Implementing Splunk Second Edition

Splunk is a type of analysis and reporting software for analyzing machine-generated Big Data. It captures, indexes, and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards, and visualizations. It aims to make machine data accessible across an organization for a variety of purposes.

Implementing Splunk Second Edition is a learning guide that introduces you to all the latest features and improvements of Splunk 6.2. The book starts by introducing you to various concepts such as charting, reporting, clustering, and visualization. Every chapter is dedicated to enhancing your knowledge of a specific concept, including data models and pivots, speeding up your queries, backfilling, data replication, and so on. By the end of the book, you will have a very good understanding of Splunk and be able to perform efficient data analysis.

Who this book is written for

If you are a data analyst with basic knowledge of Big Data analysis but no knowledge of Splunk, then this book will help you get started with Splunk. The book assumes that you have access to a copy of Splunk, ideally not in production, and many examples also assume you have administrator rights.

What you will learn from this book

- Enrich your data with lookups and commands
- Transform your data into useful and beautiful reports
- Build professional-looking, informative dashboards
- Get to know what Splunk data models and pivots are
- Learn about pivot editor, pivot elements, filters, Sparklines, and more
- Manage configurations from one to thousands of instances
- Extend Splunk with scripts and advanced configuration
- Create fields from your unstructured data
- Write searches that are fast and lean

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 2 'Understanding Search'
- A synopsis of the book’s content
- More information on *Implementing Splunk Second Edition*
About the Authors

Vincent Bumgarner has been designing software for over 20 years, working with many languages on nearly as many platforms. He started using Splunk in 2007 and has enjoyed watching the product evolve over the years.

While working for Splunk, he has helped many companies train dozens of users to drive, extend, and administer this extremely flexible product. At least one person in every company he has worked with has asked for a book, and he hopes that this book will help fill their shelves.

James D. Miller is an IBM-certified, accomplished senior engagement leader and application / system architect / developer / integrator with over 35 years of extensive application and system design and development experience. He has held positions such as National FPM practice leader, certified solutions expert, technical leader, technical instructor, and best practice evangelist. His experience includes business intelligence, predictive analytics, web architecture and design, business process analysis, GUI design and testing, data and database modeling and systems analysis, design, and development of applications, systems and models based on cloud, client/server, web and mainframe.

His responsibilities have included all aspects of solution design and development, including business process analysis and reengineering, requirement documentation, estimating and planning/management of projects, architectural evaluation and optimization, test preparation, and the management of resources. Other experience includes the development of ETL infrastructures—such as data transfer automation between mainframe (DB2, Lawson, Great Plains, and so on) systems and the client/server model-based SQL server—web-based applications, and the integration of enterprise applications and data sources.
In addition, he has acted as an Internet application development manager and was responsible for the design, development, QA, and delivery of multiple websites, including online trading applications, warehouse process control and scheduling systems, and administrative and control applications. Mr. Miller was also responsible for the design, development, and administration of a web-based financial reporting system for a $450-million organization, reporting directly to the CFO and his executive team.

In various other leadership roles, such as project and team leader, lead developer, and applications development director, Mr. Miller has managed and directed multiple resources using a variety of technologies and platforms.

James has authored *IBM Cognos TM1 Developer's Certification Guide* and *Mastering Splunk*, both by Packt Publishing and a number of whitepapers on best practices, such as *Establishing a Center of Excellence*. He continues to post blogs on a number of relevant topics based on personal experiences and industry best practices.

James also holds the following current technical certifications:

- IBM Certified Developer Cognos TM1
- IBM Certified Analyst Cognos TM1
- IBM Certified Administrator Cognos TM1
- IBM Cognos 10 BI Administrator C2020-622
- IBM Cognos TM1 Master 385 Certification
- IBM OpenPages Developer Fundamentals C2020-001-ENU
- IBM Certified Advanced Solution Expert Cognos TM1

His technology specialties include IBM Cognos BI and TM1, SPSS, Splunk, dynaSight/ArcPlan, ASP, DHTML, XML, IIS, MS Visual Basic and VBA, Visual Studio, PERL, WebSuite, MS SQL Server, Oracle, SQL Server on Sybase, miscellaneous OLAP tools, and so on.
Preface

Splunk is a powerful tool to collect, store, alert, report, and study machine data. This machine data usually comes from server logs, but it could also be collected from other sources. Splunk is, by far, the most flexible and scalable solution available to tackle the huge problem of making machine data useful.

The goal of the original version of this book was to serve as an organized and curated guide to Splunk 4.3. This version endeavors to preserve that objective, while focusing on the latest version (at the time of writing) of Splunk—6.2.0. In fact, care has been taken to call out the differences between the versions. In addition, new content has been added, covering search acceleration methods, backfilling, data replication, and Hunk.

As the documentation and community resources available to Splunk are vast, finding important pieces of knowledge can be daunting at times. My goal is to present what is needed for the effective implementation of Splunk in as concise and useful a manner as possible.

What this book covers

Chapter 1, The Splunk Interface, walks you through the elements of the user interface.

Chapter 2, Understanding Search, covers the basics of the searches, paying particular attention to writing efficient queries.

Chapter 3, Tables, Charts, and Fields, shows you how you can use fields for reporting and then covers the process of building your own fields.

Chapter 4, Data Models and Pivots, explains and defines Splunk data models and pivots, along with the pivot editor, pivot elements and filters, Sparklines, and more.
Preface

Chapter 5, *Simple XML Dashboards*, first uses the Splunk web interface to build our first dashboards. The chapter then examines how you can build forms and more efficient dashboards.

Chapter 6, *Advanced Search Examples*, walks you through examples of using Splunk's powerful search language in interesting ways.

Chapter 7, *Extending Search*, exposes a number of features in Splunk to help you to categorize events and act upon search results in powerful ways.

Chapter 8, *Working with Apps*, covers the concepts of an app, helps you in installing a couple of popular apps, and then helps you in building your own app.

Chapter 9, *Building Advanced Dashboards*, explains the concepts of advanced XML dashboards and covers practical ways to transition from simple XML to advanced XML dashboards.

Chapter 10, *Summary Indexes and CSV Files*, introduces the concept of summary indexes and shows you how they can be used to improve performance. It also discusses how CSV files can be used in interesting ways.

Chapter 11, *Configuring Splunk*, explains the structure and meaning of common configurations in Splunk. The chapter also explains the process of merging configurations in great detail.

Chapter 12, *Advanced Deployments*, covers common questions about multi-machine Splunk deployments, including data inputs, syslog, configuration management, and scaling up.

Chapter 13, *Extending Splunk*, demonstrates ways in which code can be used to extend Splunk for data input, external querying, rendering, custom commands, and custom actions.
Understanding Search

To successfully use Splunk, it is vital that you write effective searches. Using the index efficiently will make your initial discoveries faster, and the reports you create will run faster for you and for others. In this chapter, we will cover the following topics:

- How to write effective searches
- How to search using fields
- Understanding time
- Saving and sharing searches

Using search terms effectively

The key to creating an effective search is to take advantage of the index. The Splunk index is effectively a huge word index, sliced by time. The single most important factor for the performance of your searches is how many events are pulled from the disk. The following few key points should be committed to memory:

- **Search terms are case insensitive**: Searches for error, Error, ERROR, and ErRoR are all the same thing.
- **Search terms are additive**: Given the search item, mary error, only events that contain both words will be found. There are Boolean and grouping operators to change this behavior; we will discuss in this chapter under *Boolean and grouping operators*.
- **Only the time frame specified is queried**: This may seem obvious, but it's very different from a database, which would always have a single index across all events in a table. Since each index is sliced into new buckets over time, only the buckets that contain events for the time frame in question need to be queried.
- **Search terms are words, including parts of words**: A search for foo will also match foobar.
With just these concepts, you can write fairly effective searches. Let’s dig a little deeper, though:

- **A word is anything surrounded by whitespace or punctuation**: For instance, given the log line `2012-02-07T01:03:31.104-0600 INFO AuthClass Hello world. [user=Bobby, ip=1.2.3.3]`, the "words" indexed are `2012, 02, 07T01, 03, 31, 104, 0600, INFO, AuthClass, Hello, world, user, Bobby, ip, 1, 2, 3, and 3`. This may seem strange, and possibly a bit wasteful, but this is what Splunk’s index is really, really good at—dealing with huge numbers of words across a huge number of events.

- **Splunk is not grep with an interface**: One of the most common questions is whether Splunk uses regular expressions for your searches. Technically, the answer is no. Splunk does use regex internally to extract fields, including the auto generated fields, but most of what you would do with regular expressions is available in other ways. Using the index as it is designed is the best way to build fast searches. Regular expressions can then be used to further filter results or extract fields.

- **Numbers are not numbers until after they have been parsed at search time**: This means that searching for `foo>5` will not use the index, as the value of `foo` is not known until it has been parsed out of the event at search time. There are different ways to deal with this behavior, depending on the question you’re trying to answer.

- **Field names are case sensitive**: When searching for `host=myhost`, `host` must be lowercase. Likewise, any extracted or configured fields have case sensitive field names, but the values are case insensitive.
  - `Host=myhost` will not work
  - `host=myhost` will work
  - `host=MyHost` will work

- **Fields do not have to be defined before indexing data**: An indexed field is a field that is added to the metadata of an event at index time. There are legitimate reasons to define indexed fields, but in the vast majority of cases it is unnecessary and is actually wasteful. We will discuss this in Chapter 3, *Tables, Charts, and Fields*.

> It should be noted that some of the queries used throughout this book may yield different results or in some cases return no events since these searches were based on random data indexed by Splunk in our test environment.
Boolean and grouping operators

There are a few operators that you can use to refine your searches (note that these operators must be in uppercase to not be considered search terms):

- **AND** is implied between terms. For instance, `error mary` (two words separated by a space) is the same as `error AND mary`.
- **OR** allows you to specify multiple values. For instance, `error OR mary` means find any event that contains either word.
- **NOT** applies to the next term or group. For example, `error NOT mary` would find events that contain `error` but do not contain `mary`.
- **The quote marks (" )** identify a phrase. For example, "Out of this world" will find this exact sequence of words. Out of this world would find any event that contains all of these words, but not necessarily in that order.
- **Parentheses ( () )** is used for grouping terms. Parentheses can help avoid confusion in logic. For instance, these two statements are equivalent:
  - `bob error OR warn NOT debug`
  - `(bob AND (error OR warn)) AND NOT debug`
- **The equal sign (=)** is reserved for specifying fields. Searching for an equal sign can be accomplished by wrapping it in quotes. You can also escape characters to search for them. `\=` is the same as `"="`.
- **Brackets ([ ])** are used to perform a subsearch. We will discuss this in Chapter 6, Advanced Search Examples.

You can use these operators in fairly complicated ways if you want to be very specific, or even to find multiple sets of events in a single query. The following are a few examples:

- `error mary NOT jacky`
- `error NOT (mary warn) NOT (jacky error)`
- `index=myapplicationindex ( sourcetype=sourcetype1 AND ( (bob NOT error) OR (mary AND warn) ) ) OR ( sourcetype=sourcetype2 (jacky info) )`

This can also be written with some whitespace for clarity:

```plaintext
index=myapplicationindex
(
sourcetype=security
AND
()
```

[29]
Clicking to modify your search

Though you can probably figure it out by just clicking around, it is worth discussing the behavior of the GUI when moving your mouse around and clicking.

• Clicking on any word or field value will give you the option to Add to search or Exclude from search (the existing search) or (create a) New search:

• Clicking on a word or a field value that is already in the query will give you the option to remove it (from the existing query) or, as above, (create a) new (search):
Event segmentation

In previous versions of Splunk, event segmentation was configurable through a setting in the Options dialog. In version 6.2, the options dialog is not present – although segmentation (discussed later in this chapter under field widgets section) is still an important concept, it is not accessible through the web interface/options dialog in this version.

Field widgets

Clicking on values in the Select Fields dialog (the field picker), or in the field value widgets underneath an event, will again give us an option to append (add to) or exclude (remove from) our search or, as before, to start a new search.

For instance, if source="C:\Test Data\TM1ProcessError_20140623213757_temp.log" appears under your event, clicking on that value and selecting Add to search will append source="C:\Test Data\TM1ProcessError_20140623213757_temp.log" to your search:

To use the field picker, you can click on the link All Fields (see the following image):

[Image of the field picker with selected fields and options]

Selected Fields
- date_hour 10
- date_minute 38
- host 1
- source 100+
- sourcetype 100+

Interesting Fields
- date_mday 7
understanding search

Expand the results window by clicking on > in the far-left column. Clicking on a result will append that item to the current search:

If a field value looks like key=value in the text of an event, you will want to use one of the field widgets instead of clicking on the raw text of the event. Depending on your event segmentation setting, clicking on the word will either add the value or key=value. The former will not take advantage of the field definition; instead, it will simply search for the word. The latter will work for events that contain the exact quoted text, but not for other events that actually contain the same field value extracted in a different way.

Time

Clicking on the time next to an event will open the _time_ dialog (shown in the following image) allowing you to change the search to select Events Before or After a particular time period, and will also have the following choices:

- Before this time
- After this time
- At this time
In addition, you can select **Nearby Events** within *plus, minus, or plus or minus*, a number of seconds (the default), *milliseconds, minutes, hours, days, or weeks*:

![Nearby Events](image)

One search trick is to click on the time of an event, select **At this time**, and then use the **Zoom out** (above the timeline) until the appropriate time frame is reached.

**Using fields to search**

When we explored the GUI in Chapter 1, *The Splunk Interface*, you probably noticed fields everywhere. Fields appear in the field picker on the left and under every event. Where fields actually come from is transparent to the user, who simply searches for **key=value**. We will discuss adding new fields in Chapter 3, *Tables, Charts, and Fields*, and in Chapter 11, *Configuring Splunk*. 
Using the field picker
The field picker gives us easy access to the fields (currently defined) for the results of our query. Splunk will extract some fields from event data without your help, such as host, source, and sourcetype values, timestamps, and others. Additional fields to be extracted can be defined by you. Clicking on any field presents us with the details about that field in our current search results:

As we go through the following items in this widget, we see a wealth of information right away:

- **N Value, X% of events** is a good indication of whether we are getting the results we think we’re getting. If every event in your results should contain this field, and this is not 100 percent, either your search can be made more specific or a field definition needs to be modified. In addition, N Value indicates the number of unique values that this field contains.

- **Selected – Yes or No** indicates whether the field is selected (is part of the search query results) or not (simply listed as interesting additional fields found by Splunk within the data).

- **Reports – Top Values, Top Values by time, Rare values, and Events with this field**
  - **Top values** (overall) shows a table of the most common values for this field for the time frame searched.
  - **Top values by time** shows a graph of the most common values occurring in the time frame searched.
  - **Rare values** shows a table of the most unique values for this field for the time frame searched.
Events with this field adds `fieldname="*"` to your existing search to make sure that you only get the events that have this field. If the events you are searching for always contain the name of the field, in this case network, your query will be more efficient if you also add the field name to the query. In this case, the query would look like this: `sourcetype="impl_splunk_gen" network="*" network`.

- **Values** shows a very useful snapshot of the top ten most common values, **Count** is the number found for each of these values and % is the percentage that the value is found in this field in the results of the search.

## Using wildcards efficiently
Though the index is based on words, it is possible to use wildcards when needed, albeit a little carefully. Take a look at some interesting facts about wildcards:

- Only trailing wildcards are efficient: Stated simply, `bob*` will find events containing Bobby efficiently, but `*by` or `*ob*` will not. The latter cases will scan all events in the time frame specified.
- Wildcards are tested last: Wildcards are tested after all other terms. Given the search: `authclass *ob* hello world`, all other terms besides `*ob*` will be searched first. The more you can limit the results using full words and fields, the better your search will perform.

## Supplementing wildcards in fields
Given the following events, a search for world would return both events:

- 2012-02-07T1:04:31.102-0600 INFO AuthClass Hello world. [user=Bobby, ip=1.2.3.3]
- 2012-02-07T1:23:34.204-0600 INFO BarClass Goodbye. [user=Bobby, ip=1.2.3.3, message="Out of this world"]

What if you only wanted the second event, but all you know is that the event contains world somewhere in the field message? The query `message="*world*"` would work but is very inefficient because Splunk must scan every event looking for `*world`, and then determine whether world is in the field message.

You can take advantage of the behavior mentioned earlier—wildcards are tested last. Rewriting the query as `world message="*world*"` gives Splunk a chance to find all the records with world, and then inspect those events for the more specific wildcard condition.
All about time

Time is an important and confusing topic in Splunk. If you want to skip this section, absorb one concept—time must be parsed properly on the way into the index as it cannot be changed later without indexing the raw data again.

How Splunk parses time

If given the date 11-03-04, how would you interpret this date? Your answer probably depends on where you live. In the United States, you would probably read this as November 3, 2004. In Europe, you would probably read this as March 11, 2004. It would also be reasonable to read this as March 4, 2011.

Luckily, most dates are not this ambiguous, and Splunk makes a good effort to find and extract them, but it is absolutely worth the trouble to give Splunk a little help by configuring the time format. We'll discuss the relevant configurations in Chapter 11, Configuring Splunk.

How Splunk stores time

Once the date is parsed, the date stored in Splunk is always stored as GMT epoch. Epoch time is the number of seconds since January 1, 1970. By storing all events using a single time zone, there is never a problem lining up events that happen in different time zones. This, of course, only works properly if the time zone of the event can be determined when it is indexed. This numeric value is stored in the field _time.

How Splunk displays time

The text of the original event, and the date it contains, is never modified. It is always displayed as it was received. The date displayed to the left of the event is determined by the time zone of the Splunk instance or the user's preference, as specified in Your account.

<table>
<thead>
<tr>
<th>i</th>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>6/23/14 4:35:20 PM</td>
<td>&quot;451700&quot;,&quot;PCS-REVENUE CYCLE,&quot;Sold To Be Delivered&quot;,&quot;NA Metadata procedure line 13&quot;</td>
</tr>
</tbody>
</table>
How time zones are determined and why it matters

Since all events are stored according to their GMT time, the time zone of an event only matters at parse time, but it is vital to get it right. Once the event is written into the index, it cannot be changed without reindexing the raw data.

The time zone can come from a number of places, in the following order of precedence:

- The time zone specified in the log. For instance, the date 2012-02-07T01:03:23.575-0600, -0600 indicates that the zone is 6 hours behind GMT. Likewise, Tue 02 Feb, 01:03:23 CST 2012 represents the same date.

- The configuration associated with a source, host, or sourcetype, in that order. This is specified in props.conf. This can actually be used to override the time zone listed in the log itself, if needed. We will discuss this in Chapter 11, Configuring Splunk.

- The time zone of the Splunk instance forwarding the events. The time zone is relayed along with the events, just in case it is not specified elsewhere. This is usually an acceptable default. The exception is when different logs are written with different time zones on the same host, without the time zone in the logs. In that case, it needs to be specified in props.conf.

- The time zone of the Splunk instance parsing the events. This is sometimes acceptable and can be used in interesting ways in distributed environments.

- The important takeaway, again, is that the time zone needs to be known at the time of parsing and indexing the event.
Understanding Search

Different ways to search against time

Now that we have our time indexed properly, how do we search against time? The Date & Time Range picker provides a neat set of options for dealing with search times:

This picker widget is organized by:

- Presets
- Relative
- Real-time
- Data Range
- Date & Time Range
- Advanced

Let's take a look at understanding each of these.
Presets

Presets are time ranges that are pre-defined for you in Splunk Enterprise. You should be aware, though, that if you are searching potentially large amounts of data, results will return faster if you run the search over a smaller time period (rather than All time).

<table>
<thead>
<tr>
<th>Presets</th>
<th>Real-time</th>
<th>Relative</th>
<th>Other</th>
<th>All time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30 second window</td>
<td>Today</td>
<td>Last 15 minutes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 minute window</td>
<td>Week to date</td>
<td>Last 60 minutes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 minute window</td>
<td>Business week to date</td>
<td>Last 4 hours</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30 minute window</td>
<td>Month to date</td>
<td>Last 24 hours</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 hour window</td>
<td>Year to date</td>
<td>Last 7 days</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All time (real-time)</td>
<td>Yesterday</td>
<td>Last 30 days</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Real-time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Date Range</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Date &amp; Time Range</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Advanced</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relative

If the Relative presets are not what you need, you can use the custom Relative time range options to specify a time range for your search that is relative to now. You can select from the list of time range units, Seconds Ago, Minutes Ago, and so on:
Splunk also provides the ability to use **Beginning of second** (the default) or a **No Snap-to** time unit to indicate the nearest or latest time to which your time amount rounds up to. If you don't specify a *snap to* time unit, Splunk snaps automatically to the second. Unlike the **Presets**, to actually apply your (**Relative**) selections to the search, you need to click the **Apply** button.

**Real-time**

The custom **Real-time** option gives you the ability to set the start time for your real-time time range window. Keep in mind that the search time ranges for historical searches are set at the time at which the search runs. With real-time searches, the time ranges are constantly updating and the results accumulate from the beginning of your search.

You can also specify a time range that represents a sliding window of data, for example, the last 30 seconds.

When you specify a sliding window, Splunk takes that amount of time to accumulate data. For example, if your sliding window is 5 minutes, you will not start to see data until after the first 5 minutes have passed.
Windowed real-time versus all-time real-time searches

When designing your searches, it's important to keep in mind that there is a difference between Splunk real-time searches that take place within a set window (like 30 seconds or 1 minute) and real-time searches that are set to All time.

In windowed real-time searches, the events in the search can disappear as they fall outside of the window, and events that are newer than the time the search job was created can appear in the window when they occur.

In all-time real-time searches, the window spans all of your events, so events do not disappear once they appear in the window. But events that are newer than the time the search job was created, can appear in the window as they occur.

In comparison, in historical searches, events never disappear from within the set range of time that you are searching and the latest event is always earlier than the job creation time (with the exception of searches that include events that have future-dated timestamps).

Date range

You can use the custom Date Range option to add calendar dates to your search. You can choose among options to return events: Between a beginning and end date, Before a date, and Since a date (for these fields, you can either type the date into the text box, or select the date from a calendar).
Understanding Search

Date and time range
Use the custom Date & Time Range option to specify calendar dates and times for the beginning and ending of your search. Again, you can type the date into the text box or select the date from a calendar.

Advanced
Use the Advanced option to specify the earliest and latest search times. You can write the times in Unix (epoch) time or relative time notation. The epoch time value that you enter is converted to local time. This timestamp is displayed under the text field so that you can verify your entry.
Specifying time in-line in your search

You can also directly use relative and exact times in your searches. For instance, given the search item `bob error`, you can specify the time frame you want to use directly in the search, using the fields earliest and latest.

- To search for errors affecting `bob` in the last 60 minutes, use `earliest=-60m bob error`
- To search for errors affecting `bob` in the last 3 hours, snap to the beginning of the hour using `earliest=-3h@h bob error`
- To search for errors affecting `bob` yesterday, use `earliest=-1d@d latest=-0d@d bob error`
- To search for errors affecting `bob` since Monday midnight, use `earliest=-0@w1 bob error`

You cannot use different time ranges in the same query; for instance, in a Boolean search, `(earliest=-1d@d latest=-0d@d bob error) OR (earliest=-2d@d latest=-1d@d mary error)` will not work. The `append` command provides a way of accomplishing this.

_indextime versus _time

It is important to note that events are generally not received at the same time as stated in the event. In most installations, the discrepancy is usually of a few seconds, but if logs arrive in batches, the latency can be much larger. The time at which an event is actually written in the Splunk index is kept in the internal field `_indextime`.

The time that is parsed out of the event is stored in `_time`.

You will probably never search against `_indextime`, but you should understand that the time you are searching against is the time parsed from the event, not the time at which the event was indexed.

Making searches faster

We have talked about using the index to make searches faster. When starting a new investigation, the following few steps will help you get results faster:

1. Set the time to the minimum time that you believe will be required to locate relevant events. For a chatty log, this may be as little as a minute. If you don't know when the events occurred, you might search a larger time frame and then zoom in by clicking on the timeline while the search is running.
2. Specify the index if you have multiple indexes. It's good to get into the habit of starting your queries with the index name. For example, index=myapplicationindex error bob.

3. Specify other fields that are relevant. The most common fields to specify are sourcetype and host. For example, index=myapplicationindex sourcetype="impl_splunk_gen" error bob. If you find yourself specifying the field source on a regular basis, you could probably benefit from defining more source types. Avoid using the sourcetype field to capture other information, for instance datacenter or environment. You would be better off using a lookup against host or creating another indexed field for those cases.

4. Add more words from the relevant messages as and when you find them. This can be done simply by clicking on words or field values in events, or field values in the field picker. For example, index=myapplicationindex sourcetype="impl_splunk_gen" error bob authclass OR fooclass.

5. Expand your time range once you have found the events that you need, and then refine the search further.

6. Disable Field discovery in earlier versions of Splunk - there was a toggle at the top of the field picker. In version 6.2, the feature is a bit different. You can simply open the field picker and use the Select All Within Filter or Deselect All checkbox to remove any unneeded fields from the list that Splunk will extract. This can greatly improve speed, particularly if your query retrieves a lot of events. Extracting all the fields from events simply takes a lot of computing time, and disabling this option prevents Splunk from doing all that work when not needed. Take a look at the following screenshot:

<table>
<thead>
<tr>
<th>Select Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select All Within Filter</td>
</tr>
<tr>
<td>t</td>
</tr>
</tbody>
</table>

If the query you are running is taking a long time to run, and you will be running this query on a regular basis—perhaps for an alert or a dashboard—using a summary index may be appropriate. We will discuss this in Chapter 10, Summary Indexes and CSV Files.
Sharing results with others

It is often convenient to share a specific set of results with another user. You could always export the results to a CSV file and share it, but this is cumbersome. In earlier versions of Splunk, a URL could be saved and shared; in version 6.2, things are a bit different (although you still can save your search as a bookmarked URL).

The URL

To share your search as a bookmarked URL, you can click on the share icon to view the Share Job dialog:

From here, you can simply right-click on the share icon and bookmark your search for later use:
Understanding Search

You can also share your search and search results in a variety of other ways, starting by clicking on the **Save As** link:

![Save As Options](image)

This lists your options for saving the search and search results. Your choices are the following:

- Report
- Dashboard Panel
- Alert
- Event Type

**Save as report**

To save your search as a report, click on the **Report** link. This opens the **Save As Report** dialog:

![Save As Report Dialog](image)
From here, you need to do the following:

1. Enter a Title (or name) for your report.
2. Enter an optional Description to remind users what your report does.
3. Indicate if you'd like to include the Splunk Time Range Picker as a part of your report.

Once you click Save, Splunk prompts you to either review Additional Settings for your newly created report (Permissions, Schedule, Acceleration, and Embed), Add (the report) to Dashboard (we will talk more about dashboards in Chapter 5, Simple XML Dashboards), View the report, or Continue Editing the search:

In my example, I named my report My Error Report, added a description (a simple example of a save as report), and included the Time Range Picker. The following screenshot displays the saved report after clicking View:
The additional settings that can be made to the report are given as follows:

- **Permissions**: Allows you to set how the saved report is displayed: by owner, by app, or for all apps. In addition, you can make the report read only or writable (can be edited).

- **Schedule**: Allows you to schedule the report (for Splunk to run/refresh it based upon your schedule). For example, an interval like every week, on Monday at 6 AM, and for a particular time range.

- **Acceleration**: Not all saved reports qualify for acceleration and not all users (not even admins) have the ability to accelerate reports. Generally speaking, Splunk Enterprise will build a report acceleration summary for the report if it determines that the report would benefit from summarization (acceleration). More on this topic later in Chapter 2, *Understanding Search*.

- **Embed**: Report embedding lets you bring the results of your reports to large numbers of report stakeholders. With report embedding, you can embed scheduled reports in external (non-Splunk) websites, dashboards, and portals. Embedded reports can display results in the form of event views, tables, charts, maps, single values, or any other visualization type. They use the same formatting as the originating report. When you embed a saved report, you do this by copying a Splunk generated URL into an HTML-based web page.
Save as dashboard panel

We'll be discussing dashboards in Chapter 5, Simple XML Dashboards but, for now, you should know that you can save your search as a new dashboard or as a new panel in an existing one. Permissions can also be set:
Save as alert

An alert is an action that a saved search triggers based on specified results of the search. When creating an alert, you specify a condition that triggers the alert (basically, a saved search with trigger conditions). When you select Save as Alert, the following dialog is provided to configure search as an alert:

![Save As Alert dialog](image)

Save as event type

Event types are a categorization system to help you make sense of your user-defined data fields. It simplifies searches by letting you categorize events. Event types let you classify events that have common characteristics. When your search results come back, they're checked against known event types. An event type is applied to an event at search time if that event matches the event type definition.

The simplest way to create a new event type is through Splunk Web. After you run a search that would make a good event type, click Save As and select Event Type. This opens the Save as Event Type dialog, where you can provide the event type name and optionally apply tags to it:
Search job settings

Once you run a search, you can access and manage information about the search job (an individual instance of a running or completed search, pivot, or report, along with its related output) without leaving the Search page. This is done by clicking Job and choosing from the available options:
You can also perform the following tasks:

- **Edit Job Settings**: Select this to open the Job Settings dialog, where you can change the job's read permissions, extend the job's lifespan, and get a URL for the job which you can use to share the job with others. You can also put a link to the job in your browser's bookmark bar.

- **Send Job to Background**: Select this if the search job is slow to complete and you would like to run the job in the background while you work on other Splunk Enterprise activities (including running a new search job).

- **Inspect Job**: Opens a separate window and displays information and metrics for the search job via the Search Job Inspector.

- **Delete Job**: Use this to delete a job that is currently running, is paused, or which has finalized. After you have deleted the job, you can still save the search as a report.

### Saving searches for reuse

As an example, let's build a search query, save it (as a report), and then make an alert out of it. First, let's find errors that affect *mary*, one of our most important users. This can simply be the query `mary error`. Looking at some sample log messages that match this query, we see that some of these events probably don't matter (the dates have been removed to shorten the lines).

```
ERROR LogoutClass error, ERROR, Error! [user=mary, ip=3.2.4.5]
WARN AuthClass error, ERROR, Error! [user=mary, ip=1.2.3.3]
ERROR BarClass Hello world. [user=mary, ip=4.3.2.1]
WARN LogoutClass error, ERROR, Error! [user=mary, ip=1.2.3.4]
DEBUG FooClass error, ERROR, Error! [user=mary, ip=3.2.4.5]
ERROR AuthClass Nothing happened. This is worthless. Don't log this. [user=mary, ip=1.2.3.3]
```

We can probably skip the DEBUG messages; the LogoutClass messages look harmless, and the last message actually says that it's worthless. *mary error NOT debug NOT worthless NOT logoutclass* limits the results to:

```
WARN AuthClass error, ERROR, Error! [user=mary, ip=1.2.3.3]
ERROR BarClass Hello world. [user=mary, ip=4.3.2.1]
```

For good measure, let's add the sourcetype field and some parentheses.

```
sourcetype="impl_splunk_gen" (mary AND error) NOT debug NOT worthless NOT logoutclass
```
Another way of writing the same thing is as follows:

```
sourcetype="impl_splunk_gen" mary error NOT (debug OR worthless OR logoutclass)
```

So that we don't have to type our query every time, let's go ahead and save it as a report for quick retrieval.

First, choose **Save As...**, and then, **Report**.

The **Save As Report** window appears.
Enter a value for **Title**, in our case, errors affecting **Mary**. Optionally, we can add a short description of the search. The time range is filled in based on what was selected in the time picker, and we decide to include the **Time Range Picker** in the saved report. Click **Save**.

Once we see the preceding window (**Your Report Has Been Created**), we click on **Permissions** and see the Edit Permissions window:
For **Display For**, let's click on **App** (rather than the default **Owner**, as shown in the preceding screenshot):

![Edit Permissions Window]

Next, we'll check **Read** for all user roles except for **power**, since we know that certain users in our Splunk environment are members of this group (including our friend **mary**). Finally, we can click **Save**.
Understanding Search

The search report is then available under Reports:

Selecting search/report from the menu runs the search using the latest data available.

Creating alerts from searches

Let's continue with our example. We want to take our original search query, schedule it, and then set a triggered response.

Any saved search can also be run on a schedule. One use for scheduled searches is firing alerts. Let's get started with our example. Go to the Reports page (shown in the previous screenshot) and click on Open in Search for our report (errors affecting mary). This opens our saved report not as a report but as a search query (it also runs the search). From there, we can click on Save As and choose Alert:
Chapter 2

Using the **Save As Alert** window (shown in the next screenshot), we can fill in the appropriate details for our alert:

![Save As Alert window](image)

- **Title**: I kept the original search title (errors affecting mary) but added the word *alert*
- **Description**: I kept this the same, but in reality, we'd want to add more of a description
- **Alert Type**: I selected **Scheduled**, since I want this alert search to be run every day
- **Time Range**: I selected the preset **Run every day**
- **Schedule At**: I selected the preset **12:00**
- **Trigger condition**: I selected the preset **Number of Results** since I'd like to trigger an event if my search finds any errors generated by our favorite user, *mary*
- **Trigger if number of results**: I selected the preset **Is Greater than** and filled in zero (this means that I am interested in any errors that are found by my search)
After filling in the above, I can click on Next; we can see that we have more information to provide:

This time, the window is divided into the following areas: Enable Actions, Action Options, and Sharing.

**Enable actions**
- **List in Triggered Alerts**: You can check this if you want to display your triggered alert in the Splunk Alert Manager which lists details of triggered alerts for 24 hours or a specified duration
- **Send Email**: You can configure your alert to send an e-mail to specified users when the alert gets triggered
- **Run a Script**: You can have Splunk run a script when your alert gets triggered
Action options

- **When triggered, execute actions**: *Once* or *For each result*. For example, should the alert trigger for each error that Mary receives or once for all errors within a time range?

- **Throttle?**: You can use throttling (usually based upon time and/or event count) to reduce the frequency at which an alert triggers since an alert can trigger frequently based on similar results that the search returns or the schedule to run the alert.

Sharing

**Permissions — Private or Shared in App.** Should this alert be shared with other users?
Understanding Search

For our example, I've elected to trigger an e-mail to mary (marys@slunker.com) with a link to both the alert and the alert results within the e-mail so that she can review her errors. In addition (as shown in the next screenshot), I have decided to send an e-mail **Once** (for all events/errors within the time range, not for each one) and leave the alert **Private**.

![Alert settings screenshot]

After hitting **Save**, our alert is ready to go:

![Alert details screenshot]

**Summary**

In this chapter, we covered searching in Splunk and doing a few useful things with those search results. There are lots of little tricks that we will touch upon as we go forward.

In the next chapter, we will start using fields for more than searches; we'll build tables and graphs, and then, we'll learn how to make our own fields.
Where to buy this book
You can buy Implementing Splunk Second Edition from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.