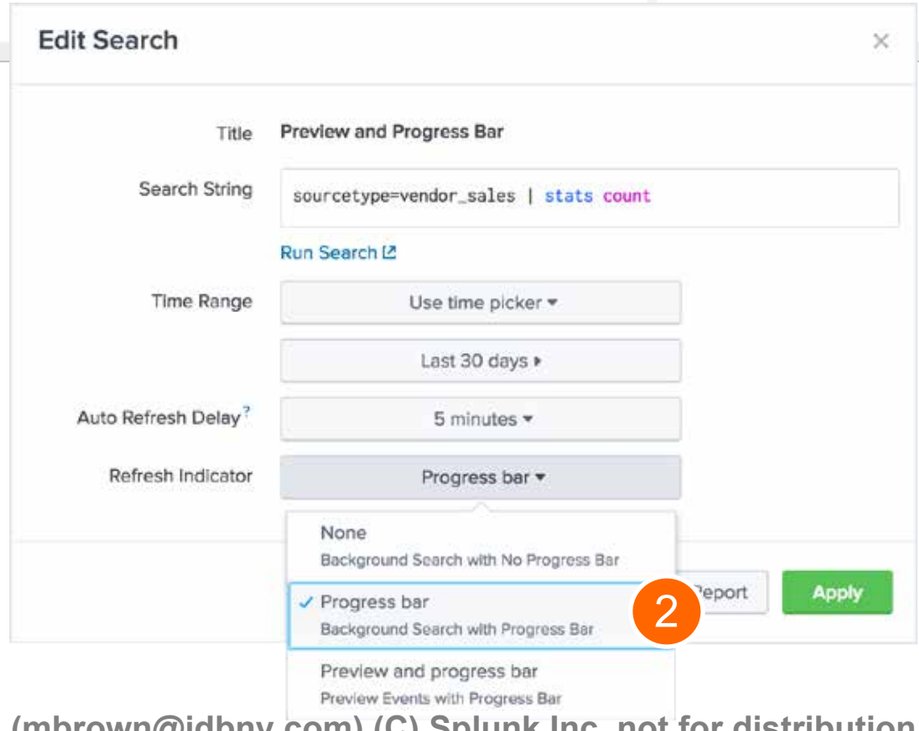



```
...
  <option name="link.exportResults.visible">0</option>
  <option name="link.inspectSearch.visible">0</option>
  <option name="refresh.link.visible">0</option>
  <option name="link.openSearch.text">Go to BCG Sales Dashboard</option>
  <option name="link.openSearch.viewTarget">bcg_sales</option>
  <option name="underLabel">Last 30 Days</option>
 </single>
</panel>
...
```

Search Refresh Indicator



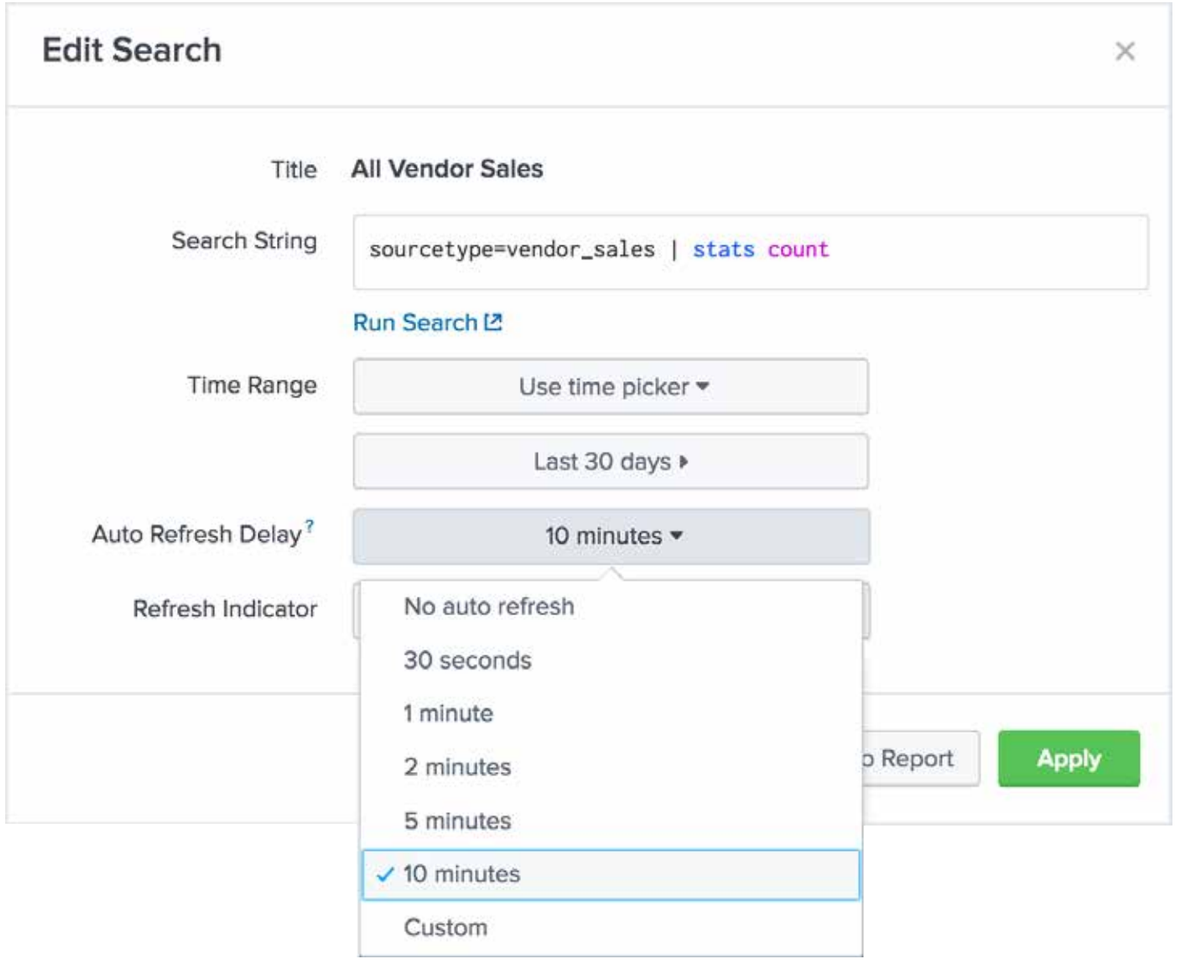




Auto-Refresh

- Inline and reports only
- Enabling for a report or base search requires editing the XML
  - Post process searches will automatically refresh when their base search is refreshed

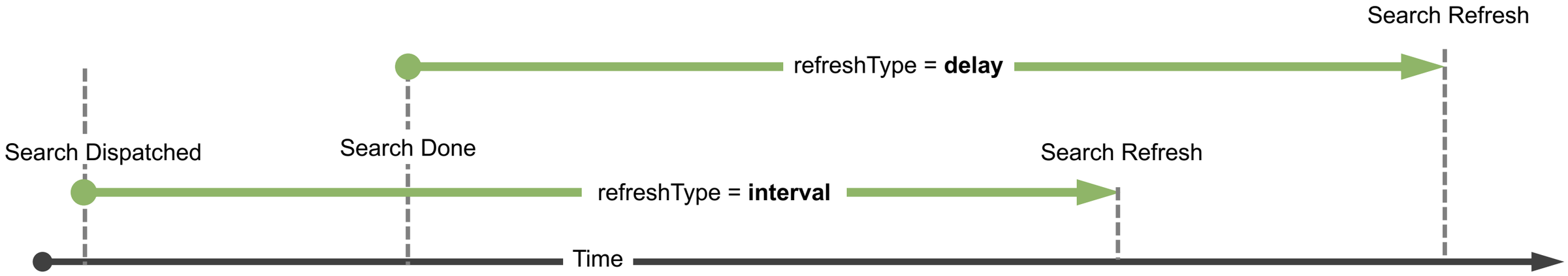
```
...
<panel>
  <single>
    <title>All Vendor Sales</title>
    <search>
      <query>sourcetype=vendor_sales | stats count</query>
      <earliest>-30d@d</earliest>
      <latest>now</latest>
      <sampleRatio>1</sampleRatio>
      <refresh>10m</refresh>
      <refreshType>delay</refreshType>
    </search>
    ...
```



Auto-Refresh (cont.)

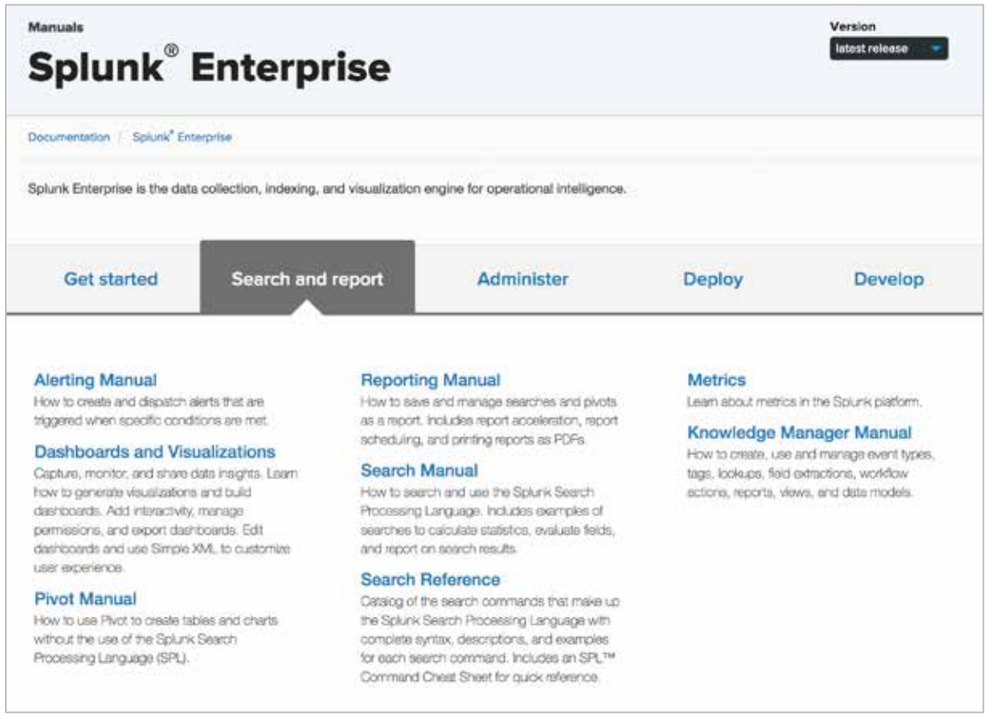
- `refresh`: amount of time between refreshes
  - Default: do not refresh
- `refreshType`: point from which the refresh time is counted
  - delay: start counting down when search is done (default)
  - interval: start counting when search is dispatched

```
...
<search>
  <query>sourcetype=vendor_sales | stats count</query>
  <earliest>-30d@d</earliest>
  <latest>now</latest>
  <sampleRatio>1</sampleRatio>
  <refresh>10m</refresh>
  <refreshType>delay</refreshType>
</search>
...
```



Useful References

- Simple XML Reference
  docs.splunk.com/Documentation/Splunk/latest/Viz/PanelreferenceforSimplifiedXML
- Visualization Reference
  docs.splunk.com/Documentation/Splunk/latest/Viz/Visualizationreference
- Chart Configuration Reference
  docs.splunk.com/Documentation/Splunk/latest/Viz/ChartConfigurationReference
- Table Visualizations
  docs.splunk.com/Documentation/Splunk/latest/Viz/TableFormats



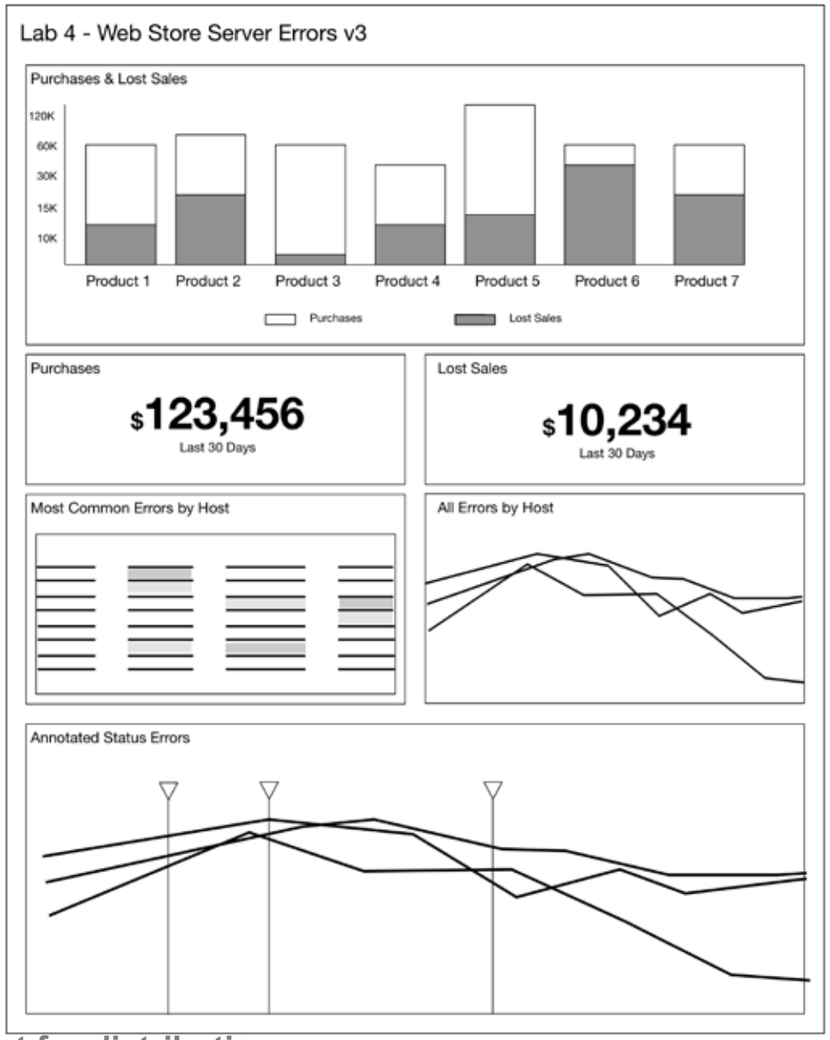
Lab 4 – Customizing Dashboards

Time: 30 minutes

Scenario: Stakeholders have approved the dashboard with some changes. They'd like to use the company's brand colors, hide panel search controls, set refresh time, have the table cell colors match value severity, and have the map display bubbles instead of pies.

Tasks:

- Customize chart colors
- Hide search controls
- Set panel refresh indicator
- Set panel refresh time
- Add column summaries and cell colors
- Add event annotations



Module 5:
Adding Drilldowns

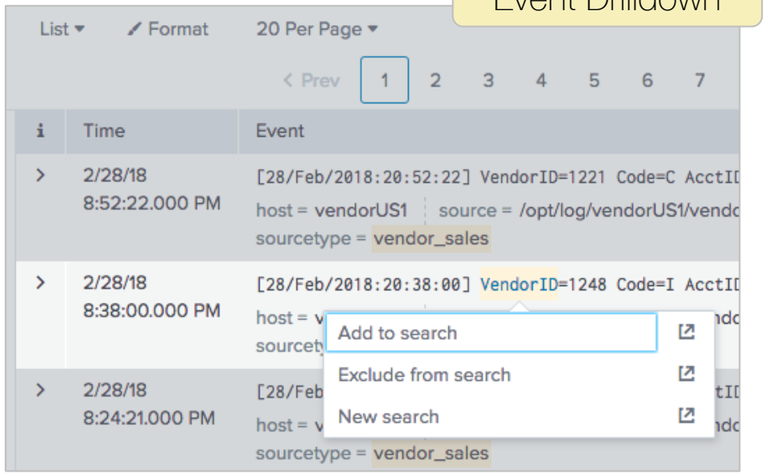
Module Objectives

- Describe four types of drilldowns
- Identify types of predefined drilldown tokens
- Create a dynamic drilldown
- Explain how to create a multiple fields dynamic drilldown

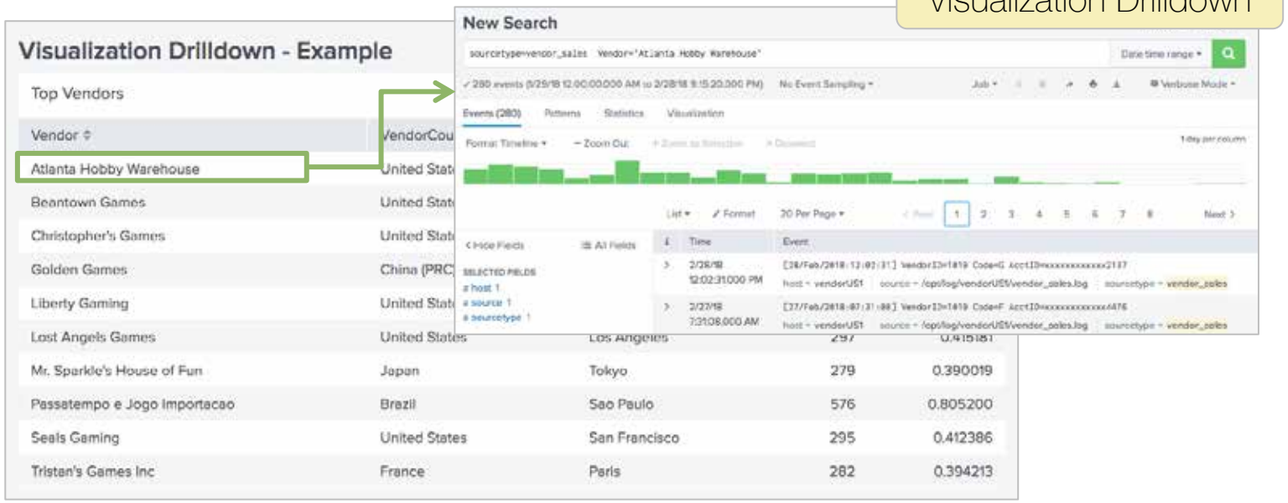
Drilldowns

- Type of event handler
  - Default: event and visualization
  - Custom: Drilldown Editor and XML Editor
- Event drilldown
  - Click an event to see a pop-up menu with options
- Visualization drilldown
  - Clicking on a visualization redirects you to search view

Event Drilldown



Visualization Drilldown

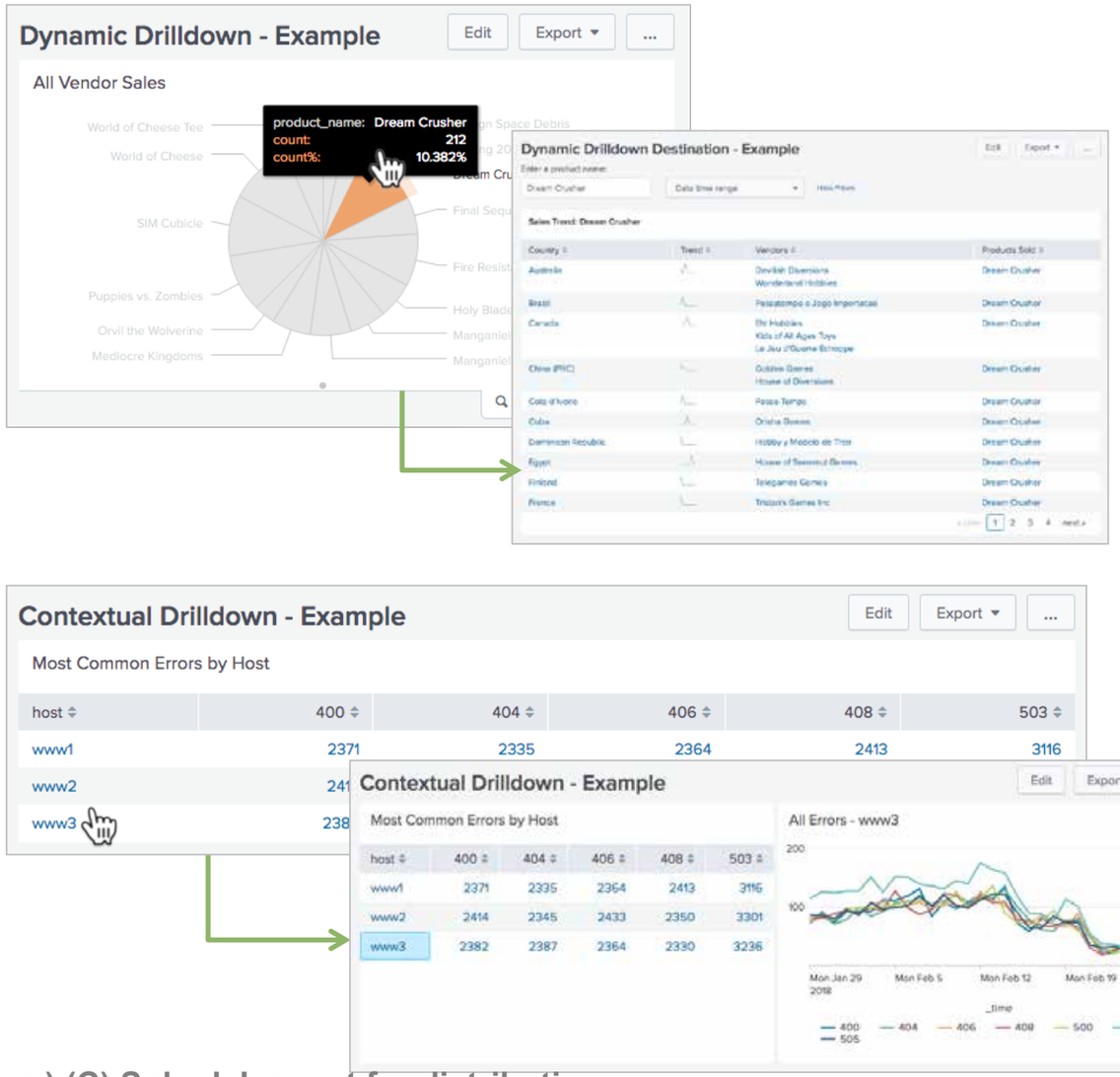


Note

Event handlers are covered in Module 6.

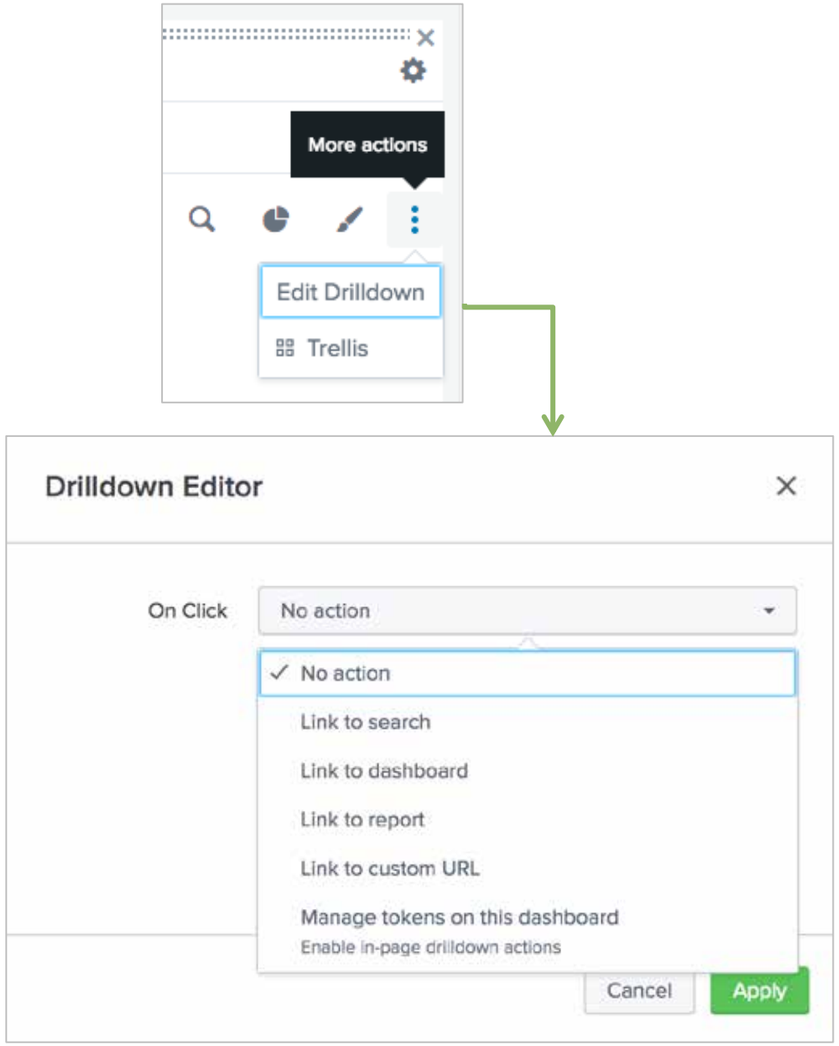
Drilldowns (cont.)

- Dynamic drilldowns
  - Pass a value from a user click to another panel, form, dashboard or external page
- Contextual drilldown
  - Dashboard elements listen for token values to be updated and trigger actions in response

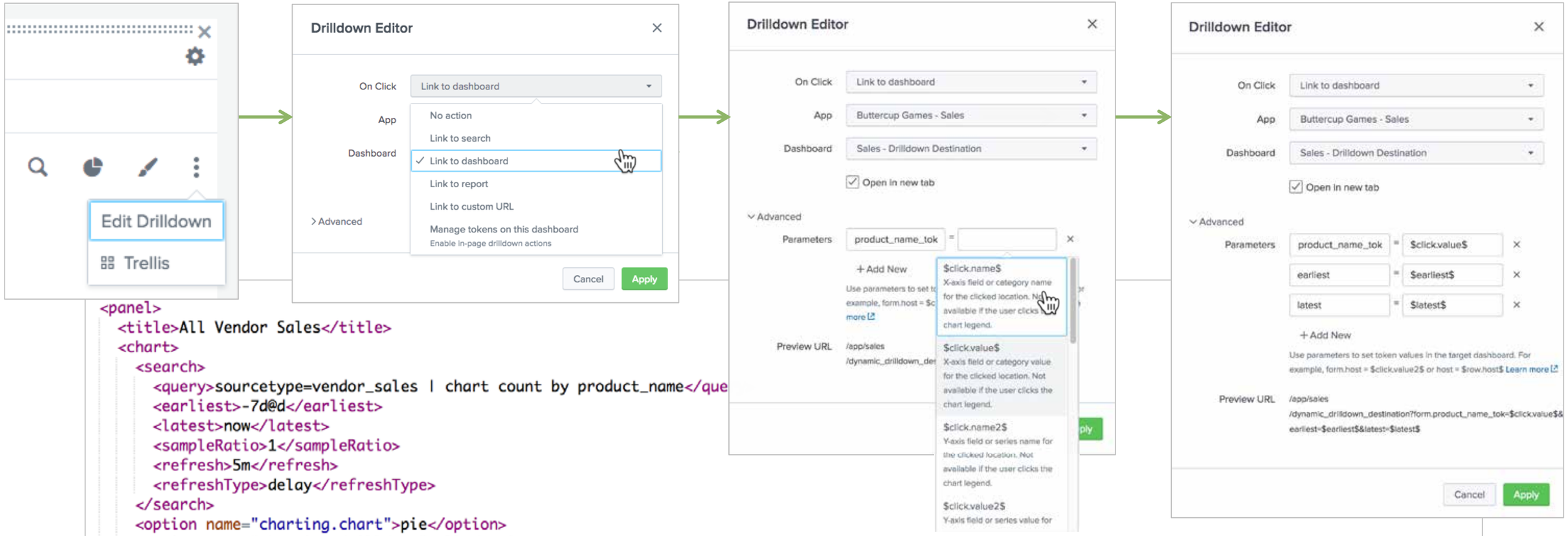


Drilldown Editor

- Use to configure a drilldown action
  - Dynamic and contextual drilldowns
- Specify a path to a destination
- Use `<set>`, `<eval>`, and `<unset>` to update token values
- Adds token filters automatically



Drilldown Editor – Dynamic Drilldown Example



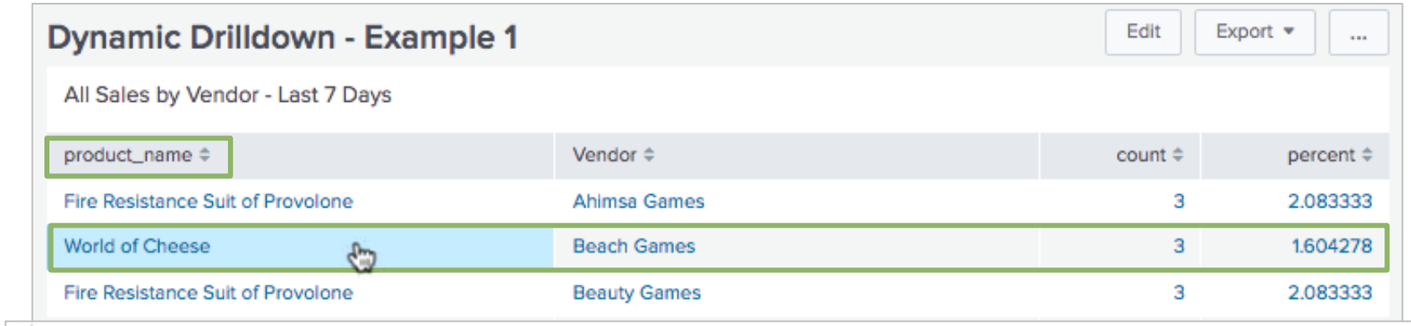

```
<panel>
  <title>All Vendor Sales</title>
  <chart>
    <search>
      <query>sourcetype=vendor_sales | chart count by product_name</que
      <earliest>-7d@d</earliest>
      <latest>now</latest>
      <sampleRatio>1</sampleRatio>
      <refresh>5m</refresh>
      <refreshType>delay</refreshType>
    </search>
    <option name="charting.chart">pie</option>
    <option name="charting.drilldown">all</option>
    <option name="refresh.display">progressbar</option>
    <drilldown>
      <link target="_blank">/app/sales/dynamic_drilldown_destination?form.product_name_tok=$click.value$&amp;earliest=$earliest$&amp;latest=$latest$</link>
    </drilldown>
  </chart>
</panel>
...
```

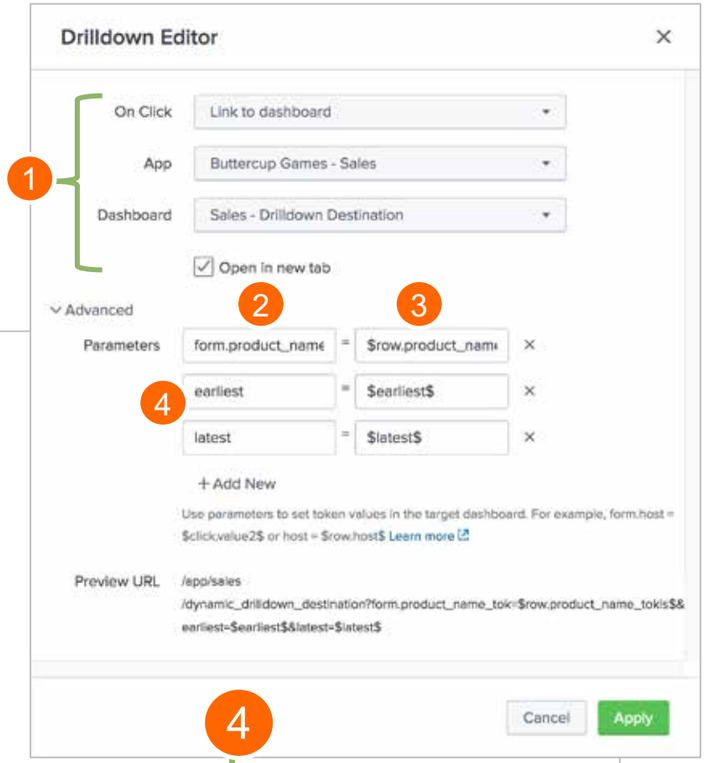
Drilldowns – Predefined Tokens

- Predefined tokens capture information when a user clicks a visualization
- Use different tokens to capture a click location or related data
- Tokens available depend on the visualization type

Dynamic Drilldown – Example 1

Table row: Use the **$row.*field_name*$** token to pass a value from a row click



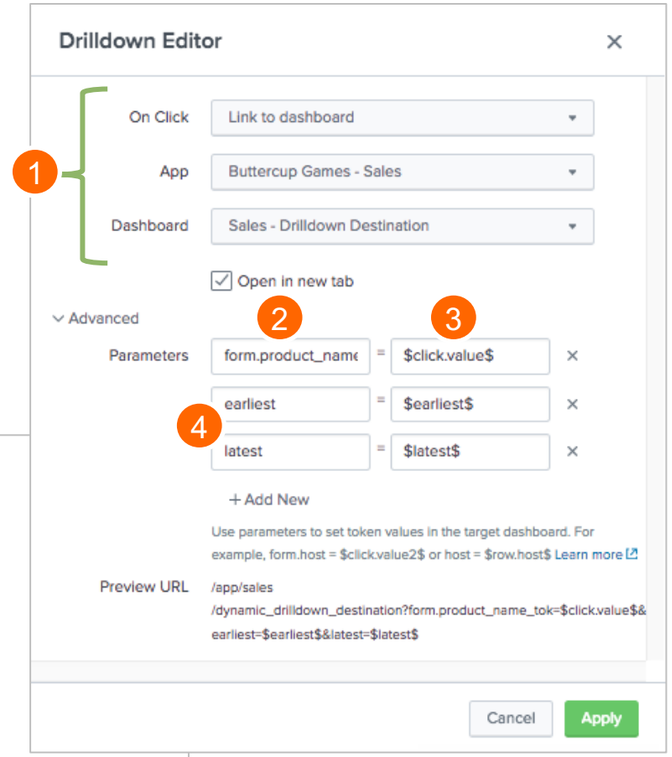



```
...
<panel>
  <title>All Sales by Vendor - Last 7 Days</title>
  <table>
    <search>
      <query>sourcetype=vendor_sales product_name="*" | top Vendor by product_name | sort Vendor</query>
      <earliest>-7d@d</earliest>
      <latest>now</latest>
      <sampleRatio>1</sampleRatio>
    </search>
    <option name="count">10</option>
    <option name="drilldown">cell</option>
    <option name="wrap">true</option>
    <drilldown>
      <link target="_blank">/app/sales/dynamic_drilldown_destination?form.product_name_tok=$row.product_name$&amp;earliest=$earliest$&amp;latest=$latest$</link>
    </drilldown>
  </table>
</panel>
...
```

1 2 3 4

Dynamic Drilldown – Example 2

Chart: Use the **`$click.value$`** token to pass a value from a click




```
...
<panel>
  <title>Top Product Sales - Last 7 Days</title>
  <chart>
    <search>
      <query>sourcetype=vendor_sales | top product_name</query>
      <earliest>-7d@d</earliest>
      <latest>now</latest>
      <sampleRatio>1</sampleRatio>
    </search>
    ...
    <drilldown>
      <link target="_blank">/app/sales/dynamic_drilldown_destination?form.product_name_tok=$click.value$&amp;earliest=$earliest$&amp;latest=$latest$
        </link>
    </drilldown>
  </chart>
</panel>
...
```

1 2 3 4

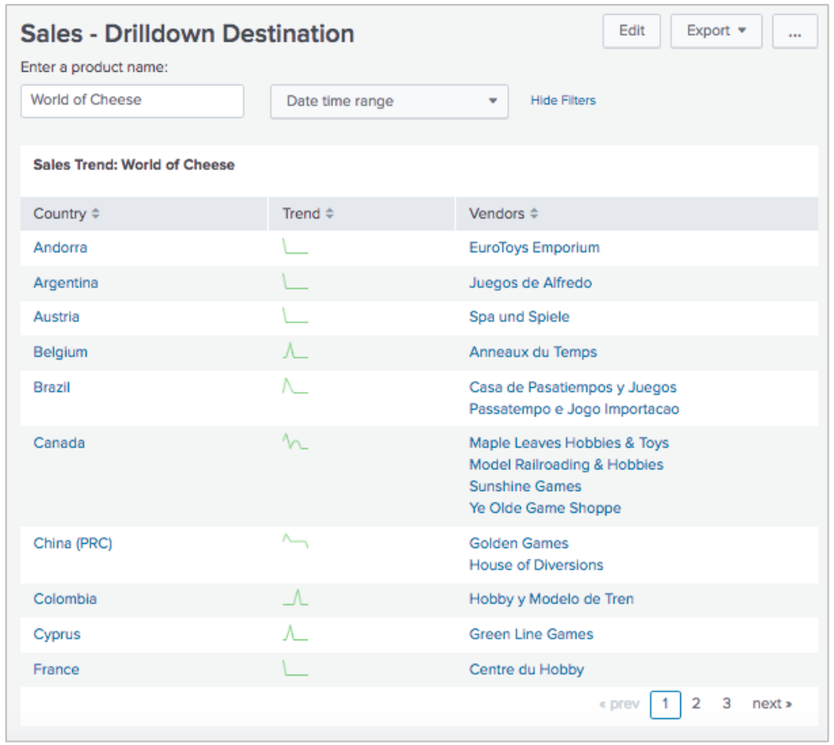
Drilldown Destination Form

Add the token to the destination form's input and search

```
<form>
  <label>Sales - Drilldown Destination</label>
  <fieldset autoRun="true" submitButton="false">
    <input type="text" searchWhenChanged="true" token="product_name_tok">
      <label>Enter a product name:</label>
      <default>*</default>
    </input>
    <input type="time" searchWhenChanged="true">
      <label></label>
      <default>
        <earliest>$earliest$</earliest>
        <latest>$latest$</latest>
      </default>
    </input>
  </fieldset>
  <row>
    <panel>
      <table>
        <title>Sales Trend: $product_name_tok$</title>
        <search>
          <query>sourcetype=vendor_sales product_name=$product_name_tok|s$ | stats sparkline(count) as Trend
          values(Vendor) as "Vendors" by VendorCountry | rename VendorCountry as "Country" | sort Country</query>
          <earliest>$earliest$</earliest>
          <latest>$latest$</latest>
        </search>
        ...
      </table>
    </panel>
  </row>
</form>
```

your custom token from the source dashboard

your custom token from the source dashboard **plus** a token filter to add quotes in the search



Note

Use a token filter in the destination form since best practice is for it to include an input — which means a token; a token on both the drilldown and the destination will not work.

Dynamic Drilldown – Multiple Fields

Use the **`condition`** element with **`field=source_field_name`** and multiple link tags to make multiple fields clickable


```
<drilldown>
 <condition field=source_field_name1>
  <link>/relative_path/view_id?form.target_token1=$row.source_field_name1$</link>
 </condition>
<drilldown>
```

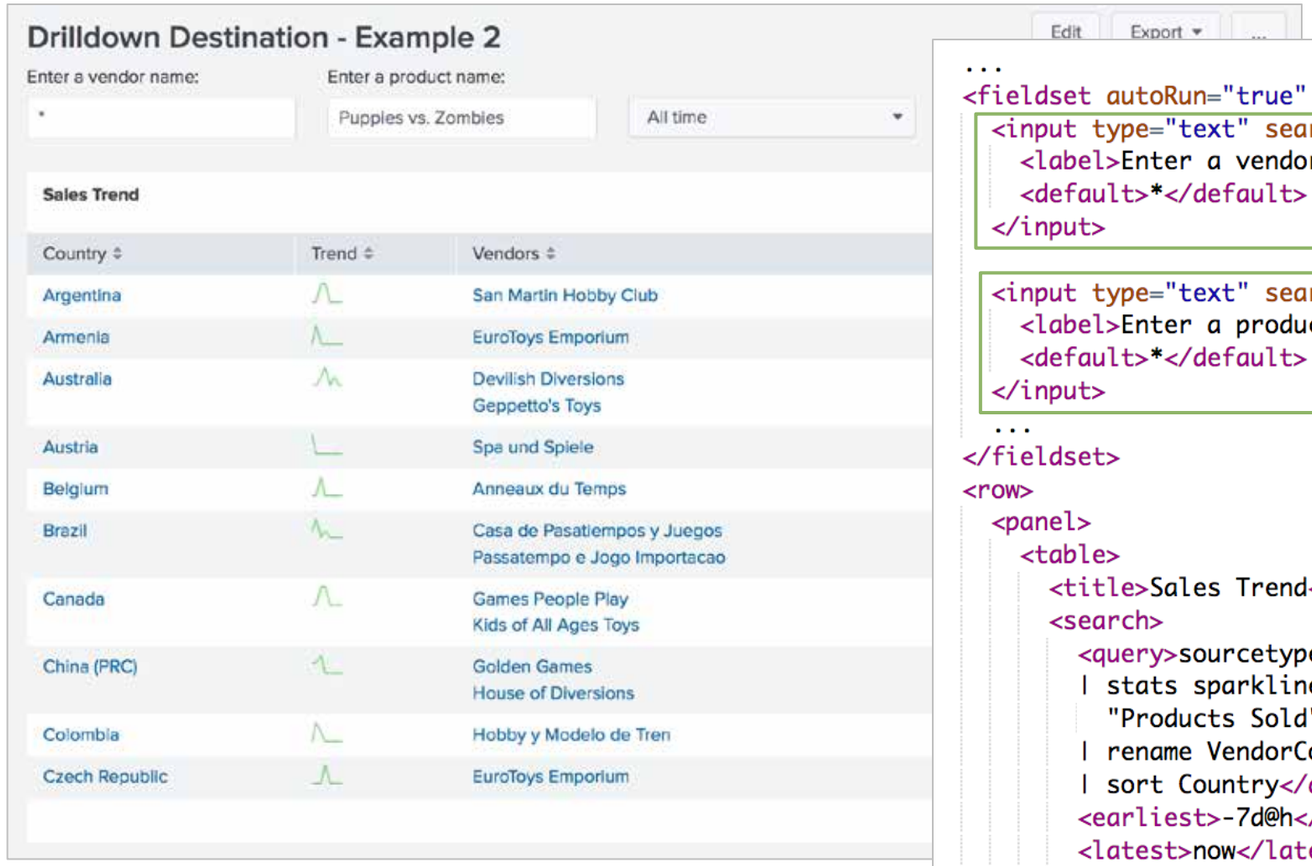
```
    <table>
      <search ref="Top Sales by Vendor and Product"></search>
      <drilldown>

        <condition field="Vendor">
          <link target="_blank">/app/sales/sales_drilldown_dest?form.vendor_tok=$row.Vendor$
          </link>
        </condition>

        <condition field="product_name">
          <link target="_blank">/app/sales/sales_drilldown_dest?form.product_name_tok=$row.product_name$
          </link>
        </condition>

      </drilldown>
    </table>
  </panel>
 </row>
</dashboard>
```

Note

Multiple field dynamic drilldown is not available in the drilldown editor. It requires editing the XML.

Drilldown Destination Form – Multiple Fields

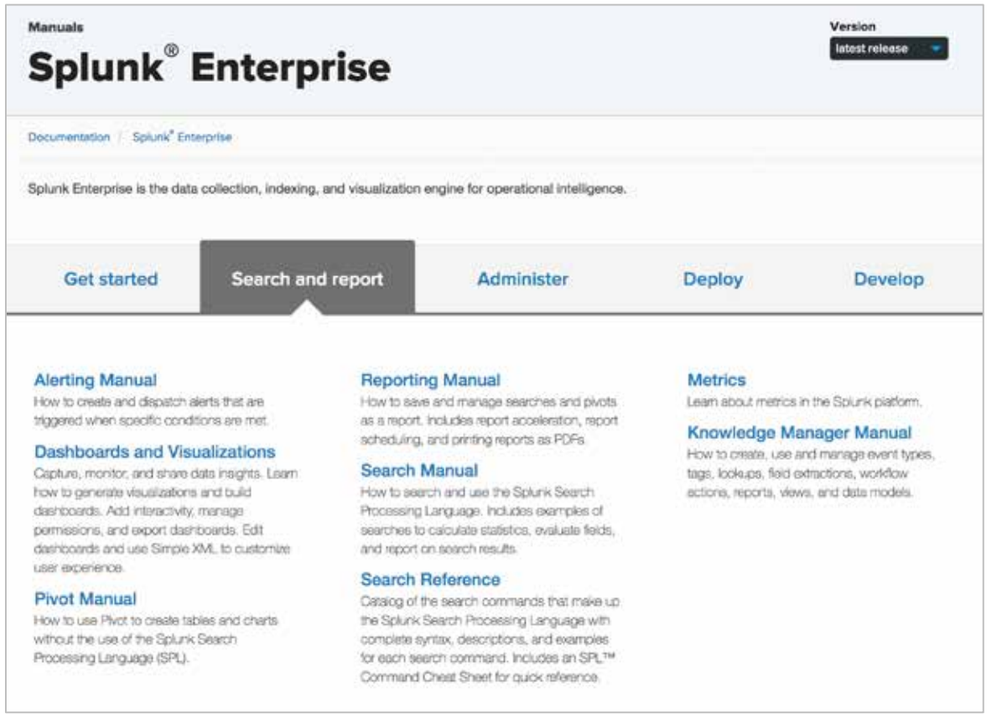
Include an input for each token being passed to the destination form




```
...
<fieldset autoRun="true" submitButton="false">
  <input type="text" searchWhenChanged="true" token="vendor_tok">
    <label>Enter a vendor name:</label>
    <default>*</default>
  </input>

  <input type="text" searchWhenChanged="true" token="product_name_tok">
    <label>Enter a product name:</label>
    <default>*</default>
  </input>
  ...
</fieldset>
<row>
  <panel>
    <table>
      <title>Sales Trend</title>
      <search>
        <query>sourcetype=vendor_sales product_name="$product_name_tok$" Vendor="$vendor_tok$"
        | stats sparkline(count) as Trend values(Vendor) as "Vendors" values(product_name) as
          "Products Sold" by VendorCountry
        | rename VendorCountry as "Country"
        | sort Country</query>
        <earliest>-7d@h</earliest>
        <latest>now</latest>
      </search>
      ...
```

Useful References

- Use Drilldown for Dashboard Interactivity
  docs.splunk.com/Documentation/Splunk/latest/Viz/DrilldownIntro
- Drilldown Elements
  docs.splunk.com/Documentation/Splunk/latest/Viz/PanelreferenceforSimplifiedXML#drilldown
- Predefined Drilldown Tokens
  docs.splunk.com/Documentation/Splunk/latest/Viz/PanelreferenceforSimplifiedXML#Predefined_drilldown_tokens



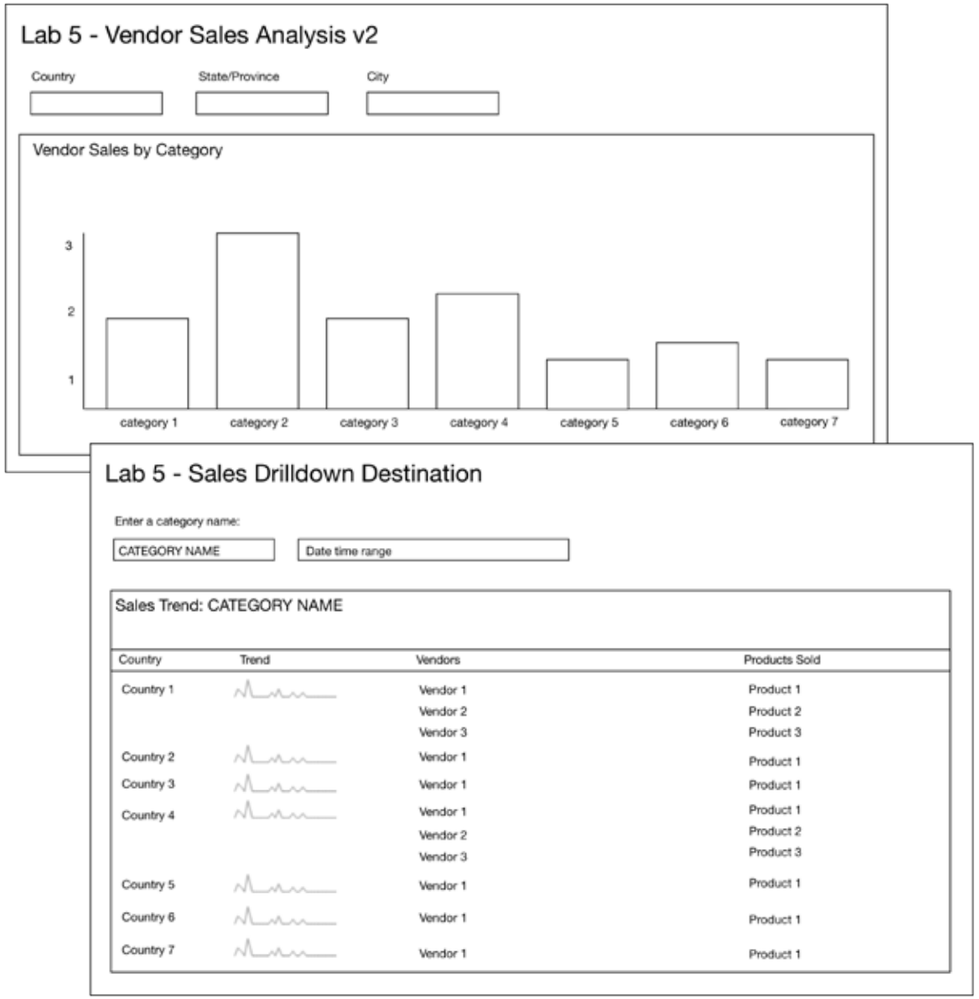
Lab 5 – Add a Dynamic Drilldown

Time: 25 minutes

Scenario: The sales team wants to add a dynamic drilldown from the Sales by Category panel to a form that displays sales trends by category

Tasks:

- Create a drilldown destination form
- Add text and time inputs
- Clone a form
- Create a dynamic drilldown



Module 6: Adding Advanced Behaviors & Visualizations

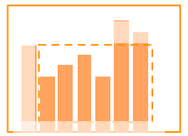
Module Objectives

- Identify types of event handlers
- Define event actions
- Define custom visualizations
- Use simple XML extensions

Event Handlers

Form Input
input value change

Search
status of a search job

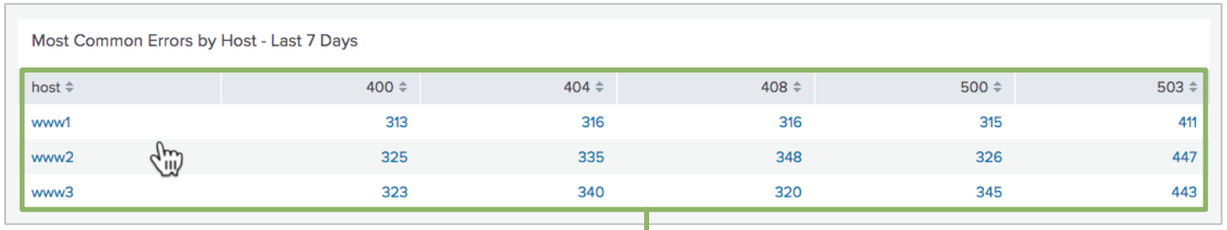


Visualization
user selections and dynamic drilldowns

- Use event handlers with event actions
  - Tokens available to each handler vary
- Perform event actions based on user behavior or search job status:
  - Execute an eval statement
  - Link to another page
  - Set & unset tokens

Event Handler – Example

- Contextual Drilldown
  - Table
    - Captures the value of the host field
    - Sets a token to make the chart visible
    - Populates the chart panel's search and title with the host value
  - Chart
    - Unsets the token to make the chart hidden




```
...
<panel>
 <table>
  <search>
   <query>sourcetype=access_combined status&gt;399
   | chart count by host, status limit=5 useother=f</query>
    <earliest>-30d@d</earliest>
    <latest>now</latest>
     <sampleRatio>1</sampleRatio>
  </search>
  <drilldown>
    <set token="host_errors_tok">$click.value$</set>
  </drilldown>
  </table>
 </panel>
 <panel>
 <title>All Errors - $host_errors_tok$</title>
 <chart depends="$host_errors_tok$">
  <search>
   <query>sourcetype=access* host="$host_errors_tok$" status&gt;399
    | timechart count by status</query>
    <earliest>-30d@d</earliest>
    <latest>now</latest>
    <sampleRatio>1</sampleRatio>
  </search>
...
  <drilldown>
    <unset token="host_errors_tok"></unset>
  </drilldown>
  </chart>
...
```

Event Handler

Event Action

Event Handler

Event Action

Event Actions – Execute an Eval Statement

- `<eval>`
  - Execute an eval statement and put the results into a token

```
<eval token="myToken">tostring(round(field))</eval>
```

Syntax by Event Handler

Form Input

puppies vs. zom

```
<change>
  <eval token="[token_name]">
```

Search

```
<search>
 <done | error | fail | cancelled | progress>
  <eval token="[token_name]">
```

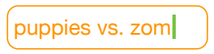
Visualization

```
<[visualization]>
 <drilldown>
  <eval token="[token_name]">
 <selection>
  <eval token="[token_name]">
```

Event Actions – Link to Another Page

- `<link>`
  - Specify a destination for a dynamic drilldown, selected input, or search
    - Dashboard `<link>/app/myApp/myView</link>`
    - Form `<link>/app/myApp/myView?form.token=$token$</link>`
    - URL `<link>myURL?q=$token$</link>`

Syntax by Event Handler

Form Input 


```
<change>
 <link>/app/appName/viewName
```

Search

```
<search>
  <done | error | fail | cancelled | progress>
  <link>/app/appName/viewName
```

Visualization

```
<selection>
 <link>/app/appName/viewName
```

Event Actions – Set & Unset Tokens

- <set>
  - Specify tokens for actions to take for specific inputs

```
<set token="myToken">$text$</set>
```

- <unset>
  - Remove a token that was previously set

```
<unset token="myToken"></unset>
```

Syntax by Event Handler

Form Input puppies vs. zom

```
<change>
 <set token="[token_name]">
 <unset token="[token_name]">
```

Search

```
<search>
 <done | error | fail | cancelled | progress>
 <set token="[token_name]">
 <unset token="[token_name]">
```

Visualization

```
<selection>
 <set token="[token_name]">
 <unset token="[token_name]">
```

depends & rejects Attributes

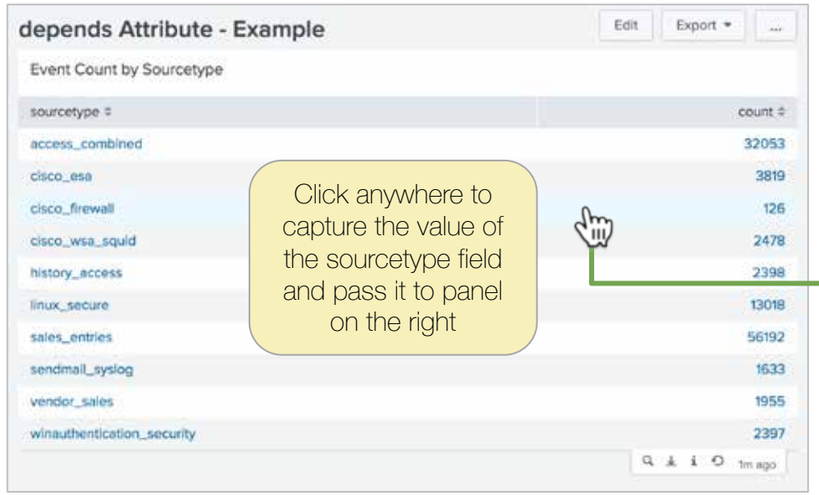
- Show or hide content based on existence of a token
  - <row>, <panel>, <chart>, <table>, <events>, <input>, <search>
- Use with event handlers, drilldowns, condition element
- Can use multiple tokens, separated by commas
- depends: hide by default, show when token exists

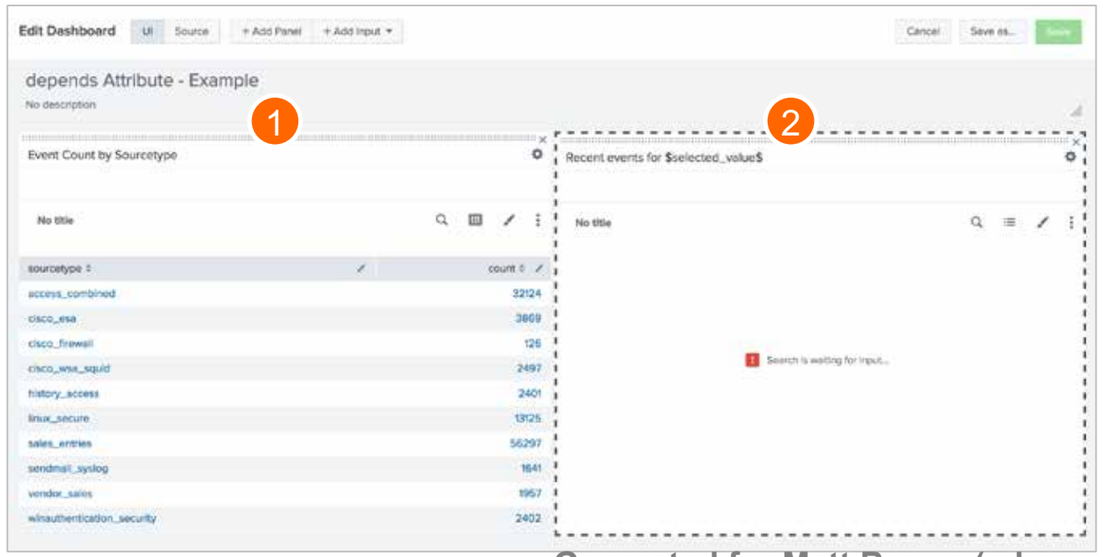
```
<panel depends="$myPanel$"></panel>
```

- rejects: show by default, hide when token exists

```
<panel rejects="$myPanel$"></panel>
```

depends Attribute – Example






```
<dashboard>
 <label>depends Attribute - Example</label>
 <row>
   <panel>
     <title>Event Count by Sourcetype</title>
     <table>
       <search>
         <query>index=main | stats count by sourcetype</query>
         <earliest>-7d@d</earliest>
         <latest>now</latest>
         <sampleRatio>1</sampleRatio>
       </search>
       <option name="count">20</option>
       <drilldown>
         <set token="myPanel">true</set>
         <set token="selected_value">$click.value$</set>
       </drilldown>
     </table>
   </panel>
   <panel depends="$myPanel$">
     <title>Recent events for $selected_value$</title>
     <event>
       <search>
         <query>index=main sourcetype=$selected_value$</query>
         <earliest>-7d@d</earliest>
         <latest>now</latest>
         <sampleRatio>1</sampleRatio>
       </search>
       <option name="count">5</option>
       <drilldown>
         <unset token="myPanel"></unset>
         <unset token="selected_value"></unset>
       </drilldown>
     </event>
   </panel>
 </row>
</dashboard>
```

1

2

Form Input Event Handler

- `<change>`
- Set and unset tokens based on user input selections
- Access an input label or choice using predefined tokens
  - `label`: captures label of an input
  - `value`: captures value of an input choice
- Available for these inputs:
  - Checkbox, dropdown, link, radio, text, time

```
<input type="radio">
 <change>
  <set token="MyInputName">$label$</set>
  <set token="MyInputChoice">$value$</set>
 </change>
</input>
```

Example
- Handler: <change>
- Input: radio button
- Event Action: set token

Syntax

```
<change>
 <eval> | <link> | <set> | <unset>
```

Form Input Event Handler – Example 1

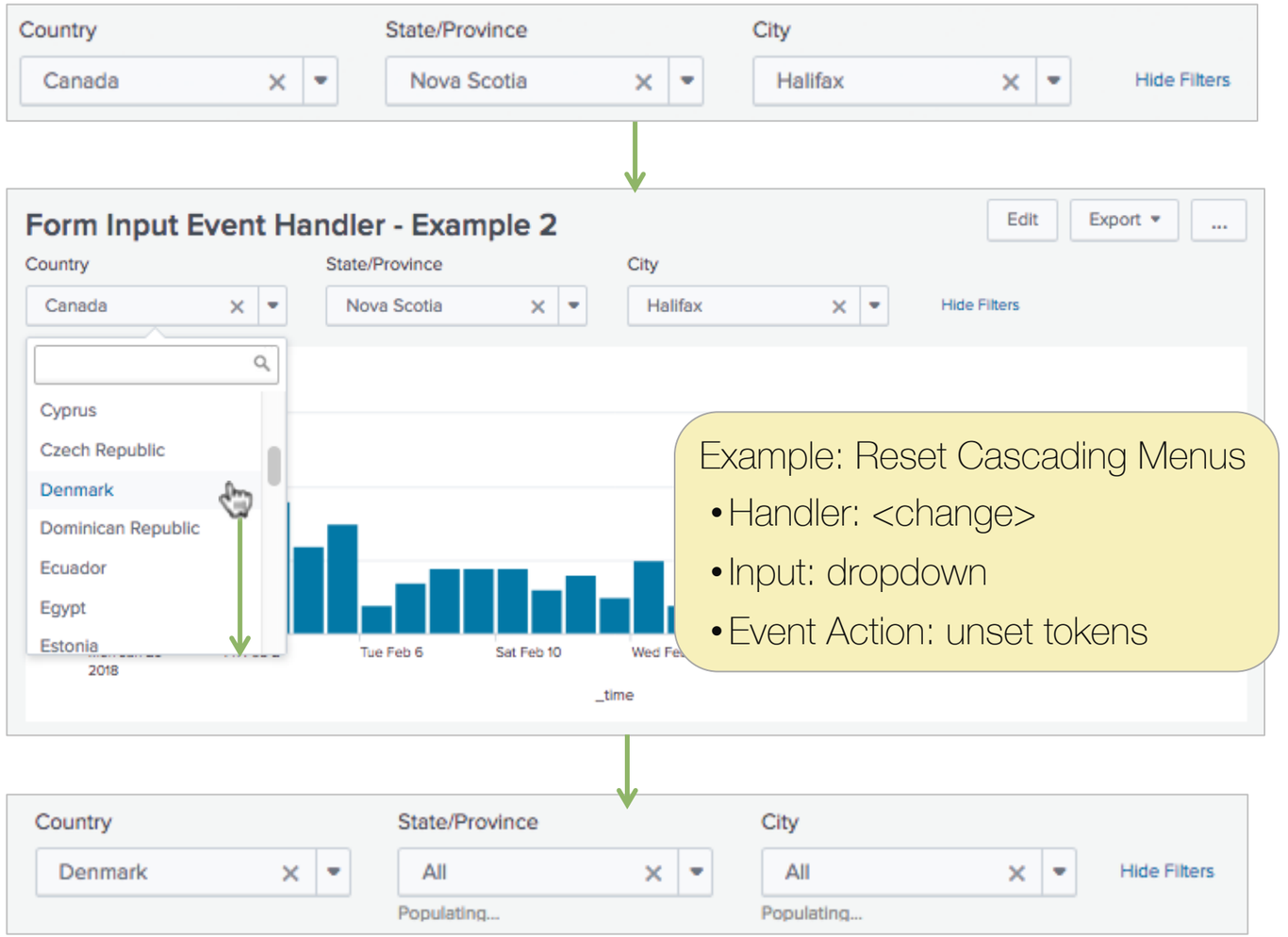
```
<form>
 <label>Form Input Event Handler - Example</label>
 <fieldset submitButton="false">
   <input type="radio">
     <label>Select a time range</label>
     <choice value="-24h@h">Last 24 Hours</choice>
     <choice value="-7d@d">Last 7 Days</choice>
     <choice value="-30d@d">Last 30 Days</choice>
     <default>Last 24 Hours</default>
     <change>
       <set token="date_label_tok">$label$</set>
       <set token="earliest_tok">$value$</set>
     </change>
   </input>
 </fieldset>
 <row>
   <panel>
     <title>Strategy Game Sales - $date_label_tok$</title>
     <chart>
       <search>
         <query>index=main sourcetype="access_combined" categoryId=STRATEGY
         | timechart count by product_name usenull=f</query>
         <earliest>$earliest_tok$</earliest>
         <latest>now</latest>
       </search>
       ...
```

Example: Use a form input's label and value in a panel title and search

- Handler: <change>
- Input: radio button
- Event Action: set token

Form Input Event Handler – Example 2

```
...
<fieldset submitButton="false">
  <input type="dropdown" token="v_country_tok" searchWhenChanged="true">
    <label>Country</label>
    <choice value="*">All</choice>
    <default>*</default>
    <fieldForLabel>VendorCountry</fieldForLabel>
    <fieldForValue>VendorCountry</fieldForValue>
    <search>
      <query>| inputlookup bcg_vendors | stats count by "VendorCountry"</query>
      <earliest>-7d@h</earliest>
      <latest>now</latest>
    </search>
    <change>
      <unset token="form.v_state_tok"></unset>
      <unset token="form.v_city_tok"></unset>
    </change>
  </input>

  <input type="dropdown" token="v_state_tok" searchWhenChanged="true">
    <label>State/Province</label>
    <choice value="*">All</choice>
    <default>*</default>
    <fieldForLabel>VendorStateProvince</fieldForLabel>
    <fieldForValue>VendorStateProvince</fieldForValue>
    <search>
      <query> | inputlookup bcg_vendors | search VendorCountry=$v_country_tok|s$
              | stats count by "VendorStateProvince"</query>
      <earliest>-7d@h</earliest>
      <latest>now</latest>
    </search>
    <change>
      <unset token="form.v_city_tok"></unset>
    </change>
  </input>
  ...
```



Search Event Handler

- `<progress>`, `<done>`, `<cancelled>`, `<error>`, `<fail>`
- Access search results or search job properties using predefined tokens
- Predefined tokens:
  - **`result`**.*<field>*
    captures named value from the first row of returned results
  - **`job`**.*<property>*
    captures named job property's value

```
<search>
 <query>index=main sourcetype="access_combined"
   | timechart count by product_name usenull=f
 </query>
  <earliest>-7d@d<earliest>
  <latest>now</latest>
 <done>
  <set token="MyEventCount">$job.eventCount$</set>
 </done>
</search>
```

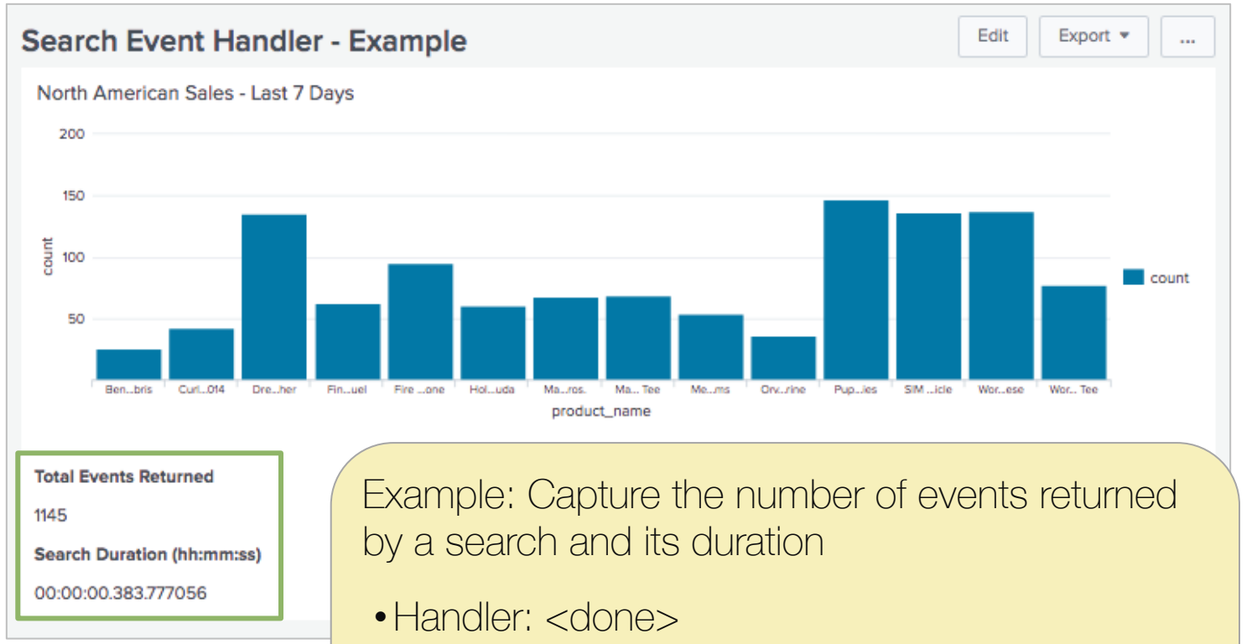
Example: Capture the number of events returned by a search
- Handler: <done>
- Job Property: eventCount
- Event Action: set token

Syntax

```
<search>
 <progress> | <done> | <cancelled> | <error> | <fail>
  (<eval> | <link> | <set> | <unset> | <condition>)
```

Search Event Handler – Example

```
<dashboard>
 <label>Search Event Handler - Example</label>
 <row>
   <panel>
     <title>North American Sales - Last 7 Days</title>
     <chart>
       <search>

        <done>
          <eval token="mySearchDuration">tostring(tonumber('job.runDuration'),"duration")</eval>
          <set token="myEventCount">$job.eventCount$</set>
        </done>

         <query>sourcetype=vendor_sales VendorID&lt;4000 | chart count by product_name</query>
         <earliest>-7d@h</earliest>
         <latest>now</latest>
         <sampleRatio>1</sampleRatio>
       </search>
       <option name="charting.chart">column</option>
       <option name="refresh.display">progressbar</option>
     </chart>

     <html>
       <h3>Total Events Returned</h3>
        <div>$myEventCount$</div>

       <h3>Search Duration (hh:mm:ss)</h3>
        <div>$mySearchDuration$</div>
     </html>

   </panel>
 </row>
</dashboard>
```



Example: Capture the number of events returned by a search and its duration

- Handler: <done>
- Job Properties: eventCount, runDuration
- Event Actions: set token, eval statement

Note

See a list of search job properties in the Search Job Inspector

Visualization Event Handler

- `<selection>`
  - Pan & Zoom
- Predefined tokens
  - **`start / end`**
    Captures the value of the x-axis (time) at the beginning and end of a selection
  - **`start.<field>/end.<field>`**
    Captures specified series' value in the y-axis at the beginning and end of a selection
  - **`selection_earliest/selection_latest`**
    Accesses the selected time range
- Available for: area, column, line charts

Example: Pan & Zoom
- Handler: <selection>
- Event Action: set token

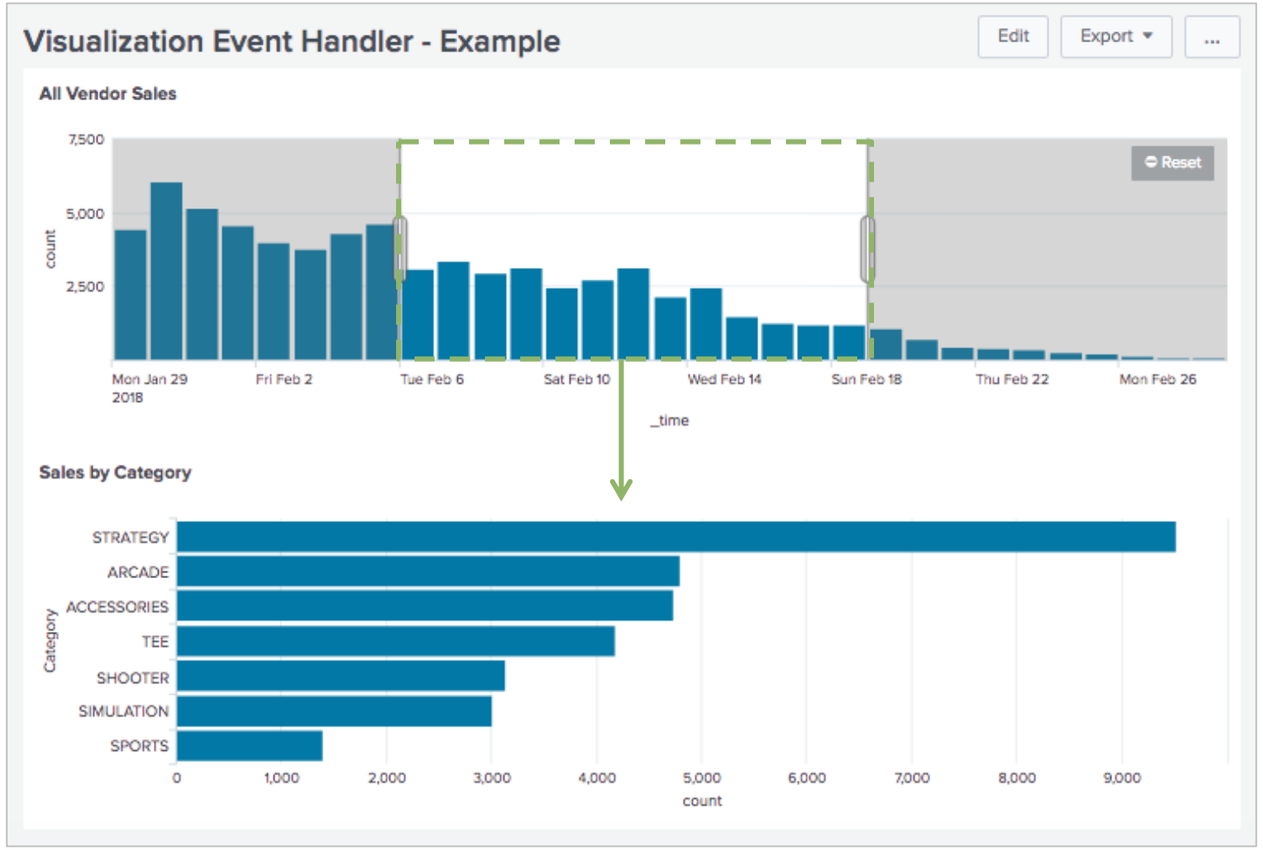
```
<chart>
...
<option name="charting.chart">column</option>
<selection>
  <set token="selection.earliest">$start$</set>
  <set token="selection.latest">$end$</set>
</selection>
</chart>
<chart>
 <title>Sales by Category</title>
 <search><query>sourcetype="vendor_sales"
   | top categoryId</query>
  <earliest>$selection.earliest$</earliest>
  <latest>$selection.latest$</latest>
</search>
<option name="charting.chart">bar</option>
</chart>
...
```

Syntax

```
<[visualization]>
 <selection>
  (<eval> | <link> | <set> | <unset> )
```

Visualization Event Handler – Example

Predefined tokens implement the selection on a chart



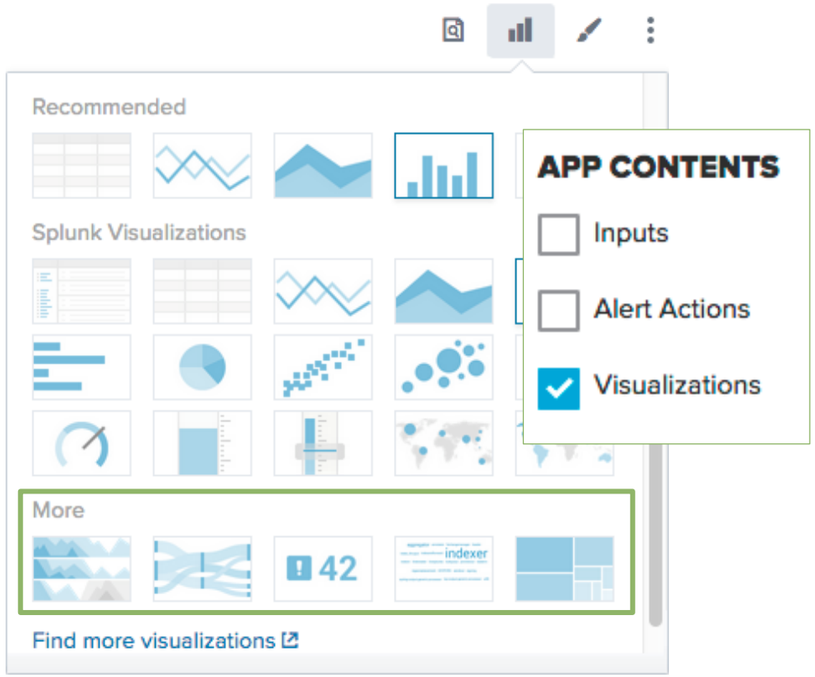
Example: Pan & Zoom
- Handler: <selection>
- Event Action: set token

```
<dashboard>
  <label>Selection Event Handler</label>
  <row>
    <panel>
      <chart>
        <title>All Vendor Sales</title>
        <search>
          <query>sourcetype=vendor_sales | timechart count</query>
          <earliest>0</earliest>
          <latest></latest>
        </search>
        <selection>
          <set token="selection.earliest">$start$</set>
          <set token="selection.latest">$end$</set>
        </selection>
      </chart>
      <chart>
        <title>Sales by Category</title>
        <search>
          <query>sourcetype="vendor_sales" | top categoryId
          | rename categoryId as Category</query>
          <earliest>$selection.earliest$</earliest>
          <latest>$selection.latest$</latest>
        </search>
        ...
```

Custom Visualizations

ADM DEV UX

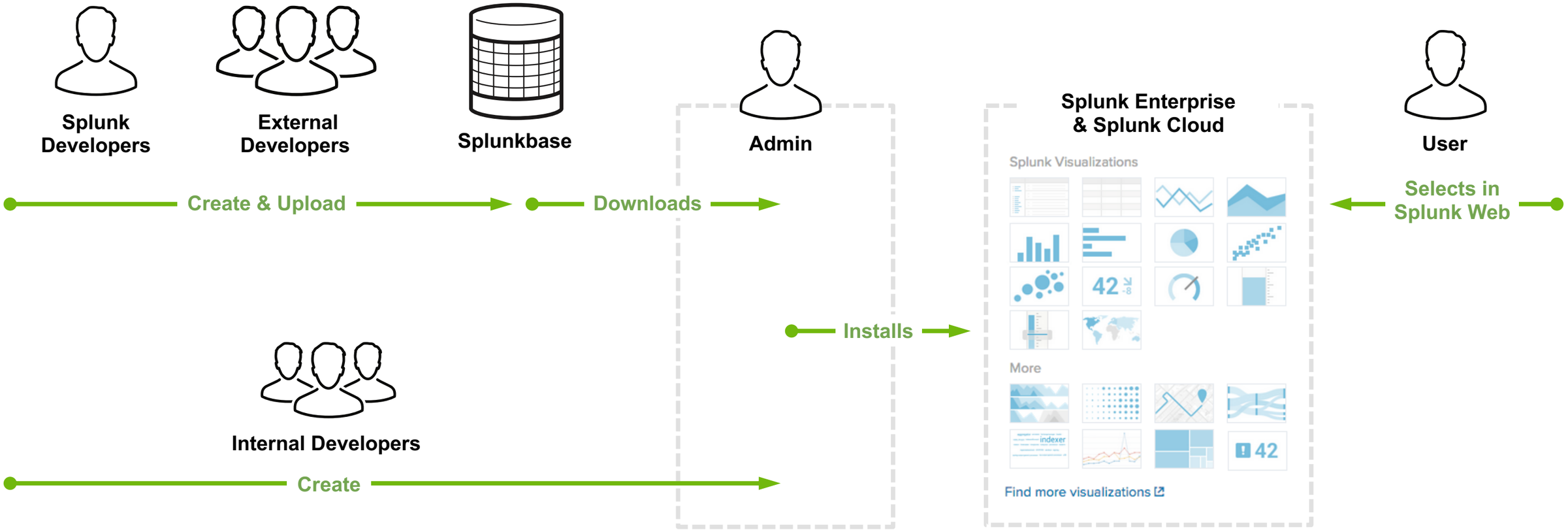
- Shareable visualizations
- Distributed with an app
- Install from splunkbase or build your own
- Many Splunk supported including:
  - Horizon Chart
  - Horseshoe Meter
  - Punchcard
  - Status Indicator
  - Treemap
  - Location Tracker
  - Parallel Coordinates
  - Timeline
  - Bullet Graph
  - Sankey Chart
  - Calendar Heat Map



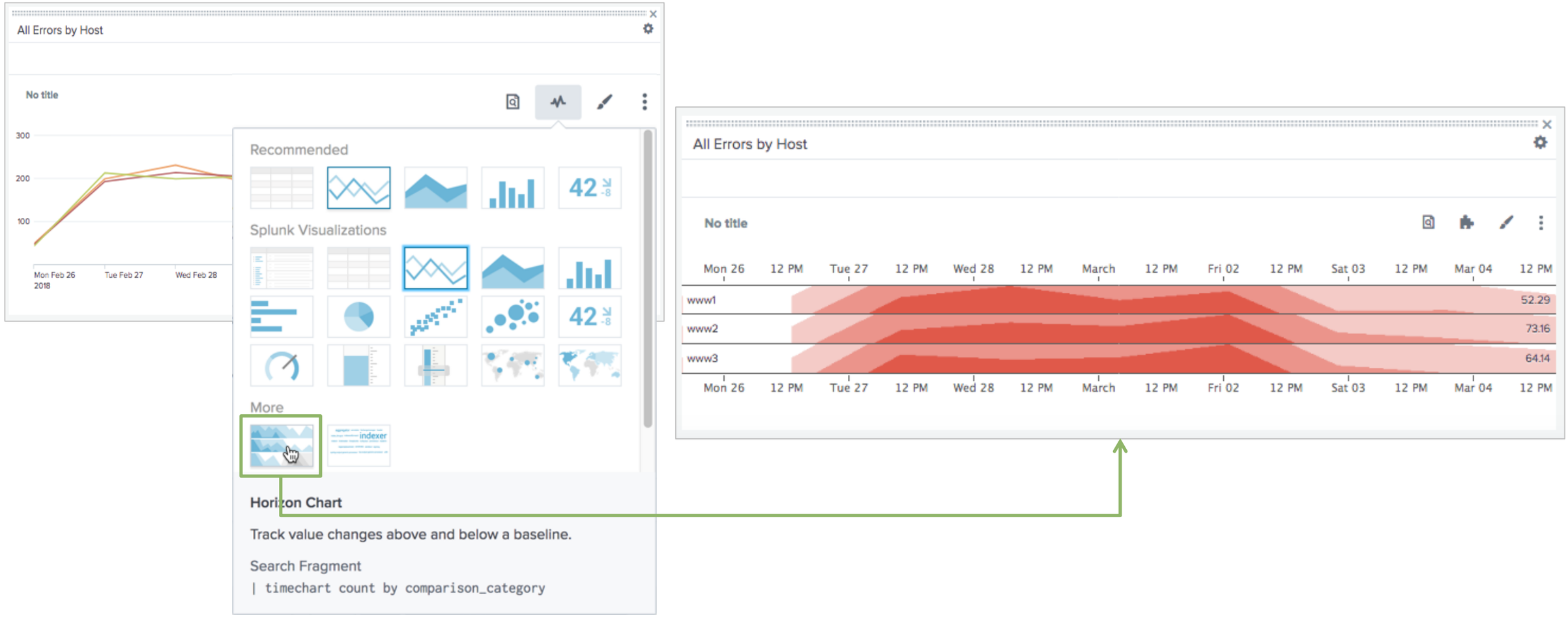
Note

Custom Visualizations are shareable on splunkbase. See: splunkbase.com

Custom Visualizations (cont.)



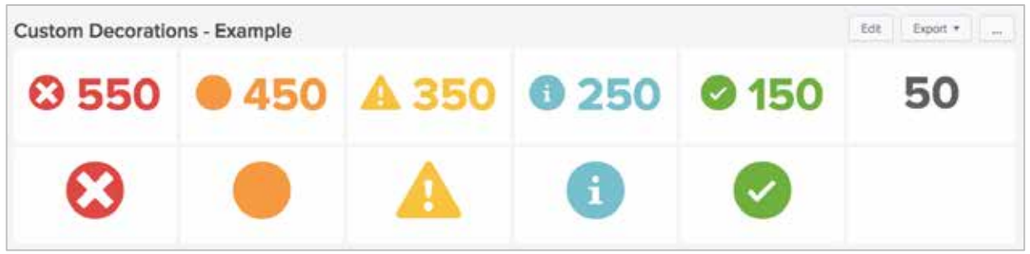
Custom Visualizations – Example

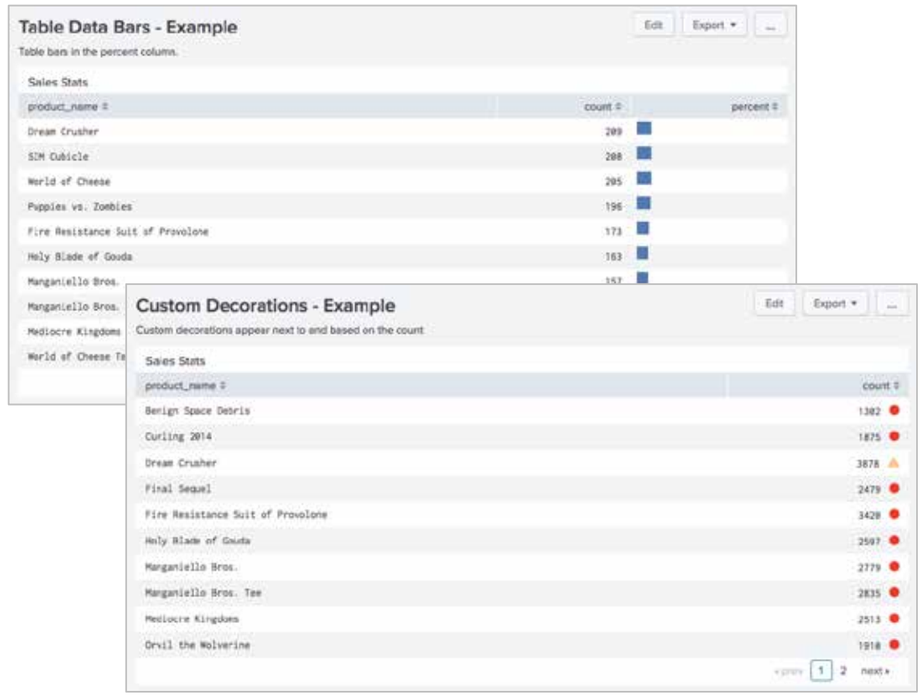


Simple XML Extensions

ADM BUS DEV UX

- Customize a view's look and behavior with your custom CSS and JavaScript
- Examples:
  - **Stylesheet:** use a custom CSS to make simple layout changes
  - **Tables with custom renderers:** use custom styles and behaviors within tables





Simple XML Extensions (cont.)

- Add CSS & JS files to: `$APP_HOME/appserver/static`
- Reference them in the view's XML:

```
<dashboard script="my_script.js" stylesheet="my_style.css">
```

- Reference **multiple** files in the view's XML:

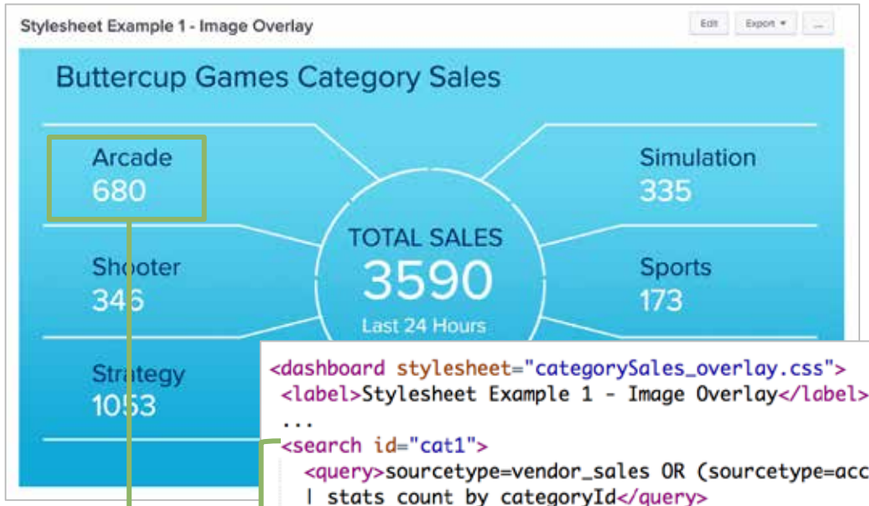
```
<dashboard script="my_script.js, my_script2.js"
 stylesheet="my_style.css, my_style2.css">
```

- Reference **another app's** files in the view's XML:

```
<dashboard script="otherApp:myVizDirectory/my_script.js"
 stylesheet="otherApp:myVizDirectory/my_style.css">
```

- Individual dashboards and forms:
  - Upload the file(s) and reference them in the XML
- All dashboards and forms in an app
  - Upload `dashboard.css` or `dashboard.js` (no XML reference)

Stylesheet Example – Image Overlay



bcg_world.png

XML

```
<dashboard stylesheet="categorySales_overlay.css">
 <label>Stylesheet Example 1 - Image Overlay</label>
 ...
 <search id="cat1">
   <query>sourcetype=vendor_sales OR (sourcetype=access_combined status=200) categoryId="ARCADE"
   | stats count by categoryId</query>
   <earliest>-24h@h</earliest>
   <latest>now</latest>
   <progress>
     <set token="cat1_tok">$result.count$</set>
   </progress>
 </search>
 ...
 <row>
   <panel id="image_overlay_panel">
     <html>
       <div class="bcg_world">
       <div class="image"/>
       <div class="dashboardTitle" id="dashTitle">Buttercup Games Category Sales</div>
       ...
       <div class="singleValueTitle" id="cat1title">Arcade</div>
       <div class="singleValue" id="cat1">$cat1_tok$</div>
       ...
       </div>
     </html>
   </panel>
 </row>
</dashboard>
```

categorySales_overlay.css

```
.dashboard-body {
    background: #FFFFFF !important;
}
#image_overlay_panel .image{
    background: no-repeat url('/static/app/search/bcg_world.png');
    position:absolute;
    top: 0px;
    left: 0px;
    width: inherit;
    height: inherit;
}
.bcg_world {
    position: relative;
    width: 1130px;
    height: 600px;
}

.singleValueTitle {
    font: "Open Sans Light", "Helvetica Neue", sans-serif;
    font-size: 3em;
    color: #003366;
}
.singleValue {
    font-size: 3em;
    color: #ffffff;
}
#image_overlay_panel #cat1title {
   position: absolute;
   top: 140px;
   left: 100px;
}
#image_overlay_panel #cat1 {
   position: absolute;
   top: 186px;
   left: 100px;
}
...
```

JavaScript Example – Table Icons

sales_stats.xml

```
<dashboard script="table_icons_inline.js" stylesheet="table_decorations.css">
  <label>JavaScript - Example 1</label>
  <row>
    <panel>
      <title>Sales Stats</title>
      <table id="table1">
        <search>
          <query>sourcetype=vendor_sales | stats count by product_name</query>
          <earliest>-30d@d</earliest>
```

Loads required libraries and components

Instantiates the icons

Binds the visualization with the search

table_icons_inline.js

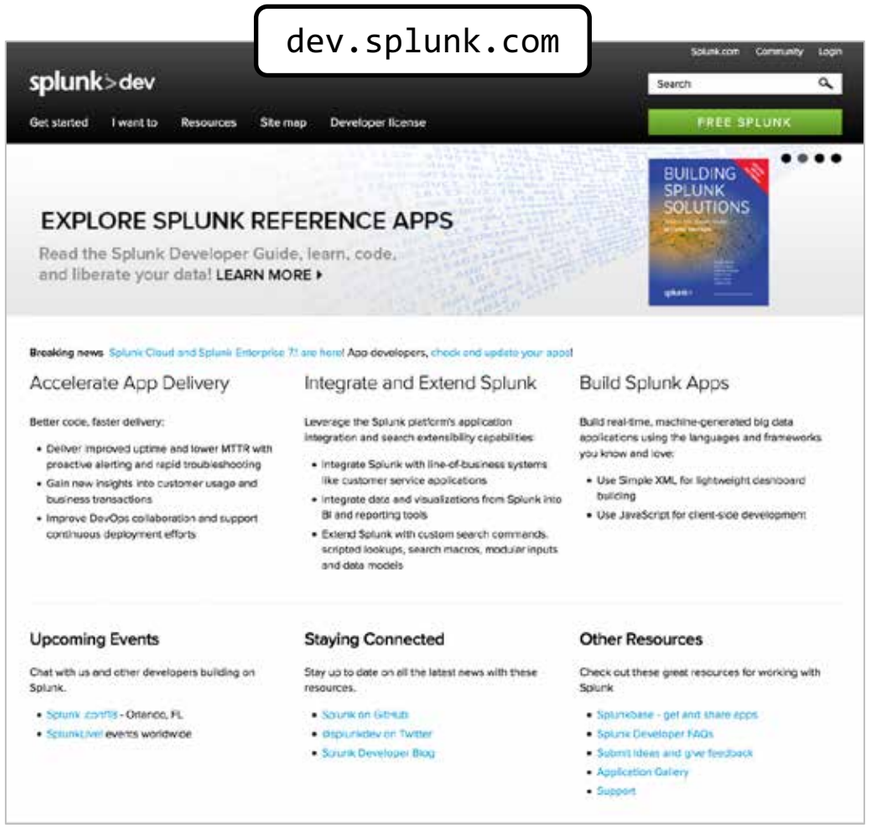
```
require([
 'underscore',
 'jquery',
 'splunkjs/mvc',
 'splunkjs/mvc/tableview',
 'splunkjs/mvc/simplexml/ready!'
], function(_, $, mvc, TableView) {

 var CustomIconRenderer = TableView.BaseCellRenderer.extend({
      canRender: function(cell) {
        return cell.field === 'count';
      },
      render: function($td, cell) {
        var count = cell.value;

        var icon;
          if(count > 5500) {
              icon = 'check';
          } else if(count > 3500) {
              icon = 'alert';
          } else {
              icon = 'alert-circle';
          }

        $td.addClass('icon-inline numeric').html(_.template('<%- text %> <i class="icon-<%-icon%>"></i>', {
              icon: icon,
              text: cell.value
          }));
        }
    });

    mvc.Components.get('table1').getVisualization(function(tableView){
        tableView.table.addCellRenderer(new CustomIconRenderer());
        tableView.table.render();
    });

});
```

table_decorations.css

```
td.icon {
   text-align: center;
}
td.icon i {
   font-size: 25px;
   text-shadow: 1px 1px #aaa;
}
td.icon .severe {
   color: red;
}
td.icon .elevated {
   color: #ffc57a;
}
...
```

Useful References

- Using the Splunk Web Framework
  dev.splunk.com/view/SP-CAAAEXG
- Modify dashboards using Simple XML Extensions
  dev.splunk.com/view/SP-CAAAE4A
- Splunk platform custom visualizations API
  dev.splunk.com/view/SP-CAAAE8Q



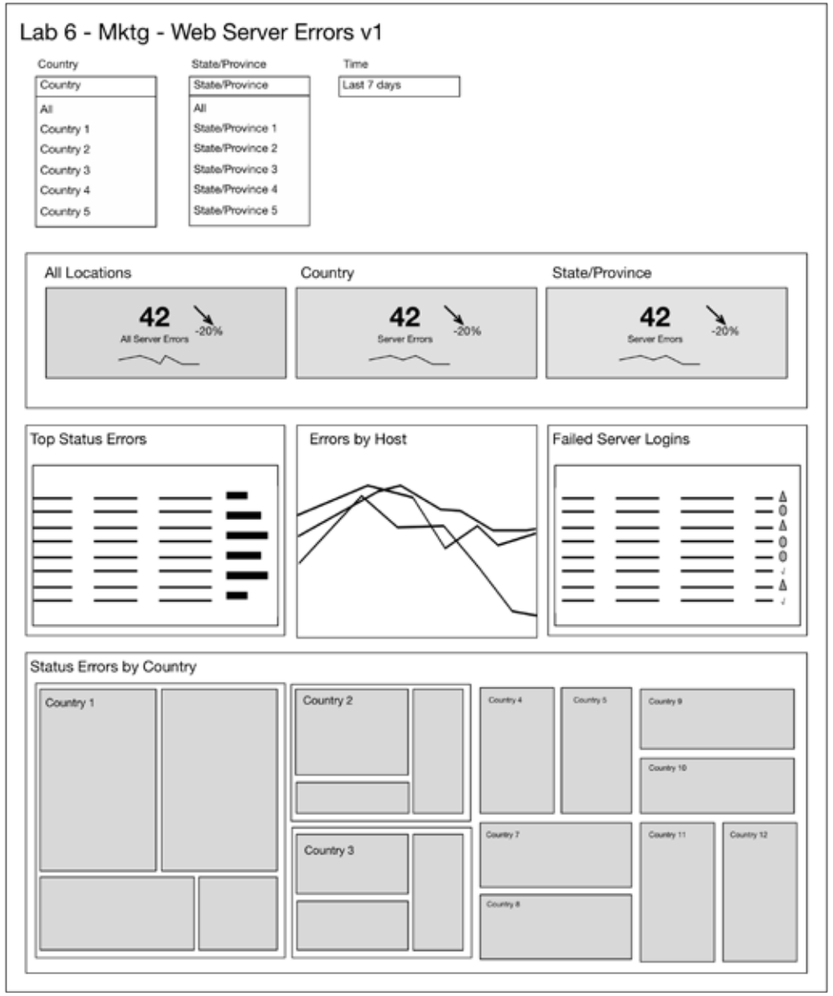
Lab 6 – Add Advanced Behavior & Visualizations

Time: 40 minutes

Scenario: The marketing team would like a form that displays metrics for web store server errors, status errors, and failed server logins.

Tasks:

- Create a dashboard
- Add two cascading inputs and a time input
- Add custom multiselect input behavior
- Add three prebuilt panels and convert to inline
- Add a chart and two table panels
- Display a data bar in a table cell
- Configure a dynamic display panel
- Display icons in a table cell
- Add a Splunk Custom Visualization



Summary

Wrap-up

- You should now be able to:
  - Apply best practices when creating views
  - Capture and access dynamic values
  - Optimize dashboard performance
  - Create well formed, global searches
  - Customize dashboard appearance
  - Create dynamic drilldowns and behaviors
  - Troubleshoot views

Next Steps

Apps & Add-ons

- Splunk Dashboard Examples
- Custom Visualizations

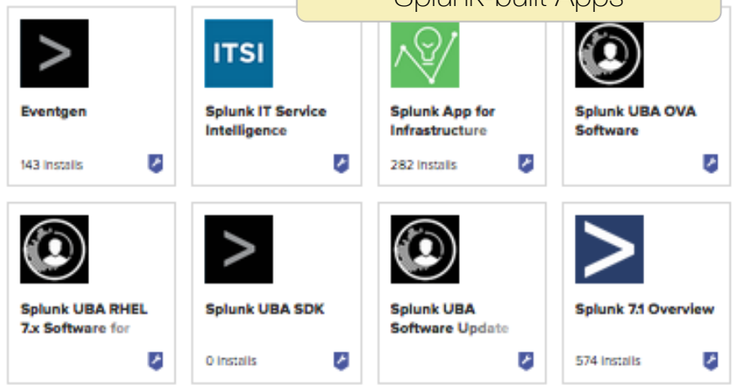
Courses

- Fundamentals 3
- Advanced Dashboards & Visualizations
- Building Splunk Apps
- Developing with Splunk's REST API
- Splunk Data Administration
- Splunk System Administration

Certification Tracks

https://www.splunk.com/en_us/training.html#certificationtrack

Splunk-built Apps



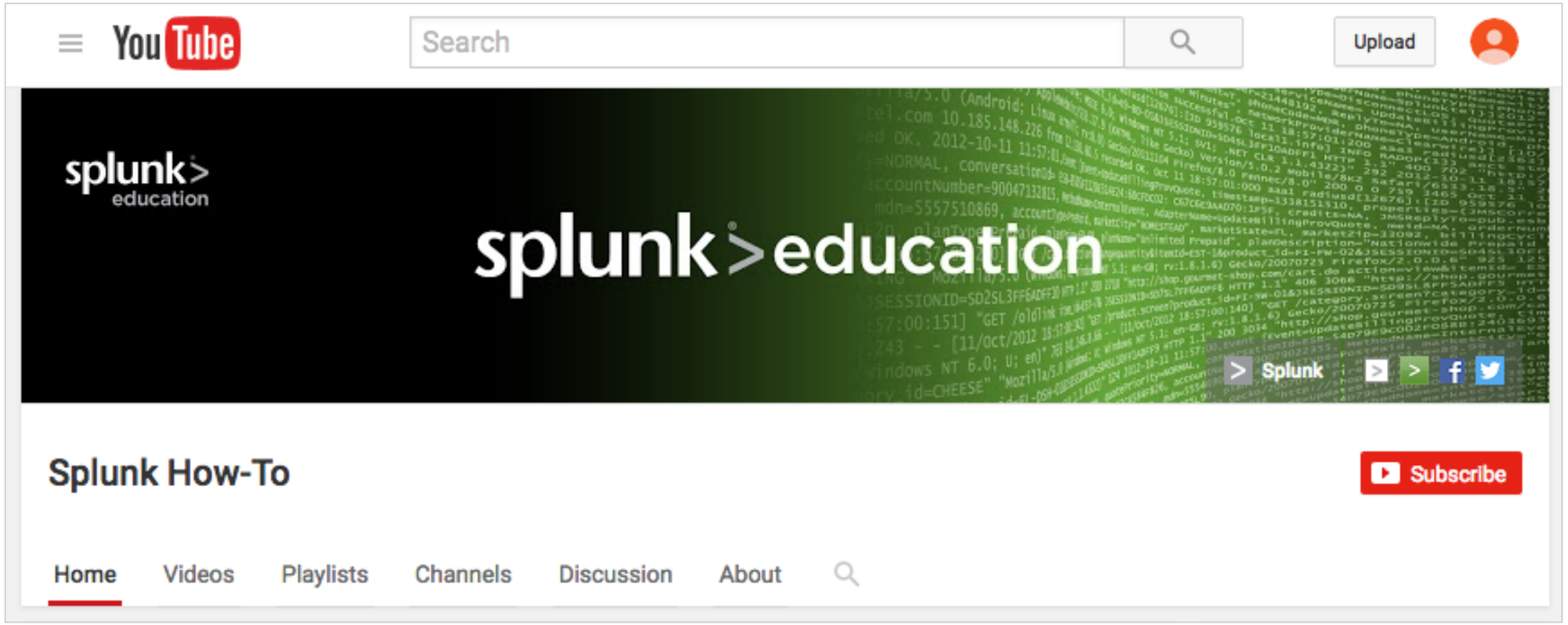
Certification

Community

- Splunk Community Portal
  splunk.com/en_us/community.html
  - Splunk Answers
    answers.splunk.com
  - Splunk Apps
    splunkbase.com
  - Splunk Blogs
    splunk.com/blog/
  - Splunk Live!
    splunklive.splunk.com
  - .conf
    conf.splunk.com
- Slack User Groups
  splk.it/slack
- Splunk Dev Google Group
  groups.google.com/forum/#!forum/splunkdev
- Splunk Docs on Twitter
  twitter.com/splunkdocs
- Splunk Dev on Twitter
  twitter.com/splunkdev
- IRC Channel
  #splunk on the EFNet IRC server

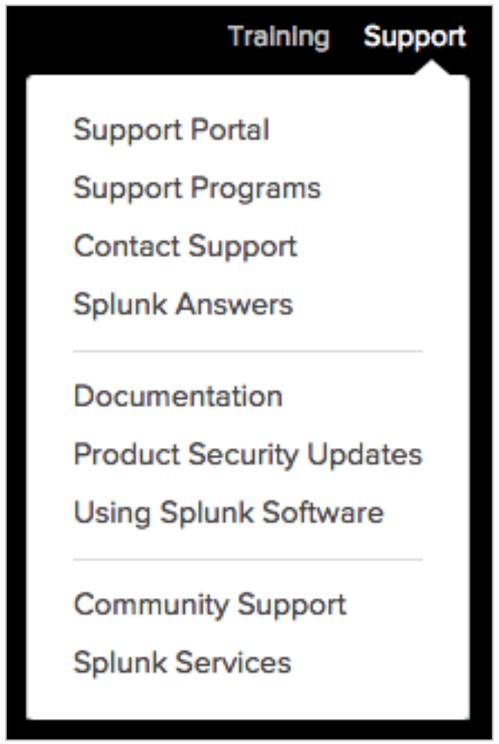
Splunk How-To Channel

- Check out the Splunk Education How-To channel on YouTube: http://splk.it/How-To
- Free, short videos on a variety of Splunk topics



Support Programs

- Web
  - Documentation: dev.splunk.com and docs.splunk.com
  - Wiki: wiki.splunk.com
- Global Support
  Support for critical issues, a dedicated resource to manage your account – 24 x 7 x 365
  - Web: splunk.com/index.php/submit_issue
  - Phone: (855) SPLUNK-S or (855) 775-8657
- Enterprise Support
  - Access customer support by phone and manage your cases online 24 x 7 (depending on support contract)



splunk> .conf19

.conf19
October 21-24, 2019

Splunk University
October 19-21, 2019

Las Vegas, NV
The Venetian Sands Expo

4 Days of Innovation

350 Education Sessions

20 Hours of Networking

sign up for notifications @ conf.splunk.com