Session 3-4: Building an Operational Intelligence Application

Table of Contents

Session 3-4: Building an Operational Intelligence Application

In this session, we will learn how to build and modify a Splunk application. We will cover the following topics:

- Creating an Operational Intelligence application
- Adding dashboards and reports
- Organizing the dashboards more efficiently
- Dynamically drilling down on activity reports
- Creating a form to search web activity
- Linking web page activity reports to the form
- Displaying a geographical map of visitors
- Scheduling PDF delivery of a dashboard

Introduction

In the previous session, we were introduced to Splunk's awesome dashboarding and visualization capabilities. We created several basic dashboards and populated them with various Operational Intelligence-driven visualizations. In this session, we will continue to build on what we have learned in the previous sessions and further advance our Splunk dashboarding knowledge. You will learn how to create a Splunk application and populate it with several dashboards. You will also learn to use some of Splunk's more advanced dashboarding capabilities such as forms, drill downs, and maps.

Splunk applications are best thought of as workspaces designed specifically around certain use cases. In this session, we will be building a new application that focuses specifically on Operational Intelligence. Splunk apps can vary in complexity from a series of saved reports and dashboards through to complex, fully-featured standalone solutions. After logging in to Splunk for the first time, you actually interface with Splunk through the launcher application, which displays a dashboard that lists the other applications installed on the system. The Search and Reporting application that we have been using in this book so far is an example of another bundled Splunk application.

TIP

Several vendors, developers, and customers have developed applications that can be used to get you started with your datasets. Most of these applications are available for free download from the Splunk app store at http://splunkbase.splunk.com.

In this session, you will also start to get to grips with the dashboard forms functionality. The best way to think about forms in Splunk is that they are essentially dashboards with an interface, allowing the users to easily supply values to the underlying dashboard searches. For example, a basic form in Splunk would be a dashboard with a user-selectable time range at the top. The user might then select to run the dashboard over the last 24 hours, and all the searches that power the dashboard visualizations will run over this selected time range.

Forms, by their very nature, require input. Luckily for us, Splunk has a number of common form inputs out-of-the-box that can be readily used by using the dashboard editor or `SimpleXML`. The available form inputs and an explanation of their common usages are detailed in the following table:

Input	**Common usage**
`Dropdown`	This is used to display the lists of user selectable values. Drop-downs can be populated dynamically using Splunk searches and even filtered based on the user selection of another drop-down. The users can also select single values or multiple values.
`Radio`	This is used for simple yes/no or single selection type values. You can only select one value at a time with the radio buttons, unlike `dropdown`.
`Text`	This is a simple textbox allowing the user to type whatever value they want to search for. The textbox is great for searching wildcard type values, such as field values of `abc*`.

Input	Common usage
`Time`	This is a time range picker. This is exactly the same as the time range picker found on the main Splunk search dashboard. You can add a time range for the entire dashboard or for individual dashboard panels.

Dashboards in Splunk are coded behind the scenes in something known as `SimpleXML`. This `SimpleXML` code can be edited directly or by use of Splunk's interactive, GUI-based dashboard editor. For the most part, this session will focus on using the GUI-based dashboard editor, which allows dashboards to be edited without touching a line of code—nice! However, you will be introduced to direct `SimpleXML` editing in order to take advantage of more advanced capabilities and options.

Ok, enough of discussion; let's get started!

Creating an Operational Intelligence application

This recipe will show you how to create an empty Splunk app that we will use as the starting point for building our Operational Intelligence application.

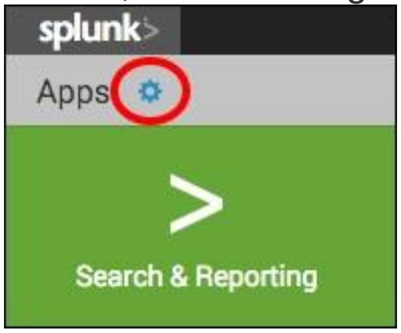
Getting ready

To go through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*. You should have also completed the recipes from the earlier sessions. You should be familiar with navigating the Splunk user interface.

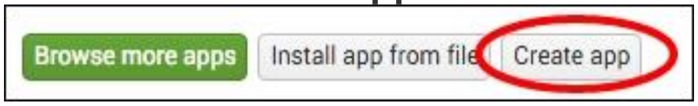
How to do it…

Follow the given steps to create the Operational Intelligence application:

1. Log in to your Splunk server.
2. From the **Apps** menu in the upper left-hand corner of the home screen, click on the gear icon.

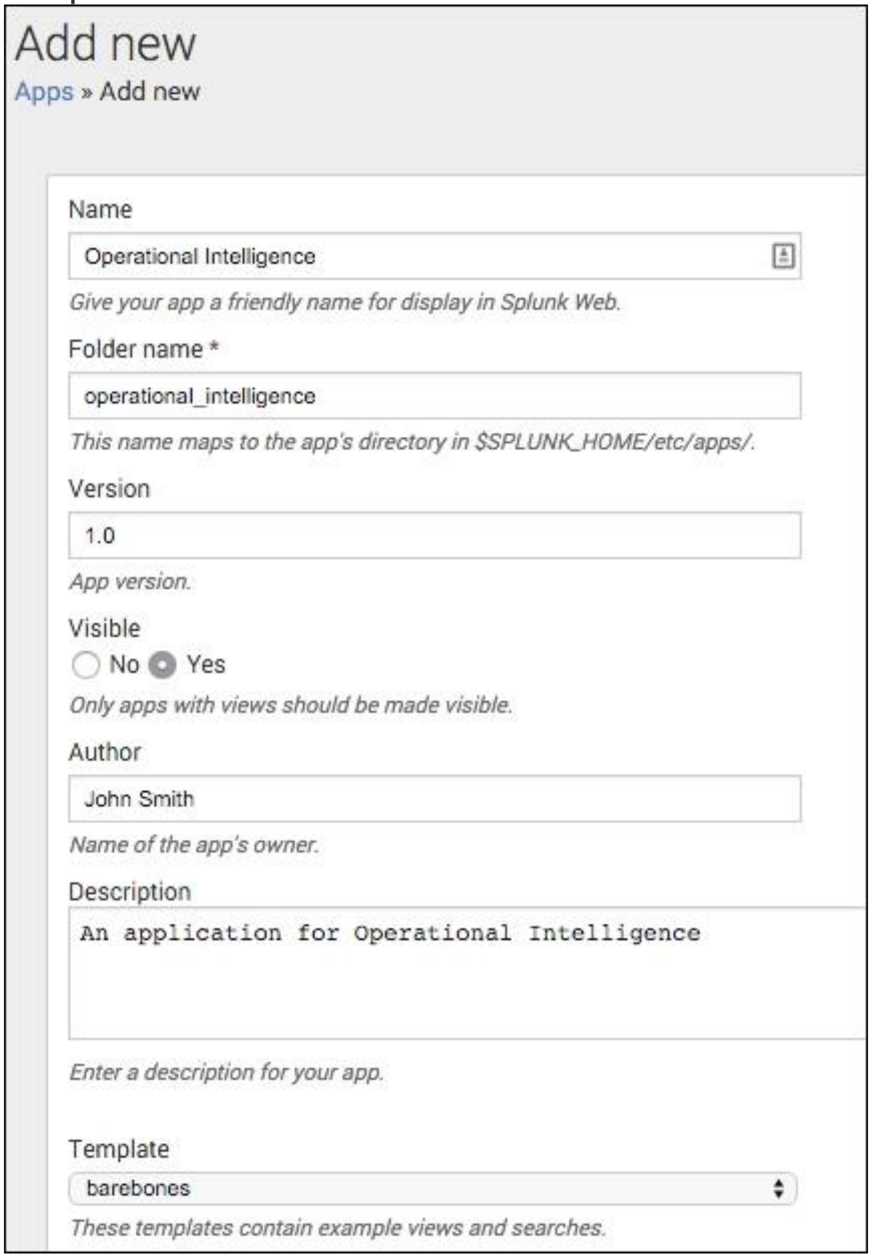


3. Click on the **Create app** button.



4. Complete the fields in the box that follows. Name the app `Operational Intelligence` and give it a folder name

of `operational_intelligence`. Add a version number and provide an author name. Ensure that **Visible** is set to **Yes** and the barebones template is selected.



Add new

Apps » Add new

Name

Operational Intelligence

Give your app a friendly name for display in Splunk Web.

Folder name *

operational_intelligence

This name maps to the app's directory in $SPLUNK_HOME/etc/apps/.

Version

1.0

App version.

Visible

No [✓ Yes]

Only apps with views should be made visible.

Author

John Smith

Name of the app's owner.

Description

```
An application for Operational Intelligence
```

Enter a description for your app.

Template

barebones

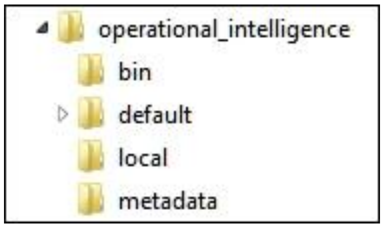
These templates contain example views and searches.

5. When the form is completed, click on **Save**. This should be followed by a blue bar with the message **Successfully saved operational_intelligence**.

Congratulations, you just created a Splunk application!

How it works…

When an app is created through the Splunk GUI, as in this recipe, Splunk essentially creates a new folder (or directory) named `operational_intelligence` within the `$SPLUNK_HOME/etc/apps`directory. Within the `$SPLUNK_HOME/etc/apps/operational_intelligence` directory, you will find four new subdirectories that contain all the configuration files needed for the barebones Operational Intelligence app that we just created.



The eagle-eyed among you would have noticed that there were two templates that could have been selected from when creating the app: `barebones` and `sample_app`. The `barebones` template creates an application with nothing much inside it, and the `sample_app` template creates an application populated with sample dashboards, searches, views, menus, and reports. If you wish, you can also develop your own custom template if you create lots of apps, which might enforce certain color schemes for example.

There's more…

As Splunk apps are just a collection of directories and files, there are other methods to add apps to your Splunk Enterprise deployment.

CREATING AN APPLICATION FROM ANOTHER APPLICATION

It is relatively simple to create a new app from an existing app without going through the Splunk GUI, should you wish to do so. This approach can be very useful when we create multiple apps with different `inputs.conf` files for deployment to Splunk UF.

Taking the app we just created as an example, copy the entire directory structure of the `operational_intelligence` app and name it `copied_app`:

```
cp -r $SPLUNK_HOME$/etc/apps/operational_intelligence/*
$SPLUNK_HOME$/etc/apps/copied_app
```

Within the directory structure of `copied_app`, we must now edit the `apps.conf` file in the default directory.

Open `$SPLUNK_HOME$/etc/apps/copied_app/default/apps.conf`and change the label field to **My Copied App**, provide a new description, and then save the `conf` file:

```
#
# Splunk app configuration file
#
[install]
is_configured = 0

[ui]
is_visible = 1
label = My Copied App

[launcher]
author = John Smith
description = My Copied application
version = 1.0
```

Now restart Splunk, and the new **My Copied App** application should now be seen in the application menu.

```
$SPLUNK_HOME$/bin/splunk restart
```

DOWNLOADING AND INSTALLING A SPLUNK APP

Splunk has an entire application website with hundreds of applications created by Splunk, other vendors, and even the users of Splunk. These are great ways to get started with a base application, which you can then modify to meet your needs.

If the Splunk server that you are logged in to has access to the Internet, from the **Apps** page you can click on the **Browse More Apps** button. From here, you can search for apps and install them directly.

An alternative way to install a Splunk app is to visit http://splunkbase.splunk.com and search for the app. You will then need to download the application locally. From your Splunk server, on the **Apps**page, click on the **Install app from file** button and upload the app you just downloaded in order to install it.

Once the app has been installed, go and look at the directory structure that the installed application just created. Familiarize yourself with some of the key files and where they are located.

TIP

When downloading applications from the Splunk apps site, it is best practice to test and verify them in a non-production environment first. The Splunk apps site is community driven and, as a result, quality checks and/or technical support for some of the apps might be limited.

Adding dashboards and reports

As we saw in the previous session, dashboards are a great way to present many different pieces of information. Rather than having lots of disparate dashboards across your Splunk environment, it makes a lot of sense to group related dashboards into a common Splunk application, for example, putting operational intelligence dashboards into a common Operational Intelligence application.

In this recipe, you will learn how to move the dashboards and the associated reports you created in the last couple of sessions into our new Operational Intelligence application.

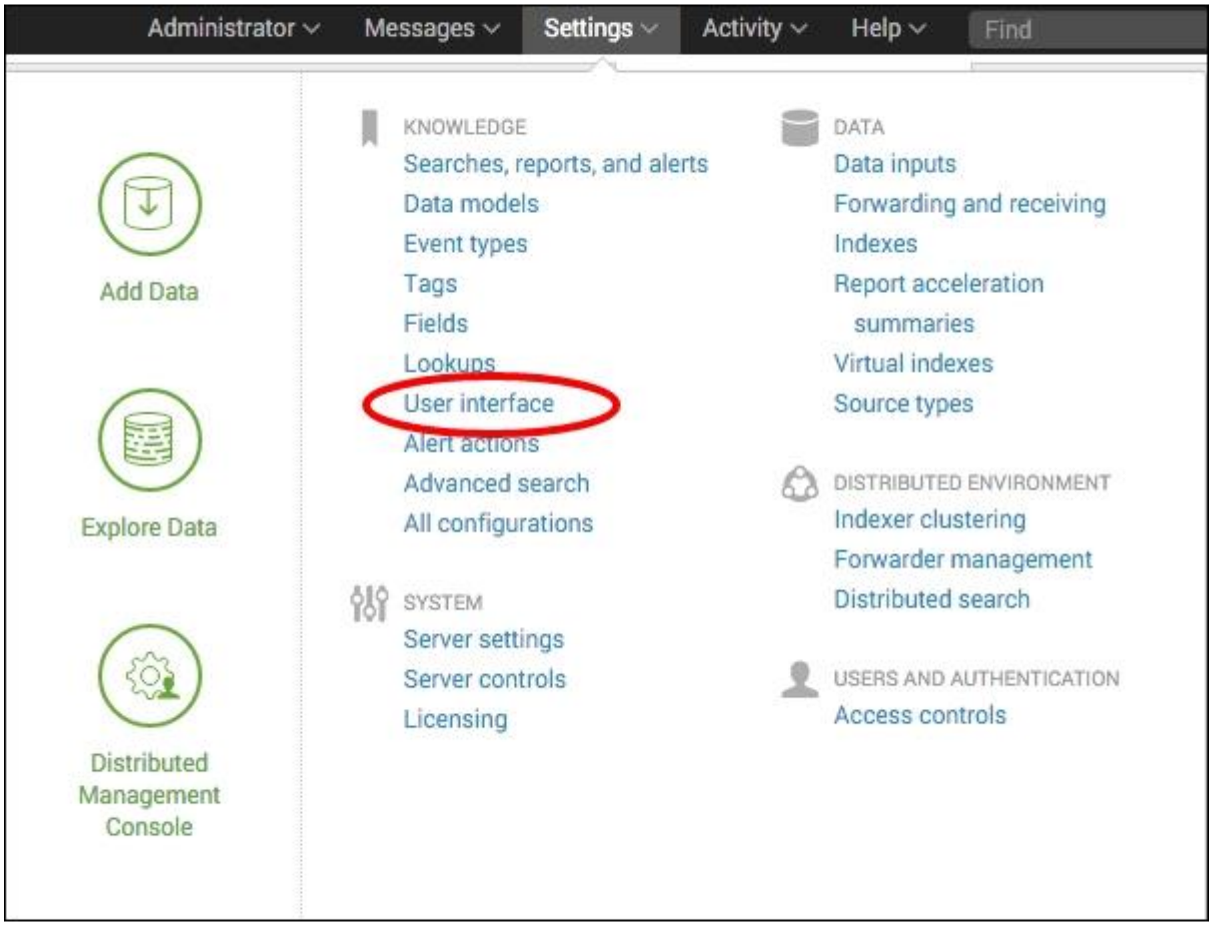
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*. You should have also completed the recipes from the earlier sessions. You should be familiar with navigating the Splunk user interface.

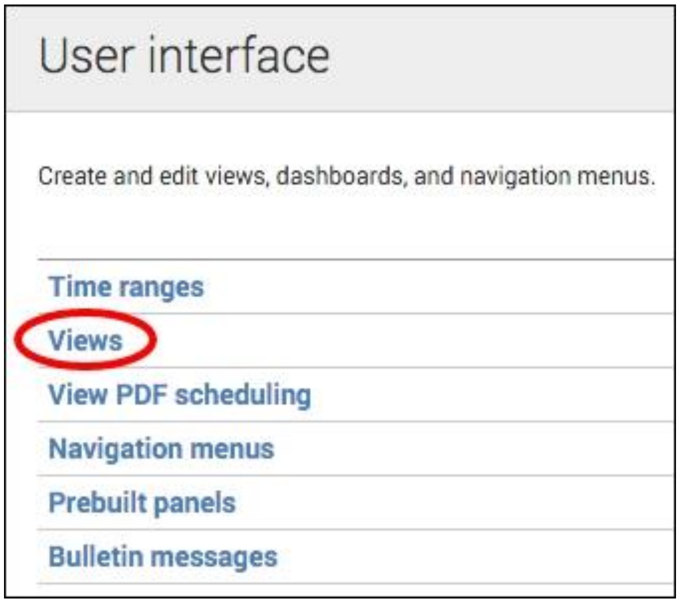
How to do it…

Follow these steps to move your dashboards into the new application:

1. Log in to your Splunk server.
2. Select the newly created **Operational Intelligence** application.
3. From the top menu, select **Settings** and then select the **User interface**menu item.



4. Click on the **Views** section.

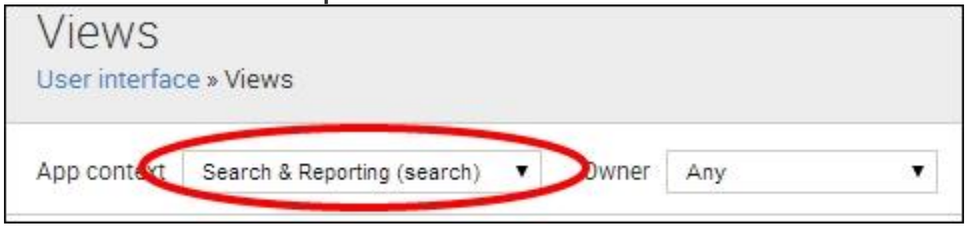


User interface

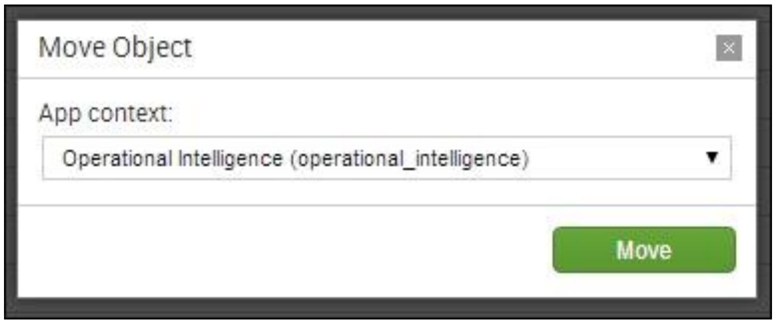
Create and edit views, dashboards, and navigation menus.

Time ranges
Views
View PDF scheduling
Navigation menus
Prebuilt panels
Bulletin messages

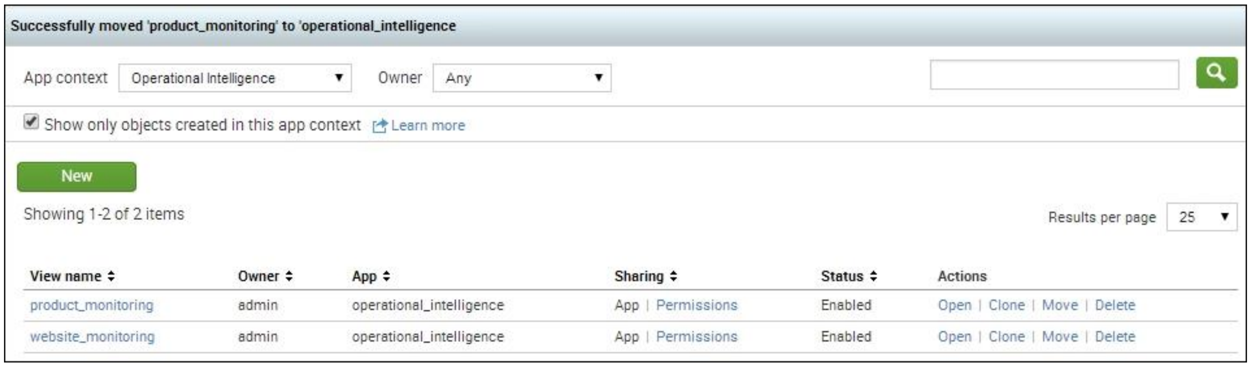
5. In the **App Context** drop-down, select **Searching & Reporting (search)**or whatever application you were in when creating the dashboards in the previous session.



6. Locate the `website_monitoring` dashboard row in the list of views and click on the **Move** link to the right of the row.
7. In the **Move Object** popup, select the **Operational Intelligence (operational_intelligence)** application that was created earlier and then click on the **Move** button.



8. A message bar will then be displayed at the top of the screen to confirm that the dashboard was moved successfully.
9. Repeat from step 5 to move the **product_monitoring** dashboard as well.



Successfully moved 'product_monitoring' to 'operational_intelligence

App context: Operational Intelligence Owner: Any

Show only objects created in this app context

Showing 1-2 of 2 items

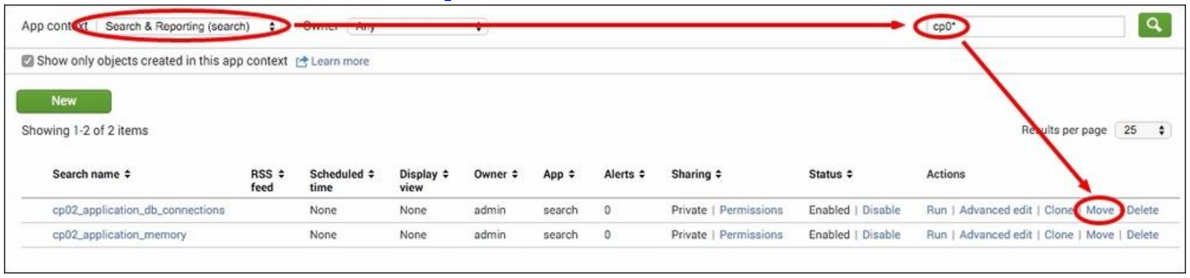
Results per page 25

View name	Owner	App	Sharing	Status	Actions
product_monitoring	admin	operational_intelligence	App | Permissions	Enabled	Open | Clone | Move | Delete
website_monitoring	admin	operational_intelligence	App | Permissions	Enabled	Open | Clone | Move | Delete

10. After the **Website Monitoring** and **Product Monitoring** dashboards have been moved, we next want to move all the reports we created in the previous recipes, as these power the dashboards and provide operational intelligence insight. From

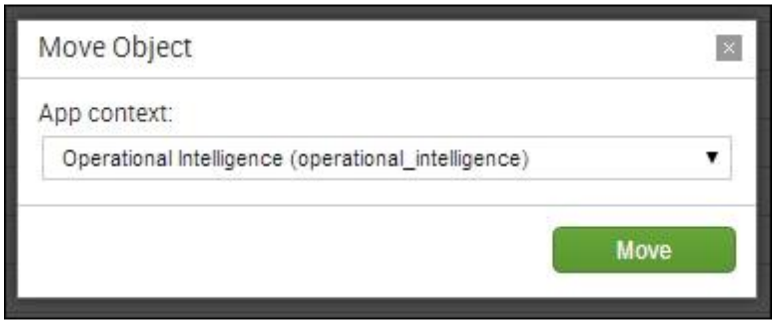
the top menu, select **Settings**, but this time, select **Searches**, **Reports**, and **Alerts**.

11. Select the **Search & Reporting (search)** context and filter by `cp0*` to view the searches (reports) created in Session 3-2, *Diving into Data – Search and Report* and Session 3-3, *Dashboards and Visualizations – Make Data Shine*. Click on the **Move** link of the first `cp0*` search in the list.



Search name	RSS feed	Scheduled time	Display view	Owner	App	Alerts	Sharing	Status	Actions
cp02_application_db_connections		None	None	admin	search	0	Private | Permissions	Enabled | Disable	Run | Advanced edit | Clone | Move | Delete
cp02_application_memory		None	None	admin	search	0	Private | Permissions	Enabled | Disable	Run | Advanced edit | Clone | Move | Delete

12. Select to move the object to the **Operational Intelligence (operational_intelligence)** application and click on the **Move** button.

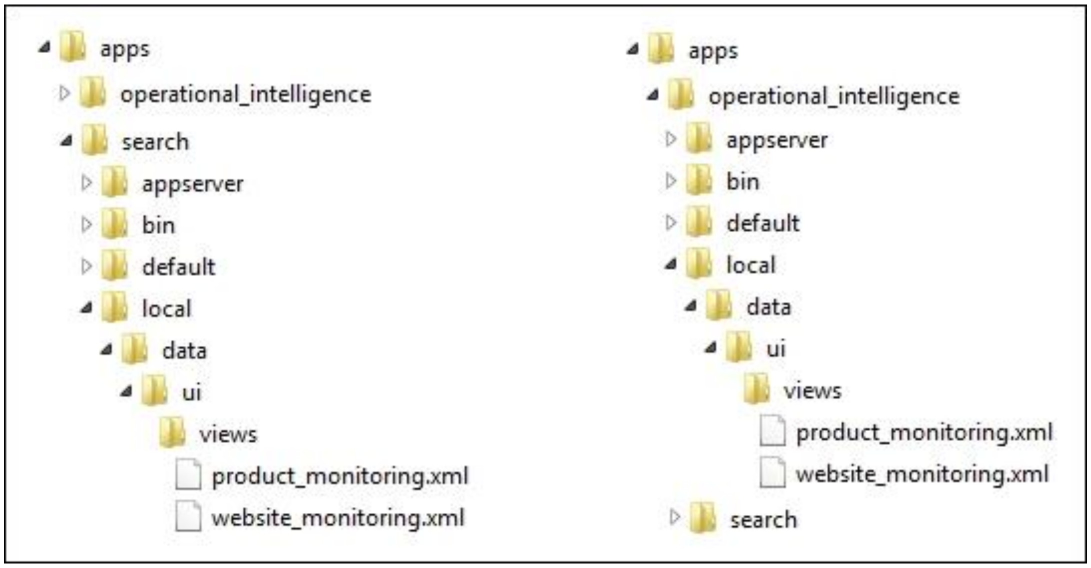


13. A message bar will then be displayed at the top of the screen to confirm that the dashboard was moved successfully.
14. Select the **Search & Reporting (search)** context and repeat from step 11 to move all the other searches over to the new Operational Intelligence application—this seems like a lot but will not take you long!

All the dashboards and reports are now moved to your new Operational Intelligence application.

How it works…

In the previous recipe, we revealed how the Splunk apps are essentially just collections of directories and files. Dashboards are XML files found within the `$SPLUNK_HOME/etc/apps` directory structure. When moving a dashboard from one app to another, Splunk essentially just moves the underlying file from a directory inside one app to a directory in the other app. In this recipe, you moved the dashboards from the **Search & Reporting** app to the **Operational Intelligence** app, as represented in the following screenshot:



As visualizations on the dashboards leverage the underlying saved searches (or reports), you also moved these reports to the new app so that the dashboards maintain permission to access them. Rather than moving the saved searches, we could have changed the permissions of each search to **Global** so that they could be seen from all the other apps in Splunk. However, the other reason we moved the reports was to keep everything contained within a single Operational Intelligence application, which we will continue to build out going forward.

TIP

It is best practice to avoid setting permissions to **Global** for reports and dashboards, as this makes them available to all the other applications when they most likely do not need to be. Additionally, setting global permissions can make things a little messy from a housekeeping perspective and crowd the lists of reports and views that belong to specific applications. The exception to this rule might be for knowledge objects such as tags, event types, macros, and lookups, which often have advantages in being available across all applications.

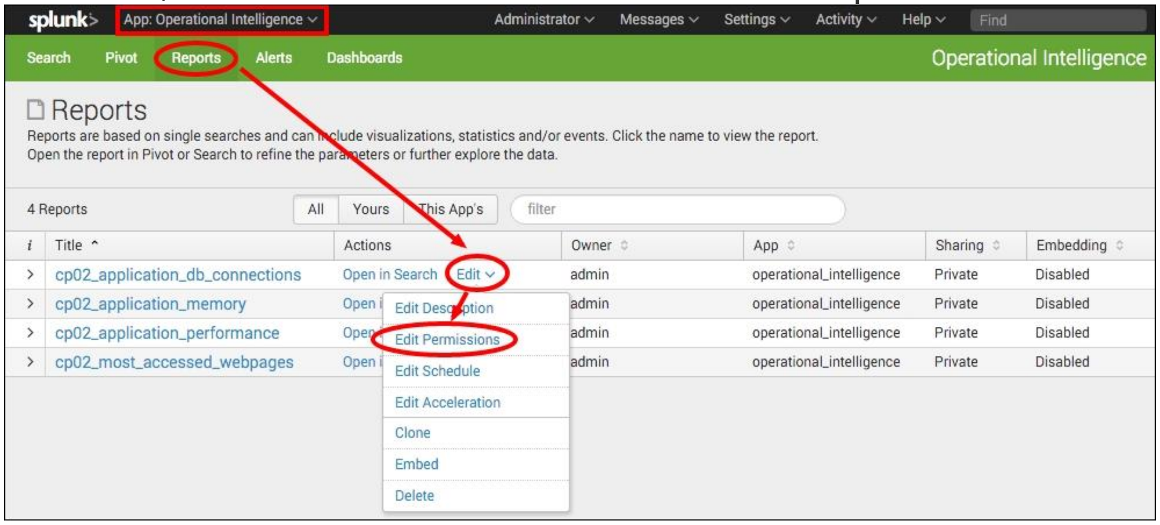
There's more…

As you went through this recipe, you probably noticed that the dashboards had application-level permissions but the reports had private-level permissions. The reports are private as this is the default setting in Splunk when they are created. This private-level permission restricts access to your user account and admin users. In order to make the reports available to the other users of your application, you need to change the permissions of the reports to **Shared in App**, as we did when adjusting the permissions of reports.

CHANGING PERMISSIONS OF SAVED REPORTS

Changing the sharing permission levels of your reports from the default **Private to App** is relatively straightforward:

1. Ensure that you are in your newly created Operational Intelligence application.
2. Select the **Reports** menu item to see the list of reports.
3. Click on **Edit** next to the report you wish to change the permissions for. Then, click on **Edit Permissions** from the drop-down list.



Reports

Reports are based on single searches and can include visualizations, statistics and/or events. Click the name to view the report.
Open the report in Pivot or Search to refine the parameters or further explore the data.

4 Reports

i	Title	Actions	Owner	App	Sharing	Embedding
>	cp02_application_db_connections	Open in Search Edit	admin	operational_intelligence	Private	Disabled
>	cp02_application_memory	Open i	admin	operational_intelligence	Private	Disabled
>	cp02_application_performance	Open	admin	operational_intelligence	Private	Disabled
>	cp02_most_accessed_webpages	Open i	admin	operational_intelligence	Private	Disabled

4. An **Edit Permissions** pop-up box will appear. In the **Display for** section, change from **Owner** to **App** and then click on **Save**.
5. The box will close and you will see that the sharing permissions in the table now display **App** for the specific report. This report will now be available to all the users of your application.

Organizing the dashboards more efficiently

In this recipe, you will learn how to use Splunk's dashboard editor to use more efficient visualizations and organize the dashboards more efficiently. This feature was introduced in Splunk 6 and enhanced even further in more recent versions.

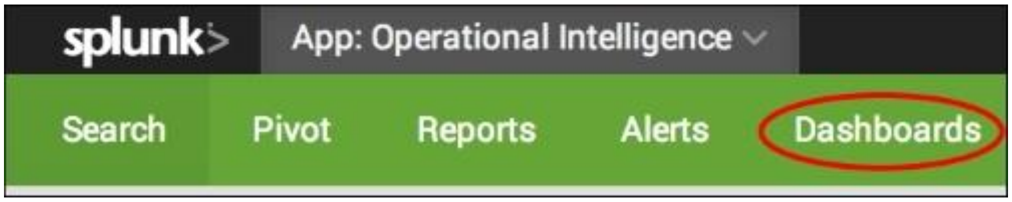
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*, and you should have completed the earlier recipes in this session. You should also be familiar with navigating the Splunk user interface.

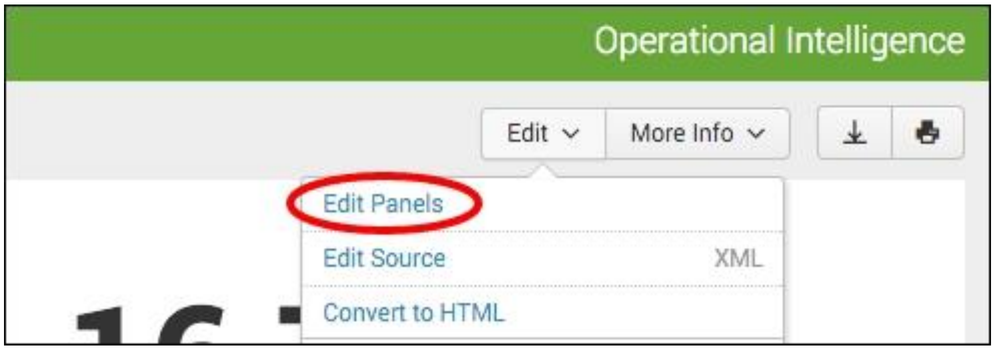
How to do it…

Follow these steps to organize the dashboards more efficiently:

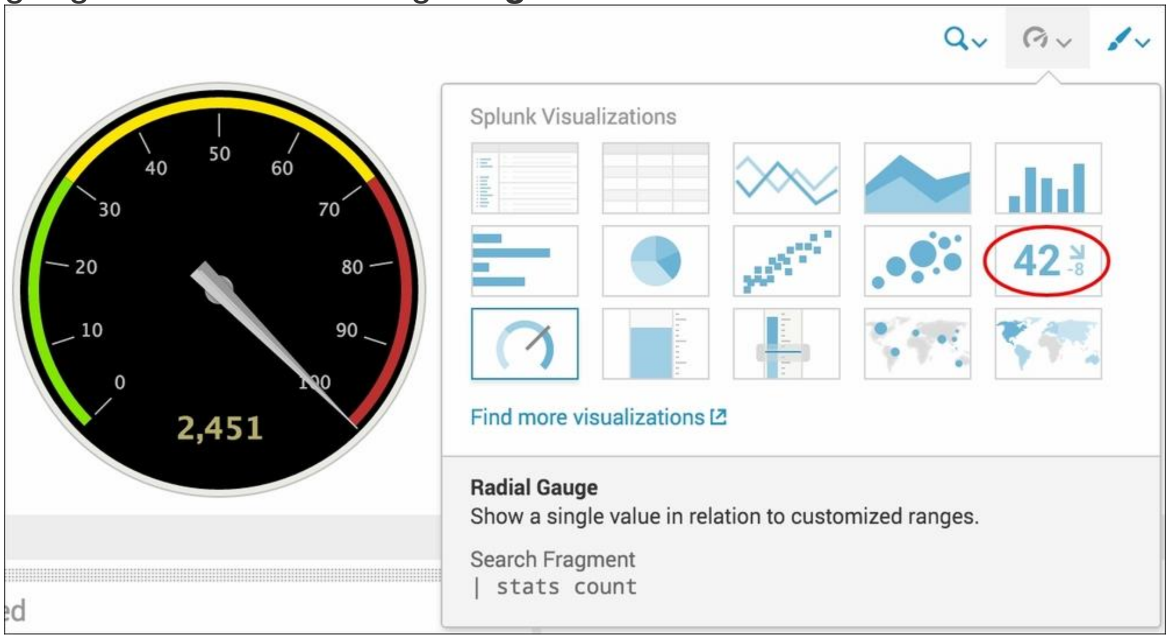
1. Log in to your Splunk server.
2. Select the **Operational Intelligence** application.
3. Click on the **Dashboards** menu item.



4. You should see the **Product Monitoring** and **Website Monitoring**dashboards that we moved into the **Operational Intelligence** app in the previous recipe. Select the **Website Monitoring** dashboard and it will be displayed.
5. You will note that there are several visualizations on the dashboard; in Splunk, they are known as panels. Select **Edit Panels** from the **Edit**menu.



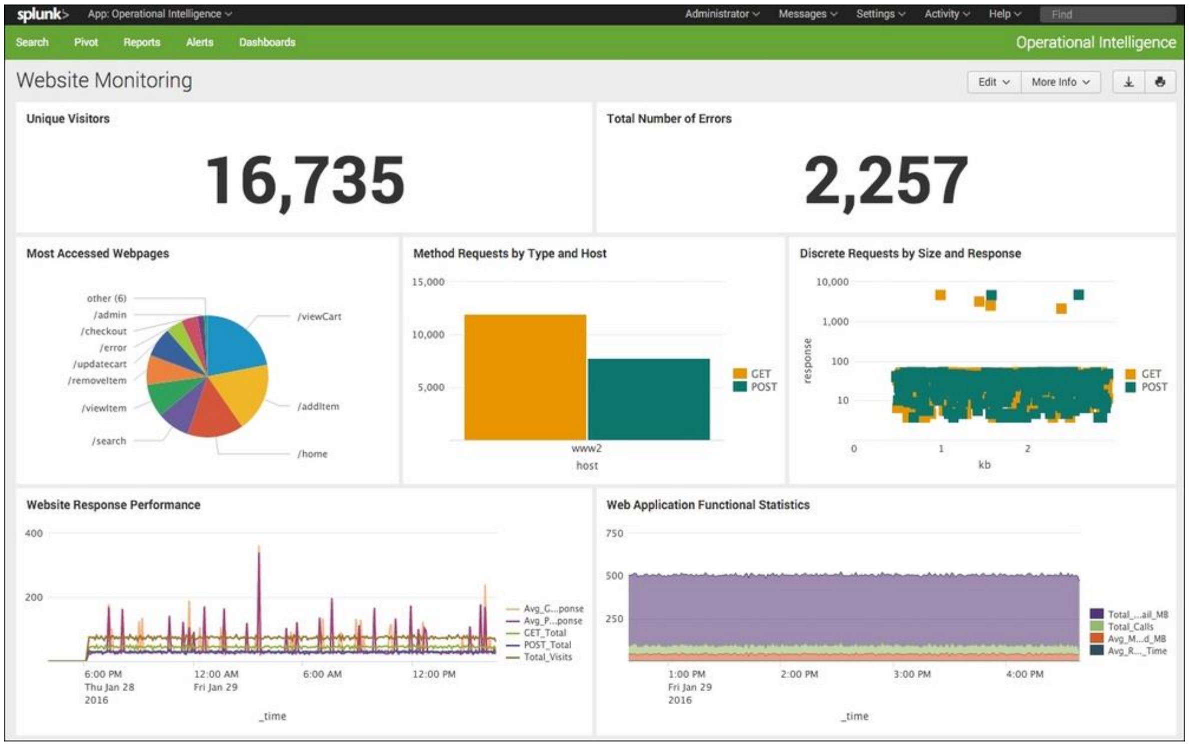
6. In the **Total Number of Errors** panel, change **Radial Gauge** to **Single Value** visualization by clicking on the radial gauge icon and selecting **Single Value**.



Radial Gauge
Show a single value in relation to customized ranges.

Search Fragment
```
| stats count
```

7. Move the other panels around until your dashboard resembles the layout in the following screenshot. Note that we have our single value panels at the top, our various charts in the middle, and then, our time series-based charts at the bottom.



8. Click on the **Done** button when complete. I think you will agree that this dashboard looks a lot better than the earlier one! Everything fits on the screen for the most part, and it is much easier on the eye.

How it works…

In this recipe, we started to experience the power of Splunk's amazing dashboard editor. The dashboard editor provides a nice usable interface that essentially shields the user from what is happening underneath the covers. When we edit the dashboard panels and the visualizations in this manner, Splunk actually writes the required `SimpleXML` code into the respective view's XML file on your behalf. When you click on **Done**, the file is essentially saved with the newly written XML under the covers. If you are hungry for more, don't worry; we are just getting started with the editor. There is plenty more to come later in this session!

NOTE

Dashboards in Splunk are also called Views. In the management interface and in the backend, they are commonly known as Views, but in the application menu, they are called Dashboards. We may use the terms interchangeably in this book.

There's more…

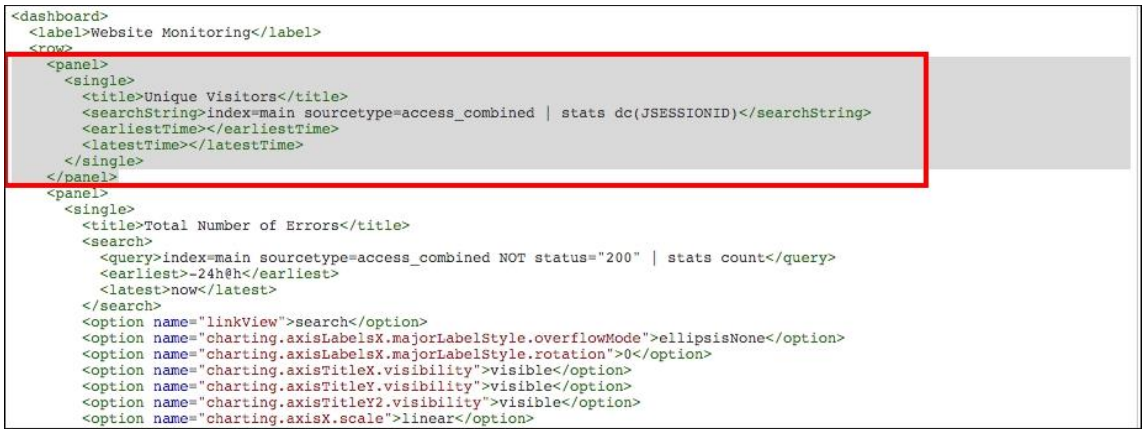
Instead of using the dashboard editor, you can edit the `SimpleXML` directly.

MODIFYING THE SIMPLE XML DIRECTLY

Let's take a look at the `SimpleXML` that is behind the **Website Monitoring**dashboard. Ensure that the dashboard is displayed on the screen. Then, click on the **Edit** button as you did earlier, but instead of clicking on **Edit Panels**, click on **Edit Source**. The underlying `SimpleXML` source code will now be displayed.

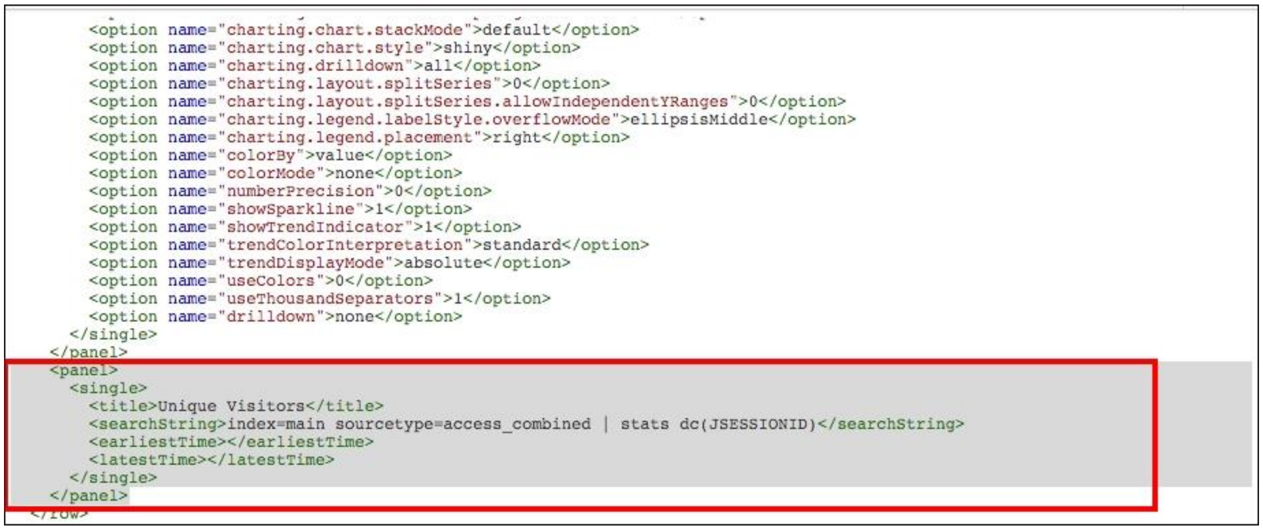
Dashboards in `SimpleXML` consist of rows, panels, and visualization elements. Dashboards can have many rows (`<row></row>`), but around three rows is advisable. Within each row, you can have multiple panels (`<panel></panel>`), and within each panel, you can have multiple visualization elements (for example, `<chart></chart>` for a chart). On the **Website Monitoring** dashboard, you should see three row elements and multiple panels with a single dashboard element on each panel.

We can edit the `SimpleXML` code directly to swap the single elements around on the top row of the dashboard. Simply select the first panel group (The **Unique Visitors** panel XML):



```
<dashboard>
  <label>Website Monitoring</label>
  <row>
    <panel>
      <single>
        <title>Unique Visitors</title>
        <searchString>index=main sourcetype=access_combined | stats dc(JSESSIONID)</searchString>
        <earliestTime></earliestTime>
        <latestTime></latestTime>
      </single>
    </panel>
    <panel>
      <single>
        <title>Total Number of Errors</title>
        <search>
          <query>index=main sourcetype=access_combined NOT status="200" | stats count</query>
          <earliest>-24h@h</earliest>
          <latest>now</latest>
        </search>
        <option name="linkView">search</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.scale">linear</option>
```

Next, move it below the second panel group (The **Total Number of Errors**panel XML).


```
        <option name="charting.chart.stackMode">default</option>
        <option name="charting.chart.style">shiny</option>
        <option name="charting.drilldown">all</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.placement">right</option>
        <option name="colorBy">value</option>
        <option name="colorMode">none</option>
        <option name="numberPrecision">0</option>
        <option name="showSparkline">1</option>
        <option name="showTrendIndicator">1</option>
        <option name="trendColorInterpretation">standard</option>
        <option name="trendDisplayMode">absolute</option>
        <option name="useColors">0</option>
        <option name="useThousandSeparators">1</option>
        <option name="drilldown">none</option>
      </single>
    </panel>
    <panel>
      <single>
        <title>Unique Visitors</title>
        <searchString>index=main sourcetype=access_combined | stats dc(JSESSIONID)</searchString>
        <earliestTime></earliestTime>
        <latestTime></latestTime>
      </single>
    </panel>
  </row>
```

Then, once done, click on the **Save** button. This is an extremely simple example, but you should start to see how we can edit the code directly rather than use the dashboard editor.

Note that there might be some `<option>` data within the panel groups that we have removed in the screenshots to simplify things. However, ensure that you move everything within the panel group.

Familiarizing yourself with Simple XML and learning how to tweak it manually will provide more functionality and probably make the process of creating a dashboard a lot more efficient.

TIP

A great way to learn `SimpleXML` is to modify something using the Splunk dashboard editor and then select to view the code to see what happened in the underlying `SimpleXML` code. Splunk also has a great `SimpleXML` reference that allows quick access to many of the key `SimpleXML` elements.
Visit http://docs.splunk.com/Documentation/Splunk/latest/Viz/PanelreferenceforSimplifiedXML for more information.

Dynamically drilling down on activity reports

When viewing a dashboard in Splunk, there is usually a very high probability that you will look at a chart or report and want to know more details about the information that you are looking at.

Splunk dashboards can be configured to let the user drill down into more details. By linking the results or data points to an underlying dashboard or report, information about what the user clicked on can provide them with the next level of detail or the next step in the process they are following.

This recipe will show you how you can configure reports to drill down into subsequent searches and other dashboards so that you can link them together into a workflow that gets the user to the data they are interested in seeing within your Operational Intelligence application.

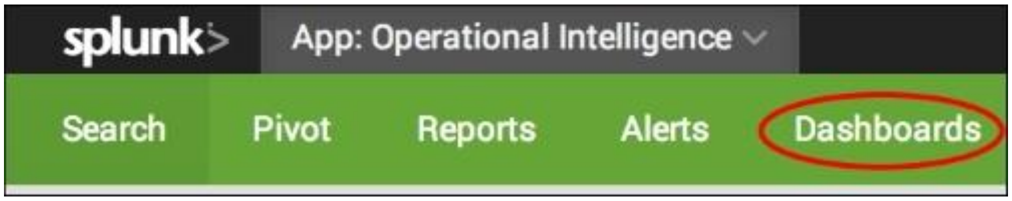
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*, and you should have completed the earlier recipes in this session. You should also be familiar with navigating the Splunk user interface.

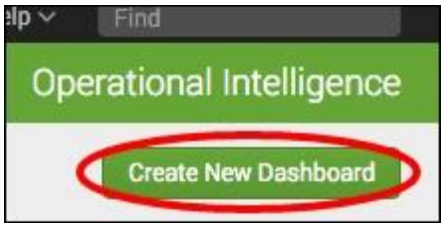
How to do it…

Follow these steps to configure a dashboard report with row drilldown capabilities:

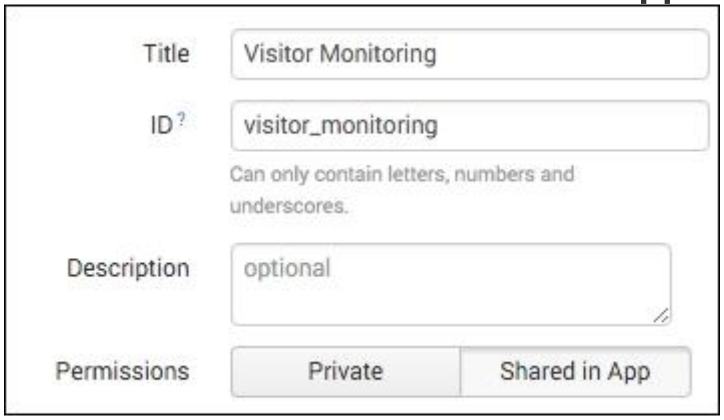
1. Log in to your Splunk server.
2. Select the **Operational Intelligence** application.
3. Click on the **Dashboards** menu.



4. Click on the **Create New Dashboard** button.



5. Name the dashboard `Visitor Monitoring` and set the **Permissions**field to **Shared in App**.



Title: Visitor Monitoring

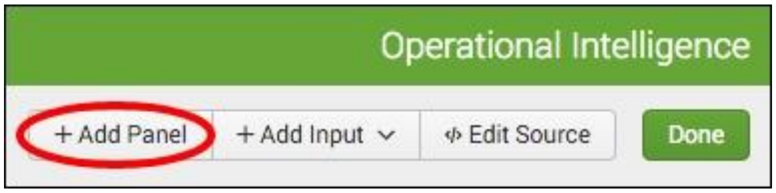
ID: visitor_monitoring

Can only contain letters, numbers and underscores.

Description: optional

Permissions: Private | Shared in App

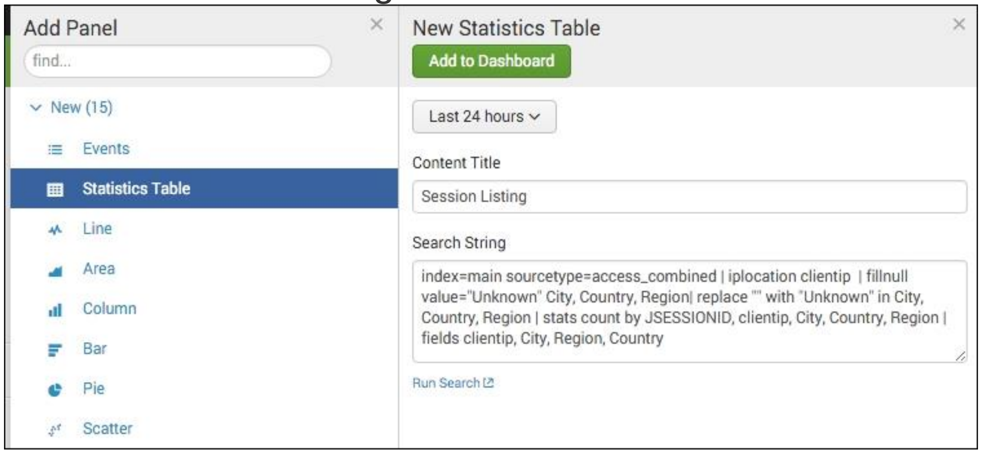
6. Click on **Create Dashboard**.
7. When the empty dashboard is displayed, click on the **Add Panel** button.



8. From the fly-out panel, expand the **New** section then click on **Statistics Table**.
9. Set the **Content Title** panel to **Session Listing**.
10. Set the **Search String** field to the following:

```
index=main sourcetype=access_combined | iplocation clientip  |
fillnull value="Unknown" City, Country, Region| replace "" with
"Unknown" in City, Country, Region | stats count by JSESSIONID,
clientip, City, Country, Region | fields clientip, City,
Region, Country
```

11. Set the time range to **Last 24 hours**.

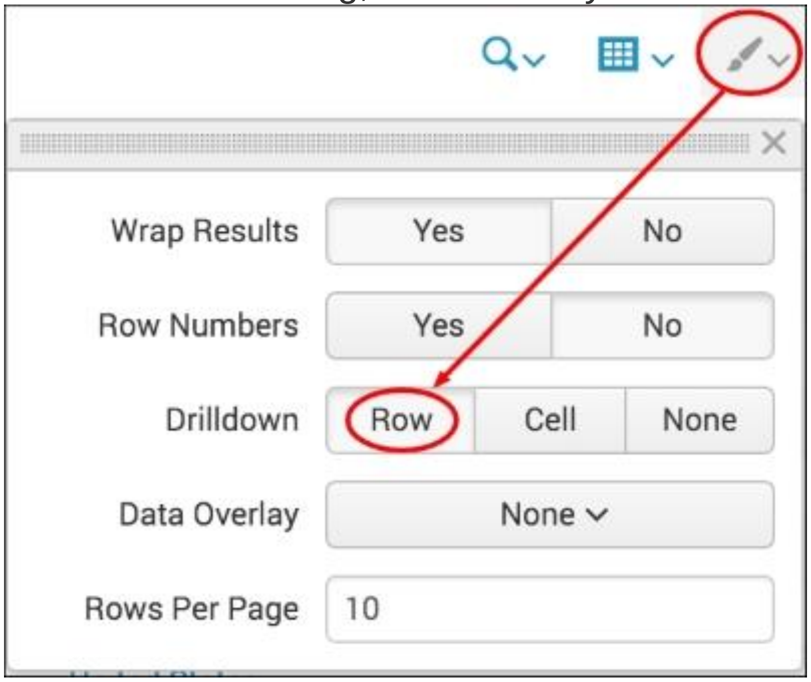


Content Title: Session Listing

Search String:

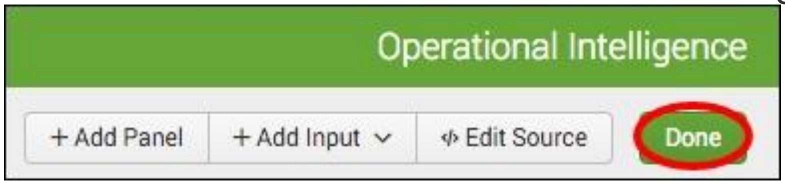
```
index=main sourcetype=access_combined | iplocation clientip  | fillnull
value="Unknown" City, Country, Region| replace "" with "Unknown" in City,
Country, Region | stats count by JSESSIONID, clientip, City, Country, Region |
fields clientip, City, Region, Country
```

12. Click on the **Add to Dashboard** button.
13. Click anywhere on the dashboard to make the fly-out panel disappear.
14. Click on the panel edit icon, select the **Row** option for the **Drilldown**setting, and click anywhere on the page.



Wrap Results: Yes No
Row Numbers: Yes No
Drilldown: [✓ Row] Cell None
Data Overlay: None
Rows Per Page: 10

15. Click on the **Done** button to finish editing the dashboard.



16. Click on a row in the dashboard table, and Splunk will now drill down to the search screen and execute a search that is filtered by the `clientip`value in the row you selected to drill down on.

How it works…

The drilldown feature of dashboards can be utilized to get your users to the next set of data they need. When they click on a table entry or a part of a chart, they set off a search that can drill down into more details about the item they clicked on. The behavior of the drilldown is controlled by the configuration of the panel in the `SimpleXML` code but also has a few options displayed by the dashboard editor.

When displaying a table of results, there are three options that can be chosen from:

Option	**Description**
`Row`	When a row is clicked on, the search that is launched by the drilldown is based on the *x*-axis value, which is the first column in the row.
Cell	When a particular cell is clicked on, the search that is launched by the drilldown is based on both the *x*-axis and *y*-axis values represented by that cell.
None	The drilldown functionality is disabled. When a user clicks on the table, the page will not change.

When displaying a chart, there are two options for the drilldown behavior that can be chosen from:

Option	**Description**
On	When a row is clicked on, the search that is launched by the drilldown is based on the values of the portion of that chart.
Off	The drilldown functionality is disabled. When a user clicks on the table, the page will not change.

When the drilldown search is started after the table or chart is clicked on, it is generally derived by taking the original search, backing off the final transforming commands, and then adding the values that were selected, depending on the drilldown setting.

TIP

When a new panel item is added, such as a chart, table, or map, the default drilldown is always turned on by default.

There's more…

The drilldown options can be customized and provide many different options to control the behavior when dashboards are clicked on.

DISABLING THE DRILLDOWN FEATURE IN TABLES AND CHARTS

To disable the drilldown feature, you can specify the **None** option in the **Drilldown** setting of the edit panel form or add/modify the following `SimpleXML` option to the panel source:

```
<option name="drilldown">none</option>
```

TIP

A full reference of drilldown options can be found in the Splunk documentation at http://docs.splunk.com/Documentation/Splunk/latest/Viz/PanelreferenceforSimplifiedXML#Panel_visualization_elements.

Creating a form for searching web activity

Presenting users with dashboards is a great way to visualize data, as we have seen. However, often people like to *slice n dice* data in many different ways, and to do this, we need to make our dashboards more interactive. We can do this using the dashboard forms functionality of Splunk, which allows the users to filter the dashboard visualizations and data based upon the criteria that are important to them.

This recipe will build on the tabular **Visitor Monitoring** dashboard you created in the previous recipe to allow for granular filtering of the tabulated results.

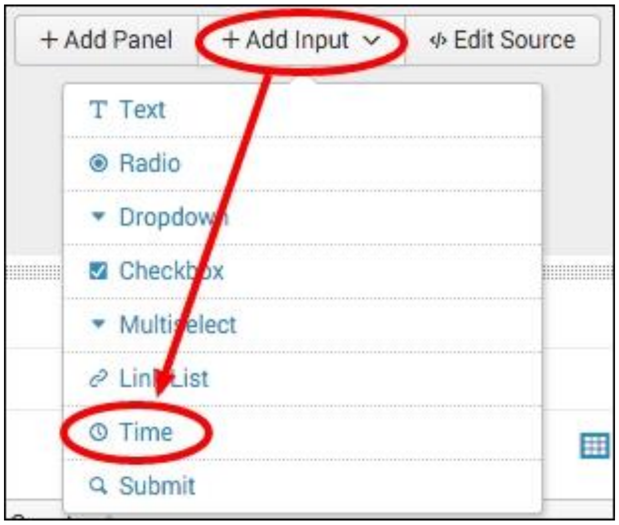
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*, and you should have completed the earlier recipes in this session. You should also be familiar with navigating the Splunk user interface.

How to do it…

Follow these steps to create a form to filter data on a dashboard:

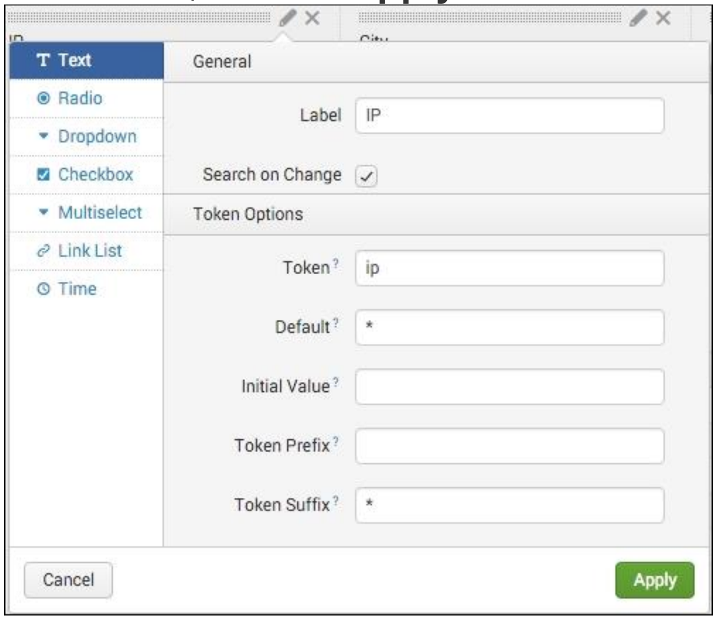
1. Log in to your Splunk server.
2. Select the **Operational Intelligence** application.
3. Click on the **Dashboards** menu item.
4. Select to view the **Visitor Monitoring** dashboard we created in the previous recipe.
5. Once loaded, click on the **Edit** drop-down and then on **Edit Panels**.
6. Click on **Add Input** and then select **Time**.



7. Click on **Add Input** again, and this time select **Text**.
8. A new text input named **field2** will appear. Above the text input you will see a little pencil icon. Click on the pencil icon to edit the input. A popup will be displayed.
9. Complete the box with the values in the following table:

Label	IP
Search on Change	Checked
Token	ip
Default	*
Token Suffix	*

10. Then, click on **Apply**.



Label: IP
Search on Change: ✓
Token: ip
Default: *
Initial Value:
Token Prefix:
Token Suffix: *

11. The box will disappear and you will see that the input is now titled IP.
12. Repeat from step 7 to add and edit the three other textbox fields (one at a time) using the following values for each:
    - **field3**:

Label	`City`
Search on Change	`Checked`
Token	`city`
Default	`*`
Token Suffix	`*`

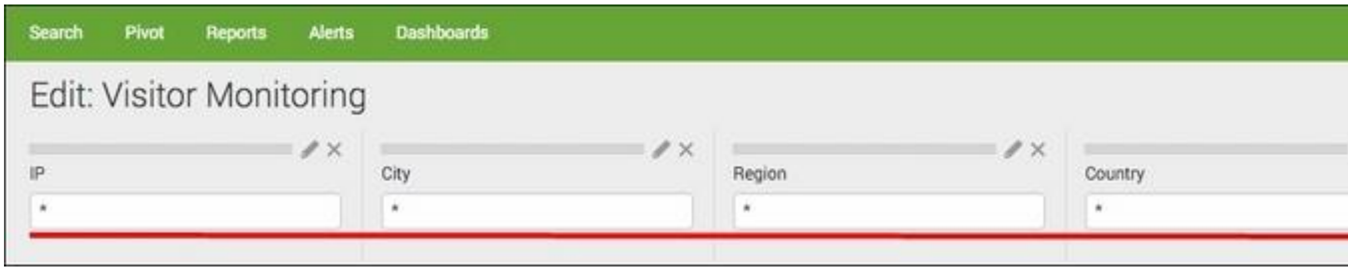
    - **field4**:

Label	`Region`
Search on Change	`Checked`
Token	`region`
Default	`*`
Token Suffix	`*`

    - **field5**:

Label	`Country`
Search on Change	`Checked`
Token	`country`
Default	`*`
Token Suffix	`*`

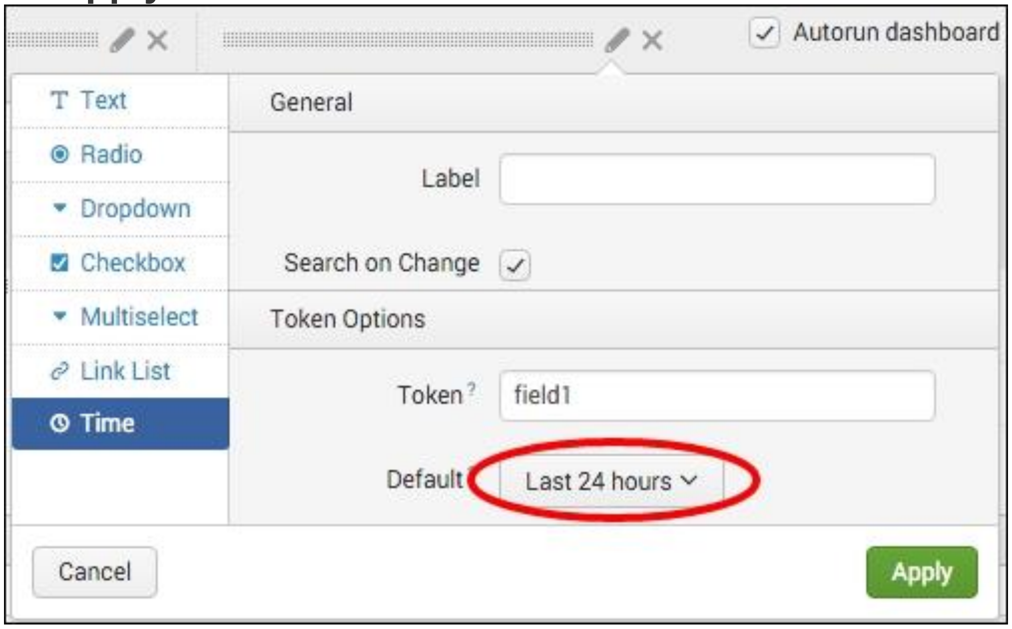
13. Once complete, you should have a total of five fields. Let's now do a bit of rearrangement. Move the **Time input** field to the far right-hand side so that it is the last input field at the top. Additionally, check the **Autorun dashboard** checkbox on the far right-hand side.



Edit: Visitor Monitoring

IP	City	Region	Country
*	*	*	*

14. Click on the pencil icon above the time input and change the default time range from **All Time** to **Last 24 Hours**. Then, click on **Apply**.



General

Label:

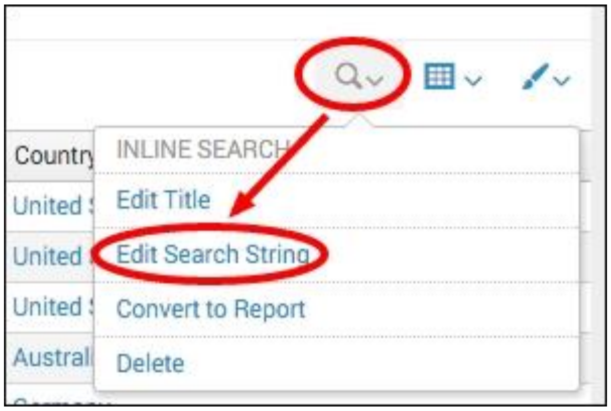
Search on Change: ✓

Token Options

Token: field1

Default: Last 24 hours

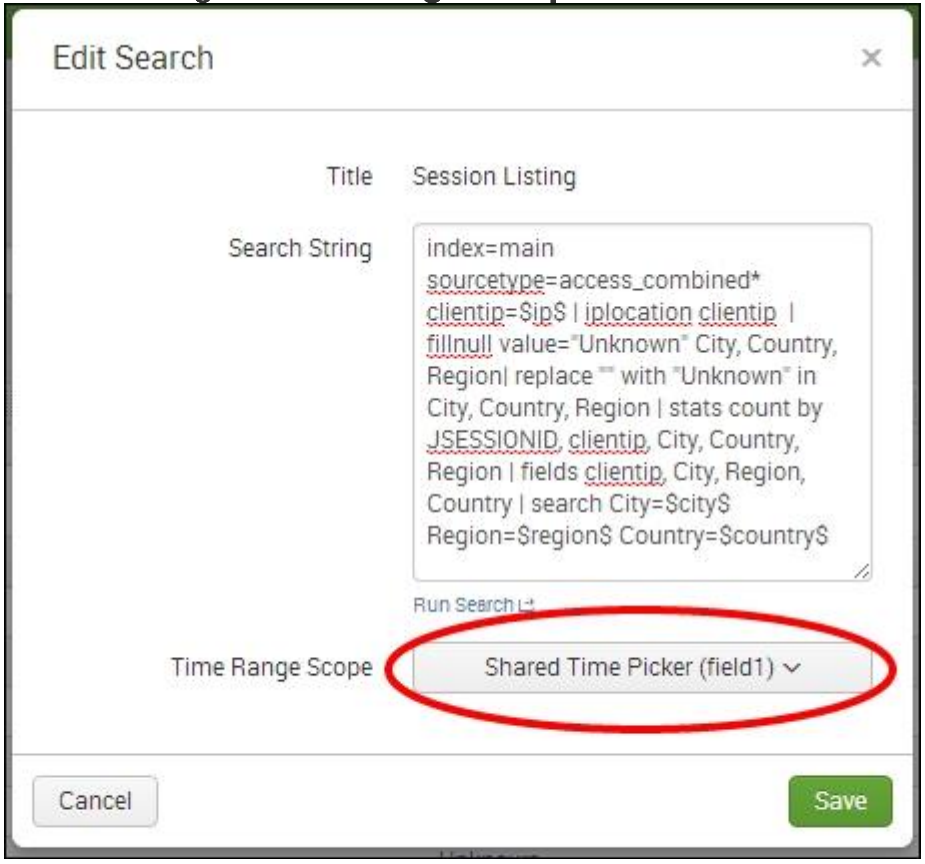
15. Next, click on **Done** on the top right-hand side of the screen to finish editing the form. You should see all your fields nicely labeled across the top with wildcard asterisks (*****) in each textbox.
16. Next, we need to link the new fields we just created with the table. Click on **Edit** and then on **Edit Panels**.
17. Select **Edit Search String** in the panel with the table. A pop-up box will appear with the current search string.



18. Replace the existing search string with the following search string. The modifications to the search have been highlighted:

```
index=main sourcetype=access_combined clientip="$ip$" |
iplocation clientip  | fillnull value="Unknown" City, Country,
Region| replace "" with "Unknown" in City, Country, Region |
stats count by JSESSIONID, clientip, City, Country, Region |
fields clientip, City, Region, Country | search City="$city$"
Region="$region$" Country="$country$"
```

19. Change **Time Range Scope** to **Shared Time Picker (field1)**.


Edit Search

Title: Session Listing

Search String:

```
index=main
sourcetype=access_combined*
clientip=$ip$ | iplocation clientip  |
fillnull value="Unknown" City, Country,
Region| replace "" with "Unknown" in
City, Country, Region | stats count by
JSESSIONID, clientip, City, Country,
Region | fields clientip, City, Region,
Country | search City=$city$
Region=$region$ Country=$country$
```

Time Range Scope: Shared Time Picker (field1)

20. Click on **Save** and then on **Done**.

21. Refresh/reload the dashboard in your browser.
22. That's it! You now have a nice form-driven table. Try testing it out. For example, if you want to filter by all the IP addresses that begin with 134, simply enter `134` into the IP textbox and press *Enter*.

How it works…

In this recipe, we only used the GUI editor, which means that Splunk changed the underlying Simple XML for us. As soon as we added the first input field, Splunk changed the opening Simple XML `<dashboard>` element to a `<form>` element behind the scenes. Each of the five inputs we added is contained within a `<fieldset>` element. For each of the inputs, Splunk creates an `<input>` element, and each input type can have a number of fields: some optional and some required. One of the key fields for each type is the **Token** field, the values of which are then used by searches on the dashboard. We assigned a token name to each input, such as `ip`, `city`, and `country`. Other fields that we populated for the text inputs were the **Default**and **Suffix** field values of `*`. This tells Splunk to search for everything (`*`) by default if nothing is entered into the textbox and to add a wildcard (`*`) suffix to everything entered into the textbox. This means that if we were to search for a city using a value of `Tor`, Splunk would search for all the cities that begin with Tor (`Tor*`), such as Toronto. We also checked the **Search on Change** box, which forces a rerun of any searches in the dashboard should we change a value of the input. After completing the editing of the inputs, we selected to autorun the dashboard, which adds `autoRun="true"` in the `<fieldset>` element of the Simple XML and ensures that the dashboard runs as soon as it is loaded with the default values rather than waiting for something to be submitted in the form.

Once we built the form inputs and configured them appropriately, we needed to tell the searches that power the dashboard visualizations to use the tokens from each of the form inputs. The **Token** field for each input contains the value for that input. We edited the search for the table on the dashboard and added additional search criteria to force Splunk to search based upon these tokens. Token names must be encapsulated by `$` signs, so our `ip`token is entered into the search as `$ip$` and our country token is entered as `$country$`. We also told our search to use the `Shared Time Picker`input rather than its own time range. This allows us to search using the time picker input we added to the form.

The end result is that anything entered into the form inputs is encapsulated into the respective tokens, and the values of these tokens are then passed to the search that powers the table in the dashboard. If we change the value of an input then the value of the input's token in the search changes, the search immediately reruns, and the search results in the table change accordingly.

There's more…

In this recipe, we began to scratch the surface of form building, using only textbox inputs and the time picker. Later in this book, we will leverage the drop-down input, and you will learn how to prepopulate the drop-down values as a result of a search and learn how to filter the drop-down values as a result of other drop-down selections.

ADDING A SUBMIT BUTTON TO YOUR FORM

In this recipe, you probably noticed that there was no **Submit** button. The reason for this was primarily because no **Submit** button was needed. We selected to autorun the dashboard and selected **Search on Change** for each input. However, there are times when you might not want the form to run as soon as something is changed; perhaps, you want to modify multiple inputs and then search. Additionally, many users like the reassurance of a **Submit**button, as it is commonly used on forms across websites and applications.

Adding a **Submit** button is extremely simple. When the dashboard is in the editing mode, simply click on the **Add Input** drop-down and select **Submit**. You will note that a green **Submit** button now appears on the form. If you now edit the text inputs and uncheck the **Search on Change** checkbox for each of them, the form will only be submitted when someone clicks on the **Submit** button.

Linking web page activity reports to the form

Form searches in Splunk do not need to be limited to displaying events and table-driven data. Rich visualizations can also be linked to forms and be updated when the forms are submitted.

This recipe will show you how you can extend a form to include charts and other visualizations that can be driven by the form created to show visitor traffic and location data.

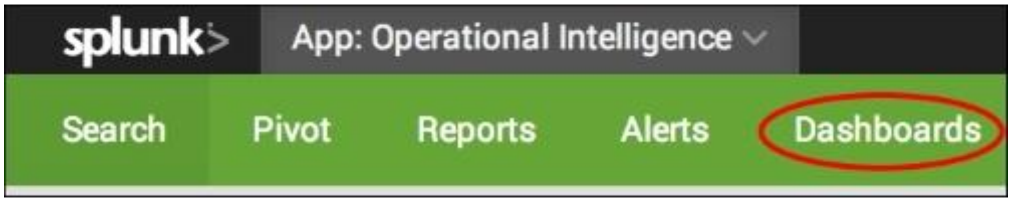
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*, and you should have completed the earlier recipes in this session. You should also be familiar with navigating the Splunk user interface.

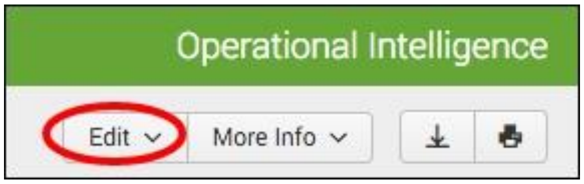
How to do it…

Follow these steps to add a web page activity chart and link it to a form:

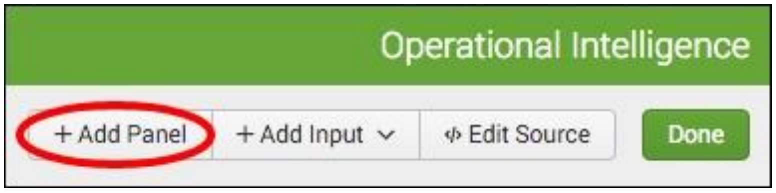
1. Log in to your Splunk server.
2. Select the default **Operational Intelligence** application.
3. Select the **Dashboards** menu item.



4. Select the **Visitor Monitoring** dashboard.
5. Click on the **Edit** button and then on **Edit Panels**.



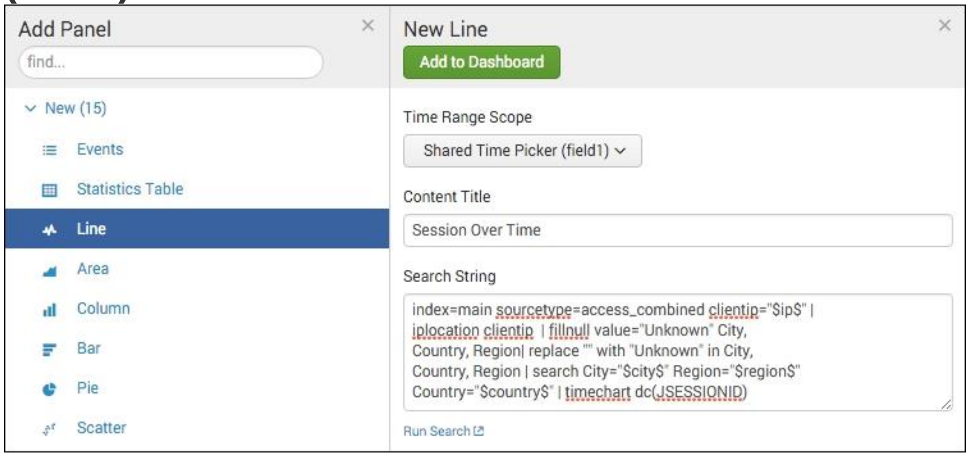
6. Click on the **Add Panel** button.



7. From the fly-out panel, expand the **New** section and then click on **Line**.
8. Set the **Content Title** panel to **Session Over Time**.
9. Set the **Search String** field to the following:

```
index=main sourcetype=access_combined clientip="$ip$" |
iplocation clientip  | fillnull value="Unknown" City, Country,
Region| replace "" with "Unknown" in City, Country, Region |
search City="$city$" Region="$region$" Country="$country$" |
timechart dc(JSESSIONID)
```

10. Set the **Time Range Scope** field to **Shared Time Picker (field1)**.

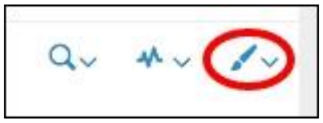


Time Range Scope
Shared Time Picker (field1)

Content Title
Session Over Time

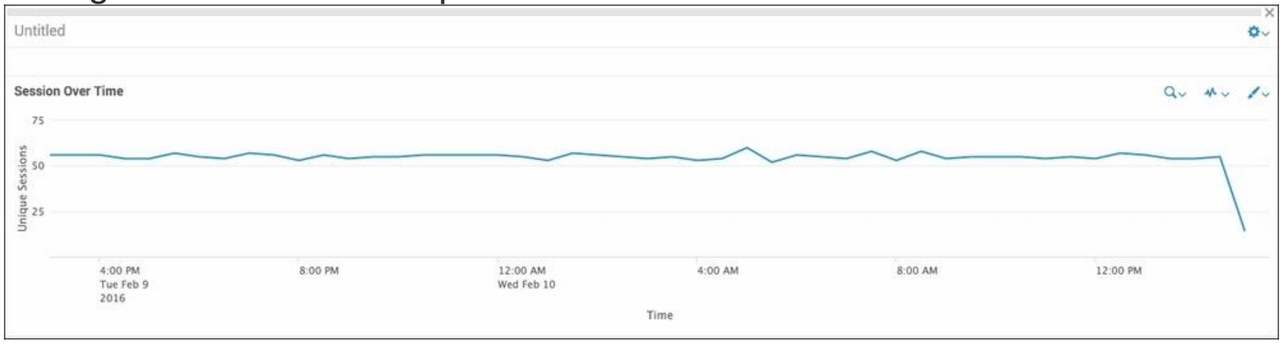
Search String
index=main sourcetype=access_combined clientip="ip" | iplocation clientip | fillnull value="Unknown" City, Country, Region| replace "" with "Unknown" in City, Country, Region | search City="$city$" Region="$region$" Country="$country$" | timechart dc(JSESSIONID)

11. Click on **Add to Dashboard**.
12. Click anywhere on the dashboard to make the fly-out panel disappear.
13. Click on the edit panel icon in the panel we just added to the dashboard.

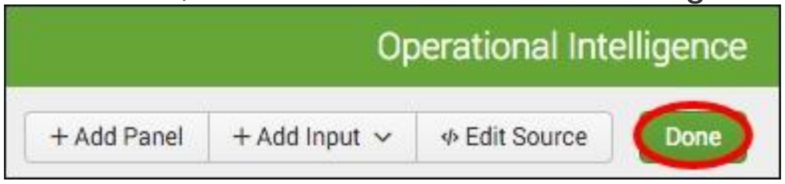


14. Update the **X-Axis** label with **Custom Title** set to **Time**.
15. Update the **Y-Axis** label with **Custom Title** set to **Unique Sessions**.
16. Set the **Legend** option to **None**.

17. Click on **Apply**. The pop-up box will disappear with the changes reflected on the panel.



18. Next, click on **Done** to finish editing the dashboard.



19. Filter by an IP of 134 or similar again, and you should see that the chart panel also changes along with the table panel.

How it works…

Adding a chart to the dashboard works in a manner very similar to the way in which the original form was created. You can utilize the field variables defined in the form in the inline search that is used for the chart. Splunk will set them when the form is submitted. The panel can also utilize the time range that was used in the form or contain a separate time range drop-down.

By building a form and several different charts and tables, you can build a very useful form-driven dashboard. One of the great uses of a form-driven dashboard is for investigative purposes. In this example, you can take any of the fields and, for instance, see all sessions that are coming from a particular country and then see the level of activity over the time period you are interested in.

There's more…

Additional customizations can be added to the charts in order to give them more meaning.

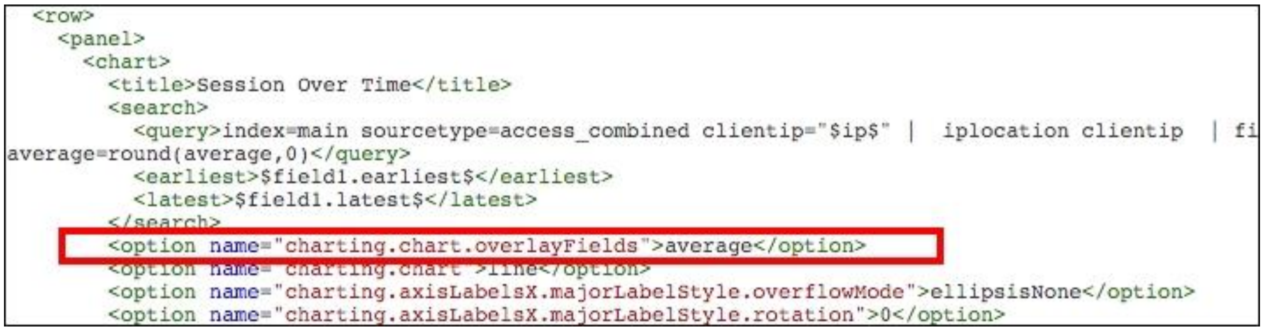
ADDING AN OVERLAY TO THE SESSIONS OVER TIME CHART

You can have Splunk overlay a field value on top of your existing chart to provide trendlines and so on. Add the following line to the end of the inline search used for the **Sessions Over Time** search:

```
| eventstats avg(dc(JSESSIONID)) as average | eval
average=round(average,0)
```

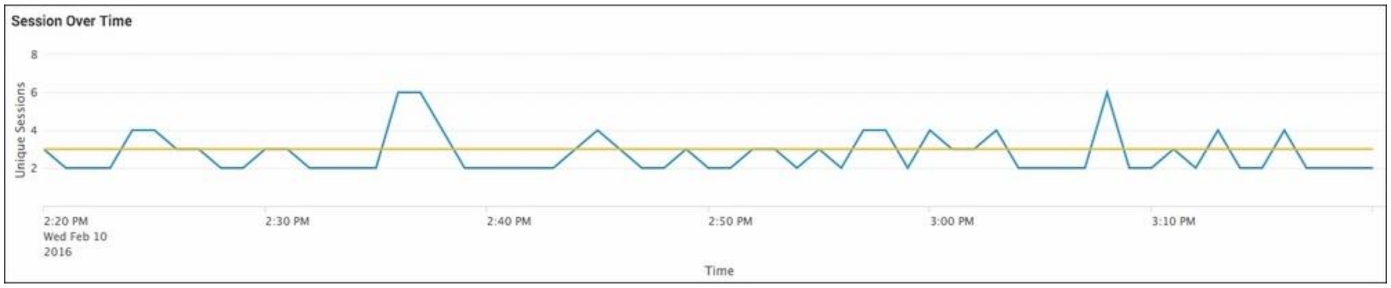
Then, add the following line to the Simple XML of the panel:

```
<option name="charting.chart.overlayFields">average</option>
```



```
  <row>
    <panel>
      <chart>
        <title>Session Over Time</title>
        <search>
          <query>index=main sourcetype=access_combined clientip="$ip$" |  iplocation clientip   | fi
average=round(average,0)</query>
          <earliest>$field1.earliest$</earliest>
          <latest>$field1.latest$</latest>
        </search>
        <option name="charting.chart.overlayFields">average</option>
        <option name="charting.chart">line</option>
        <option name="charting.axisLabelsX.majorLabelStyle.overflowMode">ellipsisNone</option>
        <option name="charting.axisLabelsX.majorLabelStyle.rotation">0</option>
```

It will then add a line that charts the average of the session count over top of the actual values. The chart overlay functionality can also be added from the panel editor under the edit panel icon.



Displaying a geographical map of visitors

Operational intelligence doesn't always need to come in the form of pie charts, bar charts, and data tables. With a wide range of operational data being collected from IT systems, there is the opportunity to display this data in ways that can be more meaningful to users or help present it in ways that can be easier to identify trends or anomalies.

One way that always provides great visibility is representing your data using a geographical map. With geolocation data available for many different data types, it becomes very easy to plot them. Using IP addresses from web server logs is a very common use case for this type of visualization. Splunk allows the easy addition of a map to a dashboard with the capability to zoom and update the portion of the map that the user is viewing.

This recipe will show you how you can configure a map panel within a dashboard and link it to a search that contains IP addresses in order to visualize from where in the world the IP traffic is originating.

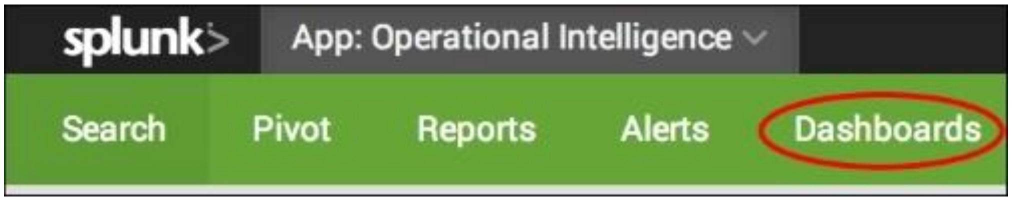
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*, and you should have completed the earlier recipes in this session. You should also be familiar with navigating the Splunk user interface.

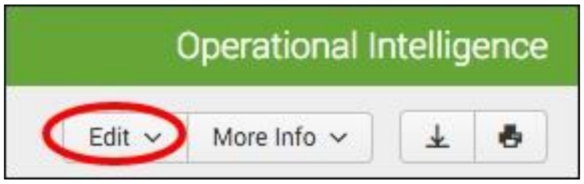
How to do it…

Follow these steps to add a map to your form-driven dashboard:

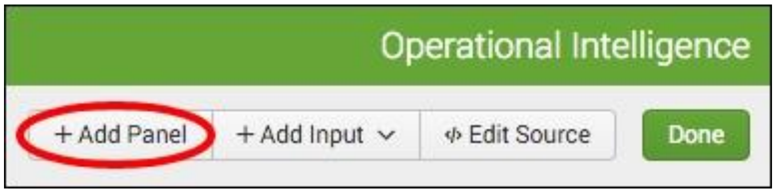
1. Log in to your Splunk server.
2. Select the **Operational Intelligence** application.
3. Click on the **Dashboards** menu item.



4. Select the **Visitor Monitoring** dashboard.
5. Click on the **Edit** button.



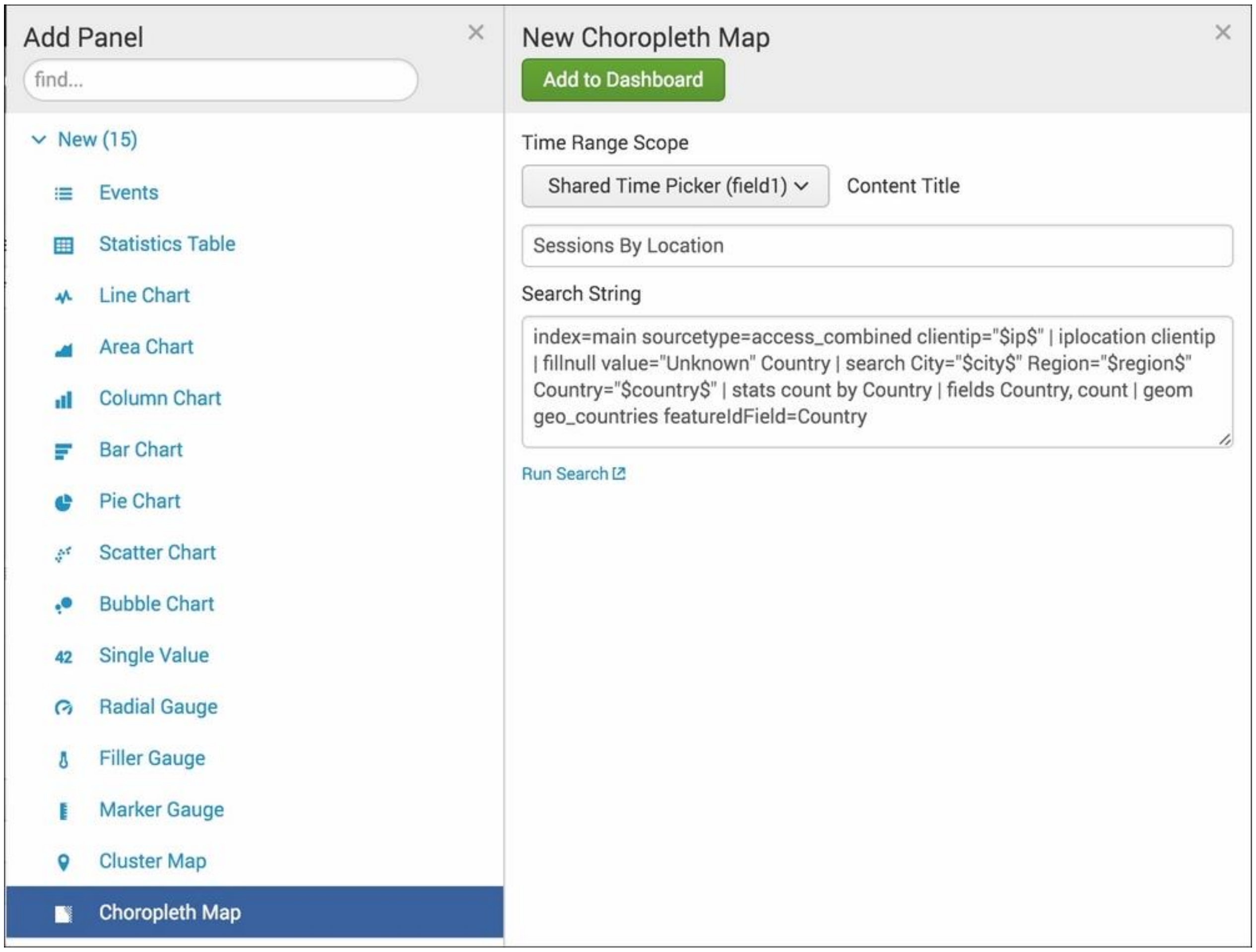
6. Click on the **Add Panel** button.



7. From the fly-out panel, expand the **New** section and then click on **Choropleth Map**.
8. Set the **Content Title** panel to **Sessions By Location**.
9. Set the **Search String** field to the following:

```
index=main sourcetype=access_combined clientip="$ip$" |
iplocation clientip  | fillnull value="Unknown" Country |
search City="$city$" Region="$region$" Country="$country$" |
stats count by Country | fields Country, count | geom
geo_countries featureIdField=Country
```

10. Set the **Time Range Scope** field to **Shared Time Picker (field1)**.



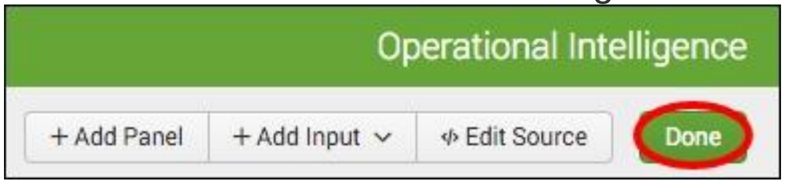
Time Range Scope: Shared Time Picker (field1)

Content Title: Sessions By Location

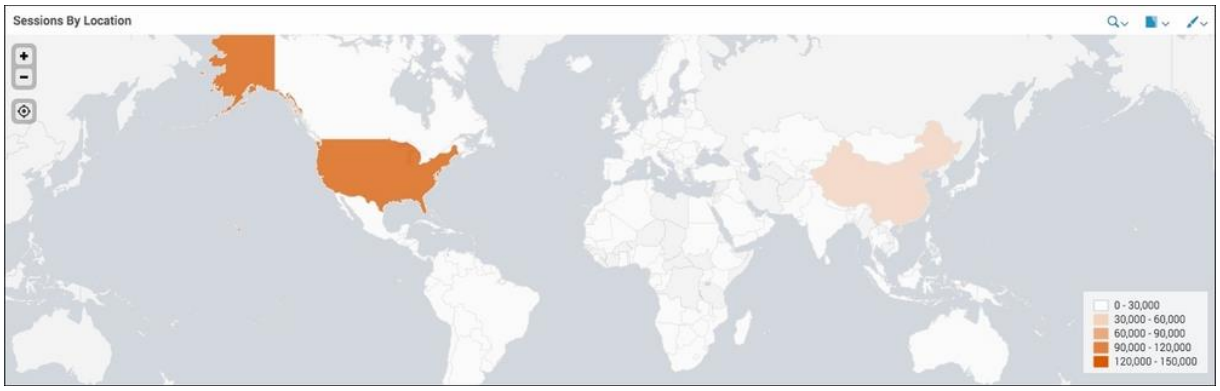
Search String:

```
index=main sourcetype=access_combined clientip="$ip$" | iplocation clientip
| fillnull value="Unknown" Country | search City="$city$" Region="$region$"
Country="$country$" | stats count by Country | fields Country, count | geom
geo_countries featureIdField=Country
```

11. Click on the **Add to Dashboard** button.
12. Click anywhere on the dashboard to make the fly-out panel disappear.
13. Click on **Done** to finish editing the dashboard.



14. Filter by an IP of 134 or similar again, and you should now see that the map panel also changes along with the table and chart panels you added earlier.



How it works…

Mapping support has been available since Splunk 4 using a third-party developed application. Since Splunk 6, native map support has been available and can be used easily within your dashboards. It was given a big boost in 6.3 with the addition of geo-lookups and Choropleth maps.

The rendering of the map is done in the same way in which most browser-based maps are generated - using many small images known as tiles that are put together in a grid layout and swapped in and out depending on the zoom level and the visible area being requested. This results in the browser and services not needing to load an entire world's worth of image data into memory. Layers are then rendered on top of the tiles based on either markers or shapes (polygons).

Splunk currently has two mapping types that can be used:

Type	**Description**
Choropleth	A Choropleth map uses shading to show relative metrics, such as population or election results, for predefined geographic regions.
Marker	Marker maps can plot geographic coordinates as interactive markers. These markers can be configured to represent a metric such as a pie chart with details about that location.

Splunk supports both a native tile server that can be used to serve the actual map images or can be configured to use the external OpenStreetMap service (http://www.openstreetmap.org/#map=5/51.500/-0.100). The native tiles do not have a very granular level of mapping detail but work in situations where there is no external connectivity or there are security reasons for not calling the external service.

In this recipe, the map panel depends on the result of the `geom` command, which looks for the necessary feature ID fields in the search results and adds its own data to that the map can use to render the shapes properly. The `iplocation` command is commonly used to map network traffic-originating locations.

The built-in IP location data within Splunk is provided by Splunk as part of Splunk Enterprise but is not always the most up-to-date data available from the Internet. It's often best practice to purchase a third-party service to get the most accurate and real-time data available, especially when it is used on critical security-monitoring dashboards and searches.

The map panel has many different configuration options that can be used to specify the initial latitude, longitude, and zoom level that should be applied when the map is initially loaded, as well as the minimum and maximum zoom levels. Drilldown in the maps is also supported.

TIP

More details on Mapping in Splunk is available at http://docs.splunk.com/Documentation/Splunk/latest/Viz/Choroplethmaps.

A full reference of map drilldown options can be found in the Splunk documentation at http://docs.splunk.com/Documentation/Splunk/latest/Viz/PanelreferenceforSimplifiedXML#Panel_visualization_elements.

There's more…

The map panel option can also be configured in several different ways in Splunk.

ADDING A MAP PANEL USING SIMPLE XML

A map panel can be added directly to a dashboard by adding the following Simple XML when editing the dashboard source:

```
<row><panel>

<map>

        <title>Count by location</title>

        <searchString>index=main sourcetype=access_combined
clientip="$ip$" | iplocation clientip  | fillnull
value="Unknown" City, Country, Region| replace "" with
"Unknown" in City, Country, Region | search City="$city$"
Region="$region$" Country="$country$" | geostats
count</searchString>

        <earliestTime>-24h@m</earliestTime>

        <latestTime>now</latestTime>

        <option name="mapping.data.maxClusters">100</option>

        <option name="mapping.drilldown">all</option>

        <option name="mapping.map.center">(0,0)</option>

</map>

</panel></row>
```

MAPPING DIFFERENT DISTRIBUTIONS BY AREA

The `geostats` command takes an aggregation term as its main argument. This term is what is used to render the pie charts that are located on the map. In this recipe, we simply ran `| geostats` count, which is the most commonly used command and simply does a single count. However, you can break out the data by product, and then the pie charts will provide segmented visual information and can be moused over to see the breakdown.

```
MySearch | geostats count by product
```

Scheduling PDF delivery of a dashboard

Getting Operational Intelligence to the users who need it the most can be challenging. They can be users who are not IT savvy, don't have the correct access to the right systems, or are executives about to walk into a client meeting to go over the latest result data.

Sometimes, all a user needs is to have data e-mailed to their inbox every morning so that they can review it on their commute to the office or have an assistant prepare for a morning briefing. Splunk allows the users to schedule a dashboard so that it can be delivered as a PDF document via e-mail to a customizable list of recipients.

This recipe will show you how to schedule the delivery of a dashboard within the Operational Intelligence application as a PDF document to an internal e-mail distribution list.

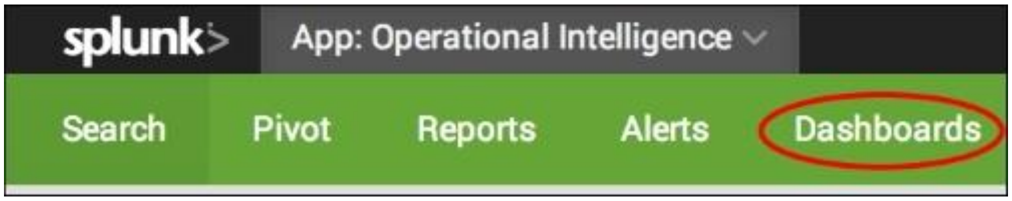
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server, with the sample data loaded from Session 3-1, *Play Time – Getting Data In*, and you should have completed the earlier recipes in this session. You should also be familiar with navigating the Splunk user interface. You should also have configured your e-mail server to work with Splunk so that Splunk can actually send e-mails to specified addresses.

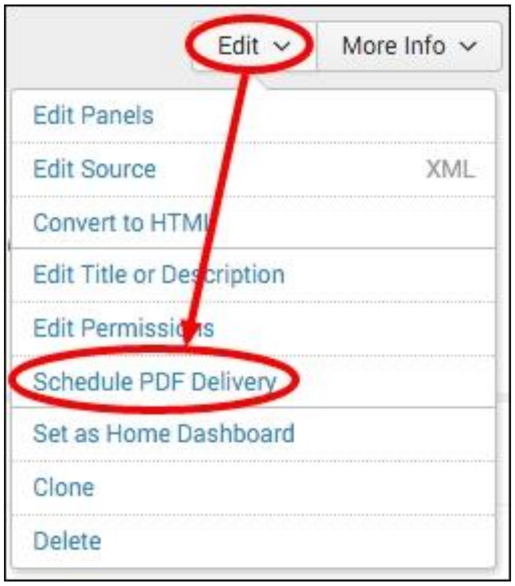
How to do it…

Follow these steps to schedule a PDF delivery of your dashboard:

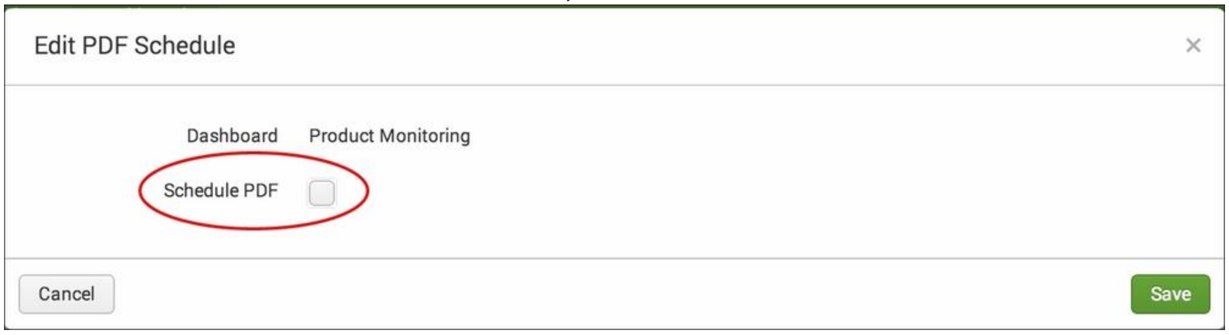
1. Log in to your Splunk server.
2. Select the **Operational Intelligence** application.
3. Click on the **Dashboards** menu item.



4. From the dashboard listing, select the dashboard you would like to deliver as a PDF document. Only the **Website Monitoring** and **Product Monitoring** dashboards can leverage PDF delivery, as the PDF delivery function is not (currently) compatible with the dashboards driven by form inputs.
5. Once the selected dashboard loads, click on the **Edit** drop-down menu on the top right-hand side of the screen.
6. Click on the **Schedule PDF Delivery** option.



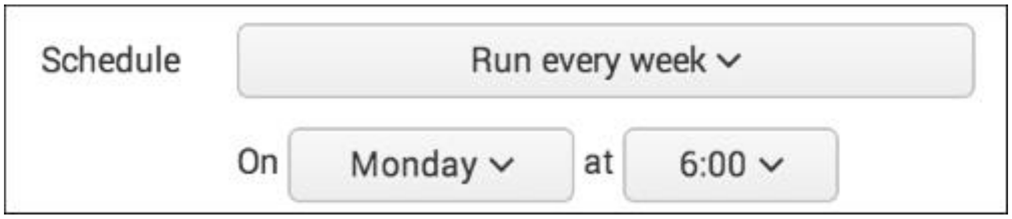
7. On the **Edit PDF Schedule** form, check the **Schedule PDF** box.



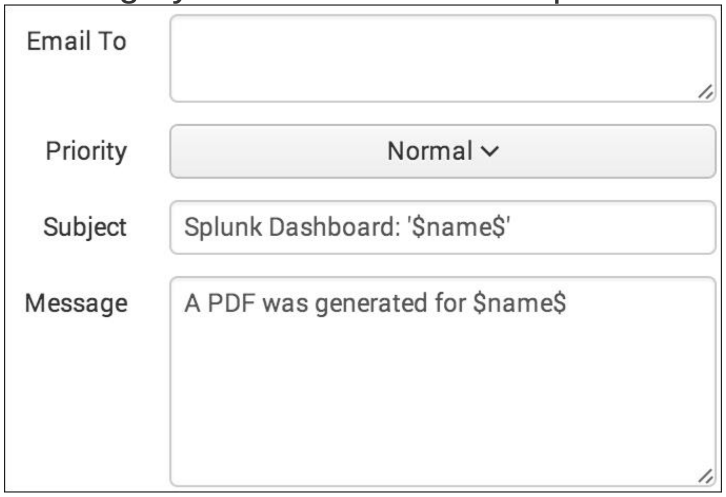
Dashboard Product Monitoring

Schedule PDF ☐

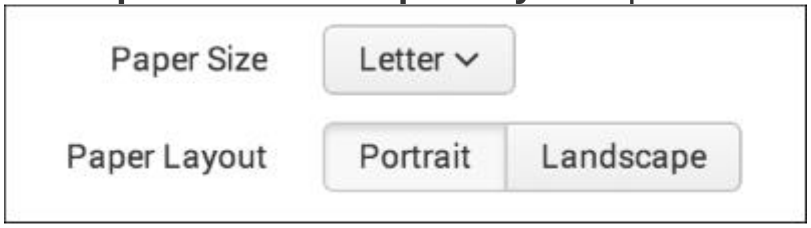
8. Modify the **Schedule** field to suit your needs. Update the drop-down and select the appropriate schedule type.


Schedule: Run every week
On Monday at 6:00

9. Enter the list of e-mail addresses you wish to send the PDF to in the **Email To** field, using commas to separate multiple e-mail addresses.
10. Select the priority of the e-mail.
11. Customize the **Subject** field with the content of the message subject you would like the recipients to see.
12. Customize the **Message** field with the content of the message you would like the recipients to see.


Email To:
Priority: Normal
Subject: Splunk Dashboard: '$name$'
Message: A PDF was generated for $name$

13. Update the layout options of the generated PDF by updating the **Paper Size** and **Paper Layout** options.


Paper Size: Letter
Paper Layout: Portrait Landscape

14. You can test your PDF and e-mail formatting using the preview options. Click on the **Send Test Email** link to send to the recipients the dashboard as it looks when the link is clicked. Then, click on the **Preview PDF** link to view a version of the PDF as it looks when the link is clicked.

15. Click on the **Save** button, and the PDF delivery of the dashboard is now scheduled.

How it works…

Since Splunk 5 was released, Splunk Enterprise has been natively able to produce PDFs of dashboards and reports. Prior to Version 5, it required a separate add-on app that only worked on Linux servers and required other operating system dependencies. The new integrated PDF features allow quicker and easier access to generate PDFs, either via a schedule and e-mailed or directly from the Web.

There are still some situations that are not able to produce PDFs, such as form-driven dashboards, dashboards created using advanced XML, and Simple XML dashboards that still contain Flash components. There are also some features, such as heat map overlays, that do not render properly.

PDFs are generated when requested by native libraries built into Splunk that render what would normally be output as HTML and encode this into the PDF. It's not an easy feat, as you have to take the page layout and orientation into consideration as the PDF is much more constrained than the browser window.

When delivering a scheduled PDF of a dashboard, you use the same mechanism that scheduled reports and alerts use.
The `sendemail`command is the backbone of the process and allows many different configuration options for the format of the message, including a full range of tokens that can be inserted into the subject and body of the messages that are replaced with job- and schedule-specific details.

TIP

For more information on the configuration options to schedule reports and dashboards, check out http://docs.splunk.com/Documentation/Splunk/latest/Report/Schedulereports.