Splunk® Enterprise Dashboards and Visualizations 7.3.1

Token usage in dashboards

Generated: 9/19/2019 8:35 am
Token usage in dashboards

Tokens are like programming variables. A token name represents a value that can change, such as a user selection in a form input. You can use tokens to access and pass these values to create more interactive dashboards.

Some tokens are predefined in Splunk software to provide environment, contextual, or user click event information. For example, you can use the $click.value2$ predefined token in a drilldown to access the table cell value that a user clicks.

You can also create custom tokens to implement interactive behavior. For example, define a $show_panel$ token that you can set or unset to control panel display.

To view the most common token commands at a glance, note that you can access the Splunk Dashboards Quick Reference guide by clicking the link in Getting started.

Use cases

You can use tokens to create interactive dashboard behavior in many contexts.

<table>
<thead>
<tr>
<th>Usage context</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search strings</td>
<td>Customize a search string by including tokens to represent dynamic values. When the search runs, it uses the token value. Search event handlers represent search state changes, such as when a search is done, in progress, or cancelled. You can add an event handler to a &lt;search&gt; element and use predefined tokens inside the handler to access search job metadata or first results row data.</td>
</tr>
<tr>
<td>Search event handlers</td>
<td>You can pass token values from a search handler to other dashboard elements to control behavior or content.</td>
</tr>
<tr>
<td>Form inputs</td>
<td>Form inputs use tokens to represent the values that users select or type. When you add an input to a form, a token generates automatically for it. You can customize the token name if needed. When forms have multiple time picker inputs, tokens connect</td>
</tr>
</tbody>
</table>
### Usage context

<table>
<thead>
<tr>
<th>Usage context</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>individual time pickers with one or more visualizations in the form. A user time range selection controls the search time range for the visualization using that token.</td>
<td>When using the <code>map</code> command in a form, use double dollar signs (<code>) to specify a variable string. For example, </code>count``.</td>
</tr>
<tr>
<td>Set tokens on page load</td>
<td>Show customized initial content when a dashboard loads by setting token values in the <code>&lt;init&gt;</code> element.</td>
</tr>
<tr>
<td>Drilldown</td>
<td>Use tokens to configure drilldown behavior. Predefined and custom tokens let you customize content in linked searches, dashboards, or URLs. You can also use tokens to create interactive behavior in the same dashboard.</td>
</tr>
<tr>
<td>Define chart pan and zoom areas</td>
<td>Use predefined tokens to define pan and zoom areas for charts.</td>
</tr>
</tbody>
</table>

### Using tokens in a search

Tokens capture and pass values in a dashboard. Token values can come from various sources, including form inputs and predefined token values for visualizations. Searches can access token values.

In a search, token name syntax uses `$...$` delimiters. For example, if you define a form input token as `field_tok`, you can specify the token in a search as `$field_tok$`. Here is an example.

```
<search>
index=_internal source=*splunkd.log | stats count by $field_tok$
</search>
```

See Token filters for advanced syntax to access token values.

### Define search tokens

You can set search tokens for a dashboard to display search job metadata or to control dashboard behavior.

There are many ways to use search tokens. Here are some example use cases.
• Including a search result count in a visualization title.
• If a search returns no results, run a different search or hide the panel.
• Hide or show panels if a search fails.

There are also various advanced options for working with search tokens. Options include the following:

• Show the time range of the search below the visualization element using HTML.
• Build a custom HTML element and insert search results as tokens.
• Define token values based on the result of token-specific eval expressions.

**Search event elements and job properties**

There are several search event handlers that you can use in Simple XML dashboards.

<table>
<thead>
<tr>
<th>Handler name</th>
<th>Access to search job properties?</th>
<th>Access to first results row?</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;progress&gt;</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt;done&gt;</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt;cancelled&gt;</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>&lt;error&gt;</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>&lt;fail&gt;</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Within a search event handler, you can access specific job properties with tokens. For example, here are some commonly used job metadata tokens.

• $job.earliestTime$: Initial job start time.

• $job.latestTime$: Latest time recorded for the search job.

• $job.resultCount$: Number of results a search job returned.

• $job.runDuration$: Time, in seconds, for the search to complete.

• $job.messages$: List of error and/or debug messages generated by the search job.
To learn about more search job properties, see View search job properties in the Search Manual.

Search tokens for dynamic display example

Here is an example of the <search> element for a dashboard that hides a panel if no search results are returned.

```xml
<search id="search_logic">
  <query>$index_switcher$ | top sourcetype</query>
  <earliest>-60m@m</earliest>
  <latest>now</latest>

  <progress>
    <!-- match attribute for condition uses eval-like expression (see Splunk search language 'eval' command) -->
    <!-- logic: if resultCount is 0, then show a static html element, and hide the chart element -->
    <condition match="'job.resultCount' == 0">
      <set token="show_html">true</set>
    </condition>
    <condition>
      <unset token="show_html"/>
    </condition>
  </progress>
</search>
```

For more examples, see the Splunk Dashboard Examples App.

Custom logic for dashboards

Add custom logic to a dashboard with the <condition match=" "> and <eval> elements.

For both <condition> and <eval> elements, all data available from an event as well as the submitted token model is available as a variable within the eval expression.

Token syntax

As of software version 6.4, you can use either $...$ delimiters or single quote delimiters for tokens in an <eval> or <condition match=" "> statement. For example, both of the following options are valid.

- <condition match="'$job.resultCount' > 0">  
- <condition match="'$job.resultCount' > 0">  

4
Define conditional matching

Use the `<condition match=" "> element to define conditional behavior. The following example controls a token value according to the result count job property.

```
<condition match=" $job.resultCount$ == 0">
    <set token="show_table_query">true</set>
</condition>
```

You can also use a dashboard `eval` expression to define a condition to match. Here is an example using `<condition match=" "> to set a token value depending on whether the selected time range spans more than one day.

```
< condition match="relative_time(now(), earliest) - relative_time(now(), latest) > 86400">
    <!-- Selected time range spans more than a day, use summary search -->
    <set token="table_query">index=my_summary_index | timechart count</set>
</condition>
```

Troubleshoot job property access

Job properties are not available throughout a dashboard by default. If conditional logic statements in your source code are not being applied as expected to a job property, add a search event handler to access the job property in a custom token. Use the custom token in the conditional logic statement instead.

In this example, the `$search_results$` token gets the `$job.resultCount$` job property value in a `<search>` event handler. The conditional logic statement evaluates the `$search_results$` token and sets the `$show_panel$` token accordingly.

```
[...]
<search>
    <query>index=_internal </query>
    <earliest>-24h@h</earliest>
    <latest>now</latest>
    <done>
        <set token="search_results">$job.resultCount$</set>
    </done>
</search>
<drilldown>
    <condition match=" $search_results$ != 0">
```

5
Using strings in a conditional statement
If you are using a `<condition match= " " >` statement to evaluate a string value, such as a sourcetype name from the first results row, put escaped quotation marks around the string value. This prevents the dashboard parser from handling the quotation marks as special characters.

The following example sets up conditional token setting that depends on the sourcetype field value in the first results row. If the sourcetype field value in the first results row is `mongod`, the "show_table" token is set to `true`.

To specify the "mongod" string in the conditional match statement, replace the quotation marks with the equivalent HTML character entities.

For more information about using special characters in dashboard source code, see Editing Simple XML.

Define token filtering and formatting
You can use `eval` expression logic to define token filtering and formatting. For example, you can set a token value to the result of an `eval` expression.

Dashboard `eval` expression functionality
The dashboard `eval` expression has the same syntax and semantics as the `eval` expression syntax for SPL queries. Most of the same `eval` expression functionality is the same between the dashboard `eval` expression and the SPL version of `eval`. However, there are some important exceptions.

Unavailable dashboard `eval` expression functions

- `commands(X)`
- `searchmatch(X)`
- `exact(X)`
- Cryptographic hash functions:
**Eval expression functions with different behavior for dashboards**

- `relative_time(X,Y)`: Uses client time zone.
- `strftime(X,Y)`: Uses client time zone.
- `strptime(X,Y)`: Uses client time zone.

It is also important to note that regular expressions in dashboard eval expressions use the syntax and semantics of the JavaScript regular expression engine. This is not the same engine used for SPL eval expressions. If you are using regular expressions in search tokens, check that syntax and semantics match those for JavaScript.

To learn more about eval expression functions, see eval in the Search Reference.

**Custom logic examples**

You can use an eval expression in `<condition>` event handler elements. Here is an example.

```html
<condition match="[eval expression]">
  . . . [conditional actions] . . .
</condition>
```

You can also compute a token's value based on the result of an eval expression. Here is an example.

```html
<eval token="new_token">[eval expression]</eval>
```

**Define tokens for form inputs**

All form inputs have a token attribute that defines a token for the user-selected value for the input. Form inputs also have child `<prefix>` and `<suffix>` elements that further modify the value of the token. For multiselect options, there are additional elements that can modify the value of the token. See Define tokens for...
multiselect inputs.

This code snippet defines a token for a drop-down list. The selected choice for the dropdown provides the value of the token.

```html
<input type="dropdown" token="sourcetype_tok">
<label>Select a source type</label>
<default>splunkd</default>
<choice value="splunkd">splunkd</choice>
<choice value="splunk_web_access">splunk_web_access</choice>
<choice value="splunkd_ui_access">splunkd_ui_access</choice>
</input>
```

See Form input example.

**Define tokens for multiselect inputs**

A multiselect input uses the `<prefix>`, `<suffix>`, `<valuePrefix>`, `<valueSuffix>`, and `<delimiter>` elements to build the multiselection search for the selected choices. The multiselection search, which is the value of the token for the input, ensures that the input passes all selected values to the search for the form.

The following code snippet shows how to build a value for the multiselect token. If a user selects both `splunkd` and `splunk_web_access` from the multiselect input, the token value is the following search fragment:

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

The search fragment derives from:

```html
<prefix> + <valuePrefix> + [choice value] + <valueSuffix> + <delimiter>
    . . . + <suffix>
    (    sourcetype ="    splunkd"
        _OR_
)
```

```html
<input type="multiselect" token="sourcetype_tok">
<label>Select one or more source types</label>
<choice value="splunk_web_access">splunk_web_access</choice>
<choice value="splunkd">splunkd</choice>
<choice value="splunk_ui_access">splunk_ui_access</choice>
<choice value="splunkd_access">splunkd_access</choice>

<!-- Build multi-selection search: 
    (sourcetype ="value1" OR sourcetype ="value2" OR ...) 
-->
```
Define tokens for time inputs

If you have a form with panels that use different time pickers, use tokens for the time input to indicate the time picker to use for each panel. To access the earliest and latest values from a time picker, use the following modifiers to the token:

- $timer_tok.earliest$
- $timer_tok.latest$

A time input that does not define a token is global. The values selected from such a time picker applies to all visualizations that do not otherwise specify a time picker.

See Time input example.

Define tokens for conditional operations with form inputs

You can define tokens for conditional operations for form inputs. The value of the token changes according to the condition you specify. For example, you can modify searches or select different visualizations to display based on the conditional value of a token.

Conditional operations include:

- Modify searches based on the token value.
- Hide or display panels and the content of panels based on a condition.
- Select a view to open based on a token value.

Conditional operations are available for form inputs and dynamic drilldown. Form inputs use various combinations of the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Element Description

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;change&gt;</code></td>
<td>Container element for the conditions that you define. Sets the condition based on the value of the input selection. In the Conditional input example, this is the value of the selected choice for the drop-down list.</td>
</tr>
<tr>
<td><code>&lt;condition&gt;</code></td>
<td>Conditional input example, this is the value of the selected choice for the drop-down list.</td>
</tr>
<tr>
<td><code>&lt;link&gt;</code></td>
<td>Specify a link to a destination based on a condition. Defines various values for a token. The <code>&lt;search&gt;</code> element in the dashboard consumes the value of this token.</td>
</tr>
<tr>
<td><code>&lt;set&gt;</code></td>
<td>In the Conditional input example, defines value for the token <code>earliest_tok</code>. Removes a token that was previously set.</td>
</tr>
<tr>
<td><code>&lt;unset&gt;</code></td>
<td>This is useful for conditional operations that depend on a token being set. See the example at Conditional operations with form inputs.</td>
</tr>
</tbody>
</table>

#### Predefined tokens for accessing labels and values of form inputs

Splunk Enterprise provides predefined tokens to access the label and value of form inputs. Tokens are available for the following inputs:

- check box
- drop-down list
- multiselect
- radio buttons

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>Contains the specified name of a form input choice.</td>
</tr>
<tr>
<td>value</td>
<td>Contains the value of a form input choice. These tokens are useful to customize a search or place the label of the selected choice in a title or description of a panel or visualization.</td>
</tr>
</tbody>
</table>

See Access labels and values of form inputs.

### Set tokens on page load

Add an `<init>` element to a dashboard or form to reuse content or create a template. The token values inside this element are set when the dashboard page...
loads.

**Guidelines**

Within a `<dashboard>` or `<form>` element, place content to set on page load inside the following tags.

```html
<init>
</init>
```

- You can use the following event handlers to specify token settings within the `<init>` tags.
  - `<condition>`
  - `<eval>`
  - `<link>`
  - `<set>`
  - `<unset>`

- PDF scheduling is disabled for dashboards and forms that include an `<init>` element.

- Token settings made within the `<init>` element override any settings made in URL query string parameters.

- Token setting on page load is only supported for Simple XML dashboards. If you convert a dashboard to HTML, token settings within the `<init>` element are disabled.

**Example**

This form sets an app name token on page load. The token value is used in a panel label and a search with the `|s$` filter to wrap the value in quotation marks.

```html
<form>
  <label>Application Monitoring: Exchange</label>
  <init>
    <set token="app_name">my_app_name</set>
  </init>
  <row>
    <panel>
      <title>Activity Monitoring: $app_name$</title>
    </panel>
  </row>
</form>
```
Use global tokens to access environment information

Access details about the user, Splunk platform instance, and environment using global tokens. The following tokens are available.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$env:user$</td>
<td>Current user’s user name</td>
</tr>
<tr>
<td>$env:user_realname$</td>
<td>Current user full name.</td>
</tr>
<tr>
<td>$env:user_email$</td>
<td>Current user email address.</td>
</tr>
<tr>
<td>$env:app$</td>
<td>Current app context</td>
</tr>
<tr>
<td>$env:locale$</td>
<td>Current locale</td>
</tr>
<tr>
<td>$env:page$</td>
<td>Currently open page</td>
</tr>
<tr>
<td>$env:product$</td>
<td>Current instance product type</td>
</tr>
<tr>
<td>$env:instance_type$</td>
<td>Indicates whether the current instance is Splunk Cloud or an on-premises deployment</td>
</tr>
<tr>
<td>$env:is_cloud$</td>
<td>Indicates if the current instance is Splunk Cloud. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_enterprise$</td>
<td>Indicates if the current instance is a Splunk Enterprise deployment. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_hunk$</td>
<td>Indicates if the current instance is a Hunk deployment. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_lite$</td>
<td>Indicates if the current instance is a Splunk Light deployment. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_lite_free$</td>
<td>Indicates if the current instance is using a Splunk Light free license. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:is_free$</td>
<td>Indicates if the current instance is using a Splunk Enterprise free license. This token is only set when &quot;true&quot;.</td>
</tr>
<tr>
<td>$env:version$</td>
<td>Current instance product version</td>
</tr>
</tbody>
</table>
Define tokens for dynamic drilldown

**Predefined tokens for dynamic drilldown**

Splunk Enterprise provides predefined tokens for dynamic drilldown. The predefined tokens capture values according to the location a user clicks in a visualization.

Predefined token availability and captured values vary according to visualization type. See Predefined drilldown tokens section of the Simple XML Reference for a complete list of all predefined tokens.

**Define tokens for conditional operations with the `<drilldown>` element**

You can use tokens for conditional drilldown behavior such as:

- Set token values, based on a condition.
- Select a value for multivalue fields in a visualization.
  A multivalue field is a field that appears more than once with different values.
- Select a view to open based on a token value.
- Hide or show panels based on conditions.

Conditional operations are available for both form inputs and conditional drilldown. Defining tokens for conditional drilldown uses various combinations of the following tags:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;drilldown&gt;</code></td>
<td>Define link destinations for fields in a dashboard or form. You can also use with <code>&lt;condition&gt;</code> to set tokens for custom actions.</td>
</tr>
<tr>
<td><code>&lt;condition&gt;</code></td>
<td>Limit the scope of drilldown actions to specific fields.</td>
</tr>
<tr>
<td><code>&lt;selection&gt;</code></td>
<td>Use with the <code>&lt;set&gt;</code> element to set the time window for the pan and zoom features of charts. Applies to charts of type area, column, or line.</td>
</tr>
</tbody>
</table>

See Chart controls and the `<selection>` entry in the Simple XML Reference.
Use the `<set>` element to define tokens

Use the `<set>` element to define tokens for conditional use. You can use the value of another token when defining a token with the `<set>` element. For example, the following code snippet defines the `sourcetype_tok` token. This token captures the value clicked from a `<table>` element for the field `sourcetype`.

```
<drilldown>
  <condition field="sourcetype">
    <set token="sourcetype_tok">$click.value2$</set>
  </condition>
</drilldown>
```

You can use the `sourcetype_tok` token in a search:

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

Use the `<condition>` element to select a value for multivalue fields in a visualization

Multivalue fields are fields that appear multiple times in an event and have a different value for each appearance. See Configure multivalue fields in the Knowledge Manager manual.

If you have a dashboard that displays multivalue fields, use the `<condition>` element to specify a drilldown location specific to the value of a clicked field. The following example links to different destinations based on the specific value for the field. The `<link>` element consumes different predefined tokens for each condition.

```
<drilldown>
  <condition field="badges">
    <link>
      /app/foursquare_vegas/vegas_badge_1?form.badge=$click.value2$
    </link>
  </condition>
</drilldown>
```
Syntax to consume tokens

Use $...$ delimiters to access the value of a token. For example, the following search for a visualization accesses the field_tok token. A form input previously defined the field_tok token:

```
index=_internal source=*splunkd.log | stats count by $field_tok$
```

**Token filters**

Token filters ensure that you correctly capture the value of a token.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrap value in quotes</td>
<td>Ensures that quotation marks surround the value referenced by the token. Escapes all quotation characters, &quot;&quot;, within the quoted value.</td>
</tr>
<tr>
<td>$token_name</td>
<td>s$</td>
</tr>
<tr>
<td>HTML format</td>
<td>Ensures that the token value is valid for HTML formatting.</td>
</tr>
<tr>
<td>$token_name</td>
<td>h$</td>
</tr>
<tr>
<td>URL format</td>
<td>Ensures that the token value is valid to use as a URL.</td>
</tr>
<tr>
<td>$token_name</td>
<td>u$</td>
</tr>
<tr>
<td>Specify no character escaping</td>
<td>Prevents the default token filter from running. No characters in the token are escaped.</td>
</tr>
<tr>
<td>$token_name</td>
<td>n$</td>
</tr>
</tbody>
</table>

The following code snippet uses the |s filter to place quotation marks around the value returned from a token:
<search>
  <query>
    index=_internal sourcetype=$sourcetype_tok|s$ | timechart count by sourcetype
  </query>
</search>

If the value of sourcetype_tok is access_combined, it builds the following search string:

index=_internal sourcetype="access_combined" | timechart count by sourcetype

**Escape the $ token delimiter character**

If you include static text that contains the $ character, use $$ to escape the token delimiter value.

**Combine literal values with token values**

You can combine literal values with the value returned from a token. Use with the <set> element to set conditional actions based on token values.

The following template combines the captured value from the predefined token, click.value, with static text. It places the value of NewToken in quotation marks.

<set token="NewToken">sourcetype=$click.value|s$</set>

If the value of click.value is access_combined, then the value of NewToken is the following search fragment:

sourcetype="access_combined"

You can use the prefix and suffix attributes to the <set> element to specify static text for a token value. The following example sets the value for NewToken. It is equivalent to the template example:

<set token="NewToken" prefix="sourcetype="quot;" suffix="quot;">$click.value$</set>
**Access tokens to show or hide user interface components**

You can use token values to conditionally show or hide user interface components. The following elements contain the attributes `depends` and `rejects`. Use the `<set>` and `<unset>` elements to set the token values that these attributes consume.

- `<row>`
- `<panel>`
- `<chart>`
- `<event>`
- `<html>`
- `<map>`
- `<single>`
- `<table>`
- `<input>`

For example, show the `<chart>` element only when the `showChart` token has been set.

```xml
<chart depends="$showChart$">
  <!-- chart content here -->
</chart>
```

**Define tokens for pan and zoom chart controls**

Splunk Enterprise uses predefined tokens to implement the zoom feature on a chart. Using the zoom feature, you can select a portion of a data series in a chart that opens in a separate chart. See Pan and zoom chart controls.

Set the values of the predefined tokens within a `<selection>` element that is a child element of a chart. Use the token values in the original chart to display a new chart that zooms to the selection.

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>start</code></td>
<td>Captures the value of the x-axis at the beginning and end of a selection in a chart.</td>
</tr>
<tr>
<td><code>end</code></td>
<td>Valid only in the context of the chart. Assign the values to tokens that you define to access the values elsewhere in a dashboard.</td>
</tr>
<tr>
<td><code>start.&lt;field&gt;</code></td>
<td>Captures the values for the y-axis values for a selection. <code>&lt;field&gt;</code> represents a series displayed in the chart.</td>
</tr>
<tr>
<td><code>end.&lt;field&gt;</code></td>
<td>Valid only in the context of the chart. Assign the values to tokens that you define to access the values elsewhere in a dashboard.</td>
</tr>
</tbody>
</table>
See Pan and zoom chart controls for an example that shows how zoom to a selection in a time chart.

**Tokens with SplunkJS Stack**

If you are using SplunkJS Stack with JavaScript extensions, see Tokens and Data Binding on the Splunk Developer Portal to learn how to use tokens with JavaScript.

**Examples of token usage**

**Form input example**

This example shows the basic usage of tokens in form inputs. It uses a drop-down list to select the source type for the time chart. See Define tokens for form inputs.

The `<input>` element defines the `sourcetype_tok` that is consumed by the search for the visualization.
Form example: source type time chart

Multiselect input example

This example shows how to build a search string for a form input using static text and token values. This is useful for building multiselect options. See Define tokens for multiselect inputs.

The example uses the <prefix>, <suffix>, <valuePrefix>, <valueSuffix>, and <delimiter> elements to build the multiselect search string. When a user selects splunkd and splunk_web_access, it generates the following search string:

(sourcetype ="splunkd" OR sourcetype ="splunk_web_access")
<form>
  <label>Form with multiselect</label>
  <fieldset autoRun="false" submitButton="true">
    <html>
      <strong>Multiselect choices</strong>
    </html>
    <input type="multiselect" token="sourcetype_tok"
           searchWhenChanged="false">
    <label>Select one or more source types</label>
    <choice value="*">All</choice>
    <choice value="splunk_web_access">splunk_web_access</choice>
    <choice value="splunkd">splunkd</choice>
    <choice value="splunk_ui_access">splunk_ui_access</choice>
    <choice value="splunkd_access">splunkd_access</choice>
    <!-- Build multiselect search: -->
    <prefix>(</prefix>
    <valuePrefix>sourcetype ="</valuePrefix>
    <valueSuffix"> OR </valueSuffix>
    <delimiter> OR </delimiter>
    <suffix>)</suffix>
  </input>
  </fieldset>
  <row>
    <panel>
      <title></title>
      <chart>
        <search>
          <query>index =_internal $sourcetype_tok$ | stats count by sourcetype</query>
          <earliest>-24h</earliest>
          <latest>now</latest>
        </search>
        <option name="charting.chart">line</option>
        <option name="charting.axisY.scale">log</option>
      </chart>
    </panel>
  </row>
</form>
**Time input example**

This example shows how to use both a global and local time picker in a form. It also shows how to access the predefined modifiers to a time input token. See Define tokens for time inputs.

The example shows a form with both a global time picker and local time picker. The `<chart>` element contains the local time picker and uses modifiers to the `local_time_input_tok` token to access the earliest and latest values.
Conditional operations with form inputs

This example shows how to use conditional operations with form inputs. See Define tokens for conditional operations with form inputs.

The example uses the <change>, <condition>, and <set> elements to conditionally set the label for the selected time and to set the earliest time token. The search consumes the earliest time token to set the bounds for the search. This example uses the label and value predefined tokens for input choices. See Predefined tokens for accessing labels and values of form inputs.
Note: All input elements, with the exception of the time input, require a token attribute to be present. In the example, the input element defines the token, period_tok. However, this token is never consumed by the search.

```xml
<form>
  <label>Use tokens with conditional input choices</label>
  <fieldset submitButton="false">
    <input type="radio" token="period_tok">
      <label>Select a time range</label>
      <choice value="-24h@h">Last 24 Hours</choice>
      <choice value="-7d@h">Last 7 Days</choice>
      <choice value="-30d@h">Last 30 Days</choice>
      <default>Last 24 Hours</default>

    <!-- set condition based on the label defined by <choice> -->
    <!-- Within each condition, specify a custom label for display -->
    <!-- Capture the selected value in the token, earliest_tok -->
    <change>
      <condition label="Last 24 Hours">
        <set token="date_label">Yesterday</set>
        <set token="earliest_tok">$value$</set>
      </condition>
      <condition label="Last 7 Days">
        <set token="date_label">Last week</set>
        <set token="earliest_tok">$value$</set>
      </condition>
      <condition label="Last 30 Days">
        <set token="date_label">Last month</set>
        <set token="earliest_tok">$value$</set>
      </condition>
    </change>
  </input>
</fieldset>
</form>
```

Conditional Inputs

```xml
<row>
  <panel>
    <title>Conditional Inputs</title>
    <chart>
```
Access labels and values of form inputs

This example shows how to use tokens to access the labels and values of form inputs. See Predefined tokens for accessing labels and values of form inputs.

The example uses the label of the selected radio button in the title of the visualization. It uses the value of the selected radio button to determine the bounds of the search.
Use tokens with input choices to capture input labels and values.

Select a time range:
- Last 24 Hours
- Last 7 Days
- Last 30 Days

Default: Last 24 Hours

Conditional Inputs

Display selected label in the title

Source Type by $date_label$

Use the value of $earliest_tok$

To set the time range

Latest: now

Options:
- Chart: log
- Time period: Events